

UNIVERSITY OF PISA  
FACULTY OF MATHEMATICAL, PHYSICAL  
AND NATURAL SCIENCES

Master's degree in computer science

Master thesis



**Data-driven exploration of  
mobility interaction patterns**

Candidate  
Gabriele Galatolo

Supervisor

Dr. Mirco Nanni

Co-examiner

Prof. Laura Ricci

Accademic Year 2012/2013



To my family

To Silvia



## SOMMARIO

---

In questa tesi proponiamo un framework per lo studio delle interazioni tra oggetti mobili, come ad esempio auto sulle strade o pedoni in movimento all'interno di piazze. Abbiamo seguito un approccio caratteristico del data mining, basato sulla computazione di semplici eventi di *interazione* e sull'estrazione di pattern complessi che possano descrivere quali siano le combinazioni frequenti di eventi che compaiono nello stesso istante e la loro evoluzione nel tempo. Il lavoro include due casi di studio su dataset reali, rispettivamente su veicoli che percorrono un tratto autostradale e su pedoni in movimento su una piazza.



## ABSTRACT

---

In this thesis we propose an analysis framework for studying the interactions between moving objects, such as cars on roads, pedestrians in a square, etc. We follow a data mining approach, based on the computation of simple *interaction* events and on the extraction of complex patterns, describing frequent combinations of events that happen together and their evolution in time. The work includes two case studies on real datasets, respectively on cars and pedestrians.





*Ohana means family.  
Family means nobody gets left behind, or forgotten.*

— Lilo & Stitch

*If i have seen further  
it is by standing on the shoulders of the giants.*

— Isaac Newton

## ACKNOWLEDGEMENTS

---

Finally i'm here to write again this section, after three years from the last time. As the last time i would to write a lot on everyone who shared with me something along this years, but i risk to miss someone or to be not so exhaustive with all of you as i would. For someone, however, i'll do an exception.

The first person i want to thank is my supervisor, Mirco Nanni, who drove me through this fantastic experience. I hope to have done a good job: if it has been so, is certainly thanks to him.

An important acknowledgement goes to my family, who supported me through my academic career. Without them i wouldn't be here, and i wouldn't be who i am.

Obviously a special thank goes to Silvia, who tolerated me during *periods of examinations* (even if sometimes happened the contrary) but, in particular, who gave me the force to break down my limits when i was dispirited.

Last, but not least: when i downloaded the template for this thesis, in the dedication part there was a sentence taken from Lilo&Stich, the recall to the *ohana*, the idea of family that contains not only the natural one but also the friends and all the relationships important for someone. I think is not a case that this was contained at the beginning of this template, because it is thanks to you that this has been possible. Is thanks to my *ohana* that i feel quiet to deal with anything, and where i feel safer: if i could do this is by standing on your shoulders, on the shoulders of the giants.

So, thanks.

Gabriele



# CONTENTS

---

<b>i</b>	<b>INTRODUCTION AND BACKGROUND</b>	<b>1</b>
<b>1</b>	<b>MOTIVATIONS AND OBJECTIVES</b>	<b>3</b>
1.1	Interactions in a mobility context	4
1.2	Organization of the Thesis	8
1.3	Novel contributions	8
<b>2</b>	<b>RELATED WORKS AND BACKGROUND</b>	<b>11</b>
2.1	Related works	11
2.1.1	Agents movement modelling/simulation with physical models	11
2.1.2	Simulate pedestrians moving with Agent Based Models	14
2.1.3	Interactions for complex system modelling	16
2.1.4	Reality mining	17
2.2	Data-mining background	19
2.2.1	Apriori principle and association rules mining	19
2.2.2	Graph mining	21
2.3	Graph theory	22
2.3.1	Multigraphs	24
2.3.2	Graphs isomorphism problem	25
2.3.3	Temporal graphs	26
<b>ii</b>	<b>MINING DATA WITH INTERACTION PATTERNS</b>	<b>29</b>
<b>3</b>	<b>THE INTERACTION PATTERN ANALYSIS FRAMEWORK</b>	<b>31</b>
3.1	Preliminaries	32
3.2	The IPA Framework	33
3.2.1	Representing trajectories with a graph	34
3.2.2	Neighborhood creation	35
3.2.3	Measuring interactions	37
3.2.4	IPA Events	40
<b>4</b>	<b>THE STATIC INTERACTION PATTERN MINING ALGORITHM</b>	<b>45</b>
4.1	Instance patterns and interaction patterns	46
4.2	Equivalence between interaction patterns	50
4.3	The SIPM algorithm	53
<b>5</b>	<b>THE EVOLVING INTERACTION PATTERNS MINING ALGORITHM</b>	<b>59</b>
5.1	Evolving interaction patterns	60
5.2	The EvIPM algorithm	64
<b>6</b>	<b>EXPERIMENTS</b>	<b>67</b>
6.1	Analysis on NGSIM vehicle dataset	67

6.2	Analysis on a campus square dataset	73
6.3	SIPM and EvIPM performances	76
iii	CONCLUSIONS AND FUTURE WORK	81
7	CONCLUSIONS	83
7.1	Future works	83
7.2	Summary of results	85
	BIBLIOGRAPHY	87

## Part I

### INTRODUCTION AND BACKGROUND

In the following chapters we introduce the overall work of thesis: we first introduce the key ideas that drove our work, with a discussion on some possible target applications and research developments. Then we introduce some important concepts about the techniques and methodologies used in the state-of-art for the mobility data mining, finally classifying our work with respect to the current literature about mobility data analysis.



## MOTIVATIONS AND OBJECTIVES

---

In the last years the topic of big data and related ones followed an exponential growth, both for the increase of the devices from which we can accumulate data about peoples' behaviour (GPS, smart-phone, tablet, etcetera) and also for the increase of interest in the discovery of information from big amount of data which have a lot of information hidden and not immediately visible. Data mining research and applications have become an important field to discover important statistical facts that we can use to simplify our life, improve our security or help to fight some social plagues, as the tax evasion phenomena.

There are a lot of examples of applications and techniques that data mining has led: one of the most famous is the basket-market analysis, which has been firstly used to discover habits and profiles of customers to improve the productivity and the targeted selling. Another example is the study of clustering techniques and the applications based on them, as the DIVA project [34] that has been studied and developed by the Italian National Council of Research to find potential tax evaders. One recent field is the mining on mobility data where, for instance, the study of trajectories of moving cars can be analyzed to improve the existing major road or also to identify new ways of transportation to optimize vehicles movement, like the car-pooling service.

Mobility data mining is the topic on which we have focused our attention. In this field a lot of research has been done [17] to answer to several interesting questions that involve various aspects of data-mining as:

### FINDING GROUP OF AGENTS WITH COMMON CHARACTERISTICS

this task involve the evolution of clustering techniques to find groups of moving agents that share some important spatio-temporal characteristics, as the time or the street of journey;

EXTRACTION OF FREQUENT PATTERNS this topic is focused on the extraction of rules that can describe the behaviour of a subset of agents, where the found rules are verified for a considerable amount of agents;

PREDICTION OF LOCATIONS AND TRAJECTORIES this is the task to predict information about the time and the spatial position in which an agent could be, studying its previous behaviours and trajectories.

In this work we focused on the extraction of frequent patterns: in particular we want to start from a set of trajectories and, studying the *interactions* that happen between agents, to bring out frequent patterns that could describe the most common behaviours in the area of study considered.

### 1.1 INTERACTIONS IN A MOBILITY CONTEXT

We interact continuously with the external world: when we move or we talk with other persons, when we drive our car acting on car's commands and other examples like these. Sometimes we interact directly with the objects or with the environment, as when we are pushing a box in some direction, but in other cases we have an indirect influence on the others. Suppose that a person is moving and another person is walking in the opposite direction: in a certain moment, to avoid a collision, the person in front of the first can decide to take another direction to turn in order to avoid him. The decision of the second agent to choose another direction is indirectly suggested by some interactions between him and the first that we do not see but that exist, interactions that we could summarize as:

- A. the direction of the second person is opposite to the first one;
- B. the first person is in front of the other one;
- C. the distance between them is decreasing at each instant;
- D. the difference of velocity can suggest to the second agent the intention of the other to not move from its initial direction

Thus we can consider interactions as made of measurable quantities returned from a set of functions. Those values represent some possible influences that an agent could have on the others into the environment, influences that manifests themselves in form of events. Even if we have exemplified what we mean with interactions, we did not answer to some important questions: in which way those interactions clarifies what is happening between two agents? Is the set of interactions that identify the ongoing events between agents attributable to some



well-known, or frequent, behavioural pattern? Are those patterns related to other ongoing events?

To better explain how we will use the interactions and what we want to discover from data with our work, we imagine this example: consider two car, with identifiers 1 and 2, shown in figure 1.

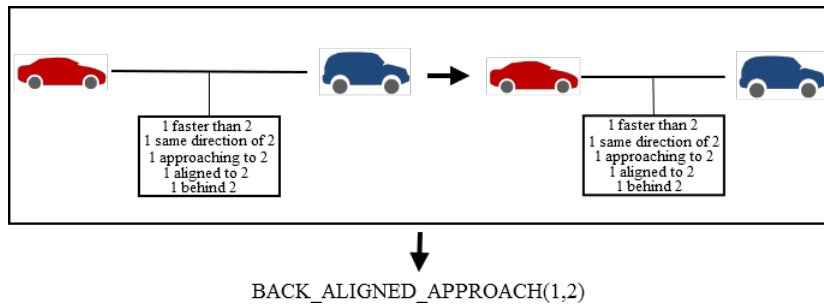


Figure 1: Two consecutive instants in which interactions, indicated in the rectangles, are formulated as differences of values of the attributes of moving of the cars.

Assuming that two neighbor agents influence each other, if we could define a set of functions that compute values about the situation between agents (in the example, difference of velocity and direction, alignment and position of the cars), we would be capable also to recognize a set of events, where each of them can be described by a specific subset of interactions repeated in time. In the example shown in figure 1, if in two consecutive instants of time we find that 1 is aligned to 2 and is approaching it, we can state that 1 is making an aligned approach on 2 that has duration 2 instants: furthermore we can also state that 1 is the subject of the event because is faster than 2.

Once we can retrieve all the events we have defined in our events' library, we could ask if there exist some patterns describing frequently seen behaviours, in function of these events. Those patterns can tell us if there are behaviours that are not immediately recognizable or that does not have a name that directly identify the pattern, as could be a group movement in which several agents maintain the distance each other while moving in the same direction. For example, combining the events found with the study of the interactions between agents, we may find a situation as the one exemplified in figure 2.

This pattern has not an explicit semantic explanation, but could simply be something that is frequently found in data, having so a statistical significance.

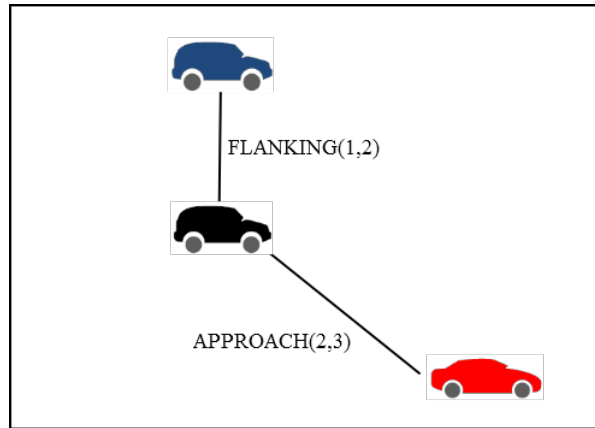
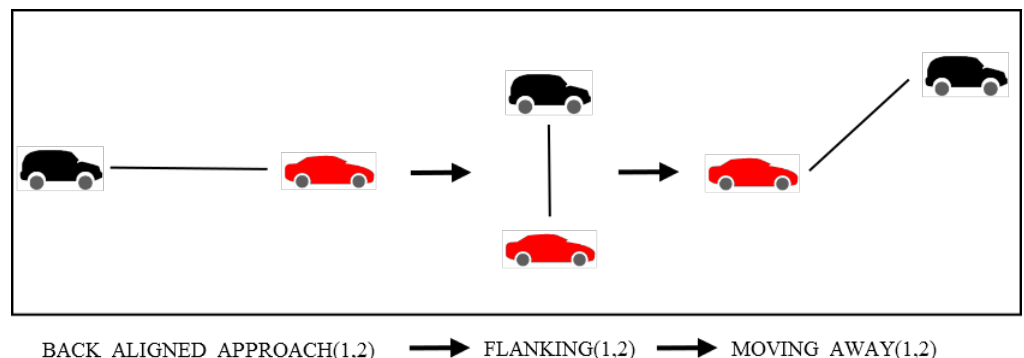


Figure 2: A complex pattern that describes three agents involved in two events happening at the same time, or in an overlapped interval of time.

When we have found these *instance patterns* we may do more: we could ask if there are sequences of those patterns that share a subset of agents frequently seen in data. This can give to the dataset owner important information regarding the possible evolutions of groups of events that characterize the studied area, but also where they are concentrated and if they have interesting time distributions. A simple sequence of instances patterns that one expect to find in a dataset in which the agents are vehicles or trucks is the one shown in figure 3.



BACK\_ALIGNED\_APPROACH(1,2)  $\longrightarrow$  FLANKING(1,2)  $\longrightarrow$  MOVING\_AWAY(1,2)

Figure 3: An overtaking between two cars.

However we are interested to study a methodology to describe not only those intuitive sequences, but to find all patterns that characterize a dataset where are involved moving agents, once defined the proper interactions and definitions of events. Furthermore we want to describe those patterns in a more general way: so, if for example a lot of different agents are involved in the same scheme as the one represented in figure 2, we want state that a frequent behaviour between cars' drivers is the one described by  $\text{flanking}(A, B) - \text{approach}(B, C)$ , where  $\{A, B, C\}$  represent three placeholders for real agents.

In figure 4 we show the scheme representing the whole process that we developed with this thesis. We will take this scheme every time we will need to explain at which level we will be and what we have to do to reach the final outputs, the **static interaction patterns** and the **evolving interaction patterns**.

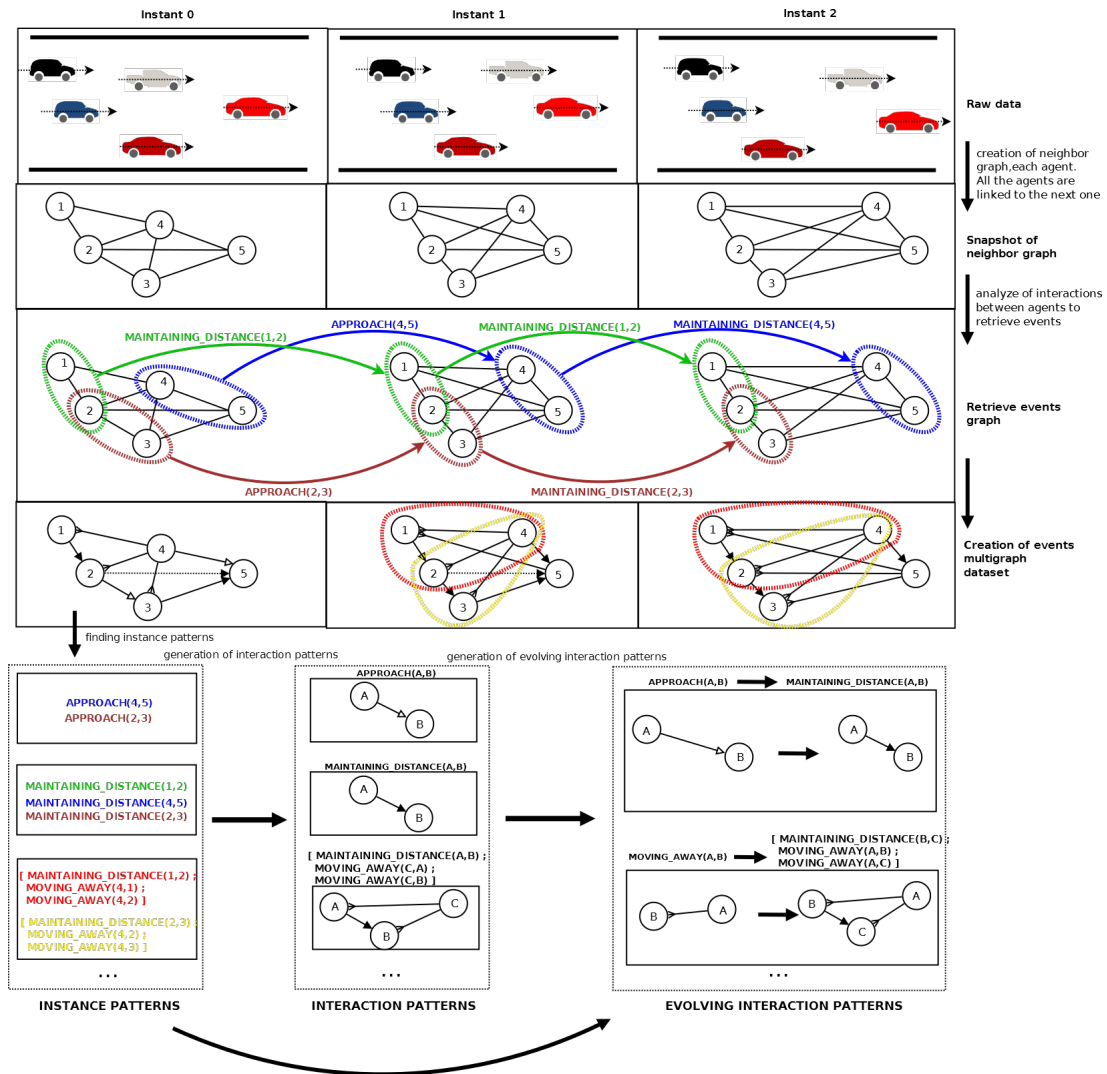


Figure 4: Schematic summary of the whole process that we developed.

Starting from raw data we will create a structure to explicit the relations between moving agents; from this structure we want to extract all the events which involve the agents in a neighborhood relation; from those events, with a further abstraction of the first representation, we want to extract a set of general patterns indicating the most common behaviours found in the dataset (in function of the previously defined events) and the most frequent evolutions within them that characterize the dataset.

## 1.2 ORGANIZATION OF THE THESIS

The thesis is organized in three parts. In the first we have done a brief introduction and in the next section we will introduce some methodologies and techniques that we have used or that we will refer in the rest of the thesis; furthermore we will make an overview on the concepts of interaction analysis, as they have been treated in literature, in particular in the fields of transportation/pedestrian movement simulation, and into the modelling/analysis of complex systems.

The second part will be focused on interaction patterns: first of all we will introduce the concepts and tools that constitute the interaction patterns analysis framework; the main task of the framework is to extract from data the abstract objects that will be used to construct and define the notion of *static instance patterns*, which are concrete groups of events with specific features, where there are involved real agents. These represent instantiations of the static interaction patterns, schemes of contemporary events that are combined between them, but where the agents are substituted by variables, because represent frequent generic schemes, i.e. schemes whose instances grouped together verify a temporal support threshold. Furthermore we will be interested to the temporal evolution of these frequent interaction patterns. To find these objects and to study their temporal evolution, we will develop and discuss two algorithms, the static interaction patterns miner (SIPM) and the evolving interaction patterns miner (EvIPM). Afterword we will present the experiments we made on two real case studies, one regarding a dataset of moving vehicles on a section of an US highway, and one relative to a small campus square taken from the data of a surveillance video. We will discuss the potential use of the developed instruments, and the possible analysis that can be done with several examples taken from the result of our experiments.

In the last section we will discuss the possible future improvements for what has been done until now, and we will analyze the methodologies used comparing them with the results obtained in the experiments.

## 1.3 NOVEL CONTRIBUTIONS

During this work some new tools and notions were introduced to reach our final goal. In particular we have newly introduced:

- A. a definition of a general framework for the retrieval of events described with a set of continuous interactions between agents, the IPA framework;
- B. the concept and definition of interaction pattern as the complex object that allows to describe a frequent behaviour into the dataset, describing it as a generalization of concrete groups of events that emerge from the data;
- C. development of the SIPM algorithm, that extract frequent interaction patterns from a set of events;
- D. development of the EvIPM algorithm to find the temporal evolution of the interaction patterns found with the SIPM algorithm.



## RELATED WORKS AND BACKGROUND

---

As every work, an important part of our time has been spent to the definition of the collocation of our work into the existing literature, the retrieving of material that could be useful to better define what we want to do and to understand or face some problems that could be useful into the definition of the methodologies and techniques that we developed. So, before starting to explain our work, we present in this chapter two fundamental things.

In the first part we present some works of the literature related to ours. In particular we will talk about some of the works related to the simulation and to the modelling of moving pedestrians/agents, and how the interactions are treated by the authors: furthermore we will see two works about the modelling of complex systems that have a more closer relation to our idea of interaction.

In the second part we will expose briefly some important concepts that we will use in the following, to better explain what are the problems encountered and what are the basis on which we worked. In particular we will make an overview on some basic concepts of graph theory, that will be fundamental in the phase of framework definition, framework that we will use to analyze interactions between agents, and for the definition of the interaction patterns; we will see some basic notions behind the basket market analysis, in particular into the definition and the use of the Apriori principle; finally we make a brief introduction to the graph mining, the part of data mining that study how to mine frequent substructures from graphs datasets, since we will use graphs and we want take from those graphs specific types of frequent patterns.

### 2.1 RELATED WORKS

#### 2.1.1 *Agents movement modelling/simulation with physical models*

One of the topic that has focused the interest of the scientists in the simulation of crowds and moving agents in general, is the one that refers to the dynamics of a crowd in a panic situation. There are a lot of cases in which the study of the behaviour of

a crowd in those situations can help to save several lives and to better understand how to address very huge crowds to assume the correct behaviour that can limit the risks for everybody. An important work that tackle this problem is the one published by Helbing, Farkas and Vicsek[3]. This work has been derived from previous studies on the *social force models* [2], which try to model the movements of an agent using similarities with physical forces and models. In this work the authors try to tackle the problem of the crowd's behaviour modelling in an emergency situation, where panic and uncoordinated motions of pedestrians are the main features of the environment. Empirical and scientific motivations have been led the authors to model the crowd as a particles system guided by gas or fluid laws. Each pedestrian has associated a function to compute the variation in its velocity that can be defined with the equation 1.

$$m_i \frac{d\mathbf{v}_i}{dt} = m_i \frac{v_i^0(t) \mathbf{e}_i^0 - \mathbf{v}_i(t)}{\tau_i} + \sum_{j(\neq i)} \mathbf{f}_{ij} + \sum_W \mathbf{f}_{iW} \quad (1)$$

In the previous equation the pedestrian with mass  $m_i$  that wants to move in the direction  $\mathbf{e}_i^0$  with a certain desired speed  $v_i^0$  tends to adapt its velocity to the current one ( $\mathbf{v}_i(t)$ ) in a certain time  $\tau_i$ . The interactions, with other agents  $j$  and with the walls around him  $W$ , are defined with the functions  $\mathbf{f}_{ij}$  and  $\mathbf{f}_{iW}$ . Each function  $\mathbf{f}$  is based on some similarity with physics phenomenons: for instance, the interaction that establish the tendency of two pedestrians to stay away from each other is modelled with a formula (called in the paper *repulsive interaction force*) very similar to the Coulomb's law, which express the electrostatic interaction between two electrically charged particles. In figure 5 we show an image taken from the experiments section of the article [3].

In particular, in this simulation, each pedestrian moves with an initial velocity of  $v_0^i = 5$  m/s choosing the desired direction randomly: the equation that describe the movement contains a panic factor  $p$  that influence each pedestrian to follow a neighbor, to make its own choice or mix both. Into the plots are shown the effects of the behaviour of the crowd varying the panic parameter. Keeping aside the implementation of the functions, is important to note that the equation takes into account the interactions of one agent as the sum of all the interactions between pairs of agents and between the pairs constituted by the current agent with each of the obstacles in the environment, the wall in this case. This is an important assumption that we will maintain in our thesis. However, into the cited work, those assumptions model well the behaviour of a crowd in a panic situation, and the authors themselves recognize the need to find a



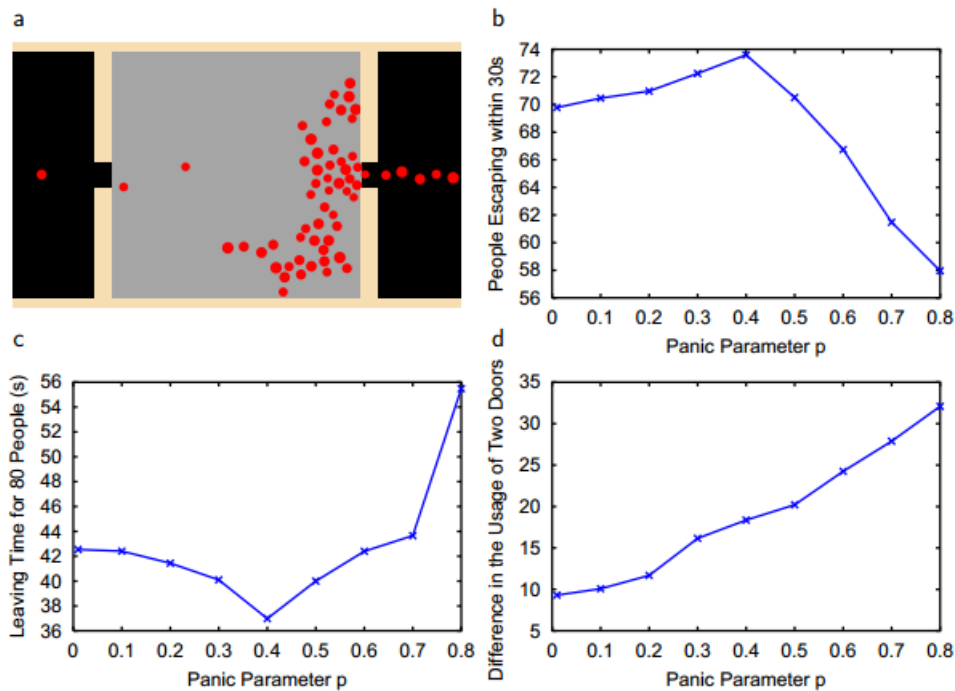


Figure 5: Simulation of 90 pedestrians trying to escape from a room: in the figure (a) a snapshot of the simulation, in the plots the variation of leaving time, difference of usage of the doors and the people escaping, varying a parameter simulating the panic in the crowd.

theory or model that can add into this representation interpersonal interactions, or other interesting interactions between the pedestrians and the environment.

Other works based on the previous one, like [4], try to tackle two problems that afflict the one presented by Helbing:

- A. retain the realism of the original model;
- B. create the basis for a more realistic model for situation in which there are a low number of pedestrian.

Thus, in these works the attention is focused on the fact that the social forces do not have physical sources and try to refine them, including other aspects as the density of the crowd in the current pedestrian's position, the distinguishing between face to back and face to face repulsion with other agents and so on.

In particular, trying to increase the realism detail of the simulation, we can see in figure 6 an example of problem tackled by Lakoba and Filkenstein: in the new formulation they include into the social force model some modifications to take into account the density of the crowd that, in a real case, can be at the base of a choice respect to others. As we can see from figure 6, the introduction of this parameter studied in that work leads

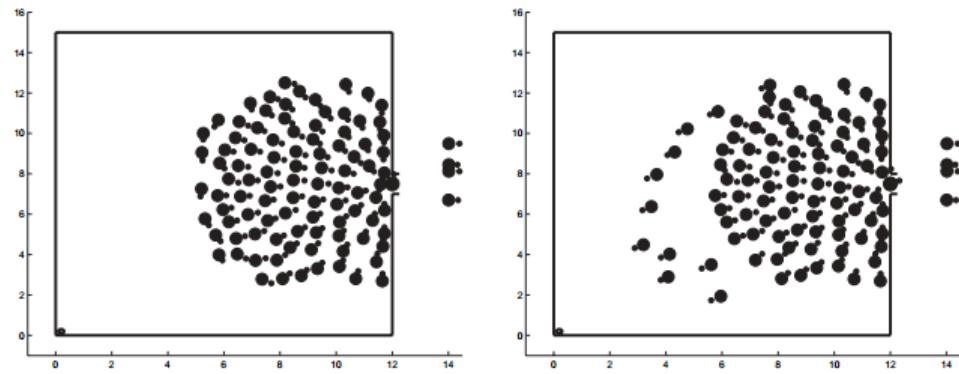


Figure 6: The effect of the crowd on the outer pedestrians: in the left they first try to open a way in the crowd but after, in the right plot, they run away as a group.

to more realistic reactions: in fact initially the pedestrians in the outer part of the crowd try to move themselves to the exit, but considering the crowd density they choose to turn back and run away. Furthermore there are some works in which the authors introduce the study of social force models in a more behavioural way to see how pedestrians react to the external stimulus. In [10], the model presented include an *interaction law*, stills inspired to the physical laws, which has composed by a set of parameters fitted with real experiments. However those models are still too dependant to a physical inspiration to give very realistic simulations in all contexts.

### 2.1.2 Simulate pedestrians moving with Agent Based Models

The previous type of models are not the unique ones: others, as the one presented in [6], try to explicit the movements of the agents with particular models called Agent Based Model (ABM). These are well-known models [7] that are made of a set of agents, each of one makes its decision individually analyzing the current status of the agent and a set of predefined rules. One of the main differences between those models and the pure mathematical ones is given by the fact that in ABM we can integrate the mathematical models with a set of rules that are involved into the decision of the move for an agent, separated from the ones that manage the physics of the movements.

In the cited work the author focus its attention on the fact that physical modelling only is not appropriate to describe the behaviour of pedestrians in low-density scenarios, and wants develop an instrument with which he could describe several scenarios without limitation on a minimum/maximum number of agents. The model he defined is divided in three types of be-

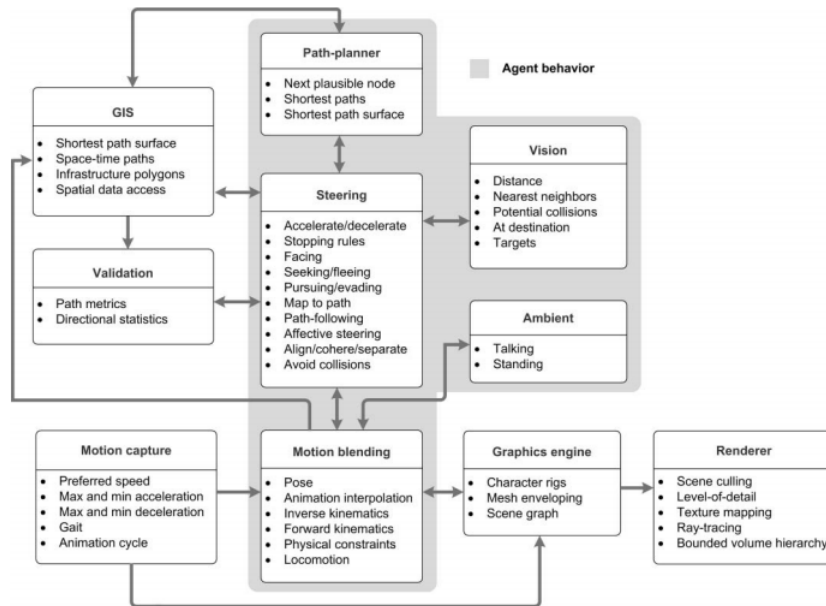


Figure 7: Flow of information for the model presented in [6]

haviours, each of one influence the others underlying, and that combined together generate the decision model for the movement of an agent:

**HIGH-LEVEL BEHAVIOUR** the high level behaviour is the one that concern the planning of the path from its origin to its destination. According to some studies, people define previously the path who want to follow and approximately the journey time[8][9], so this component is the one used to plan which is the best strategy to arrive at the destination;

**MEDIUM-LEVEL BEHAVIOUR** at this level, the model try to insert into the decision process the components that are amenable to those behaviours related to the objects and agents surrounding the current agent, and other aspects that may be taken into account at the scale of the streetscape (aspects related also to the steering of the agent, avoid collisions and so on);

**LOW-LEVEL BEHAVIOUR** at the lowest level, the model take into account aspects that are related to each single step. Following some physics rules derived from kinematics, the model aggregates the other information from high-level and medium-level to calculate the position of the agent in the next instant.

In figure 8 we show an example of high level behaviour: the agent choose initially its preferred path to reach its destination,

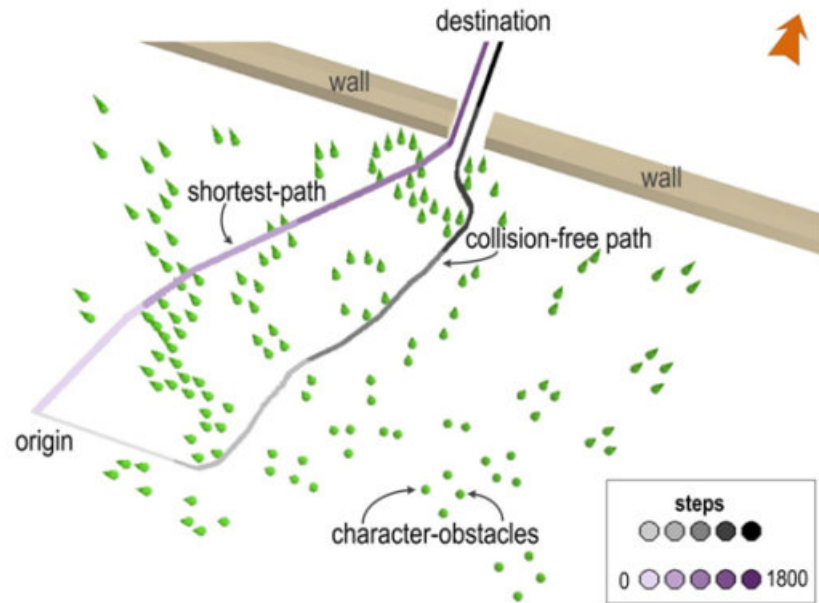


Figure 8: Example of path choice of an agent from its origin to the destination

evaluating the length of the possible paths and the probabilities of collision with other obstacles (other pedestrians and the walls). However the author reports how some potential significant details are not treated; the main lack is the one related to the focusing on individual behaviour, while the group behaviour is only considered from the point of view of the individual.

### 2.1.3 Interactions for complex system modelling

Another point of view of the study on interactions between agents is the one that tackle the whole dynamic of one system. An example is the work [12] presented by Quattrociocchi, Latorre, Lodi and Nanni. Here the authors would like to find some interesting characteristics from dataset of trajectories that could be useful to define structural characteristics of data, or to analyze them in a quantitative manner to describe particular subdivisions that can exist within data. The work's goal is to find some features, movement patterns and interaction patterns, to classify the starting trajectories searching for dangerous drivers. This is done starting from the raw data and analyzing the interaction between the neighbors of a specific agent along a part of the considered trajectory. For example functions that calculate interactions can be the difference of velocity between the considered agent and the average value in a portion of the studied area, or some other differences between a characteristic value of the agent and aggregates of the values of the

other agents in a specific region.

In this work we have a first attempt to define the notion of *interaction pattern*, even if there the authors try to describe interaction patterns in a different way respect to the definition we will give later: in that paper the studied area is divided in regions where, for each one, is calculated a descriptors set that identify some measurable quantities (as the average speed of the vehicles in that region defined as low, medium and high speed region). Thus those interaction patterns describe frequent schemes of variation that occurs for an agent moving from a region to another one. For example an interaction pattern could be the following:

$$\{l_{\text{dist}}, h_{\text{speed}}\} \Rightarrow \{m_{\text{dist}}\} \quad (2)$$

which means that is frequent to have agents moving from regions with low average distance between fast agents, to regions in which we can find a medium distance between agents.

Important elements to be considered when modelling complex systems are surely the ones related to the social interactions studied [13] in other fields, as the social sciences, to create cognitive models that can be used in the simulation to integrate a modelling more physical and mathematical with aspects related to the *internal* and conscious/unconscious choices of each agent.

#### 2.1.4 Reality mining

This work has been developed by the MIT Human Dynamics Laboratories[15] and is probably the one closer to our intentions and to our idea on mining pattern that could describe frequent agents' behaviours. The authors wanted to find some structures into the people's routines and activities, trying also to infer social relationships by analyzing the interactions between the cellular phones of the sample taken into account: using the Bluetooth devices and some ad-hoc cell tower to retrieve the data about absolute position and the relative to other Bluetooth devices, they developed two software, BlueAware and Bluedar that, running in background on the phone, have had the main task to collect data about position of the cell and other devices in the neighborhood. Once tackled the problem to collect data, the work focus its attention on two main activities:

CONTEXT INFERENCE using data about the usage of phones' applications, the authors try to understand which are the most used applications for different contexts; this can be

useful to understand, basing the decision on the place in which the user currently is, what applications will be most probably used, or if there are a significant modify into the habits of the user itself.

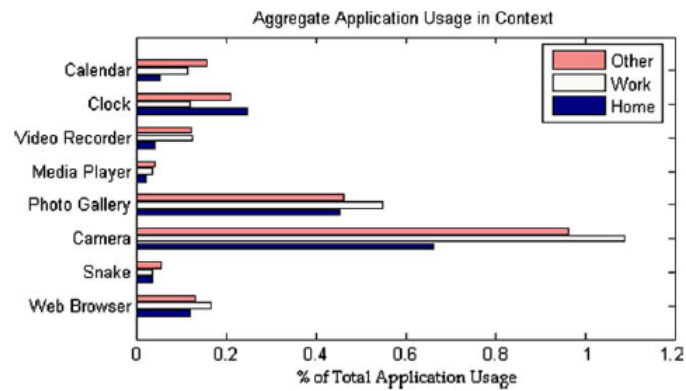


Figure 9: Phone applications' usage for the context analyzed by the article

**RELATIONSHIP INFERENCE** the context's inference is only the first part of the work, that have the ambition to do more. In fact, once inferred the user's context, one can try to infer who are the persons that the users could meet during the day. This is an indirect indication of a specific relationship that can be used to generate communities for the current user: those communities can be used to say, for example, when is most probable to meet someone, or to have general information about the specific community.



Figure 10: Friendship network (left) and the proximity network (right) elaborated from devices proximities.

All the information retrieved are finally used to describe underlying communities, building generative models to parametrize the dynamics of those networks and so on.

This is, without any doubts even if with significance differences, the work that is most closer to the one we have done. In fact even here in some way the authors try to retrieve informations and patterns to describe habits of the users by some sort of

interactions (in particular, interactions between devices and between each user and its device). However we start from a different context, we faced with small and well delimited areas, and we try to find patterns that could be described by events in functions of some low-level interactions. We can state that our work, furthermore, has the ambition to represent a more general and complete solution that could be also applied, with a preliminary brief adjustment, in this context.

## 2.2 DATA-MINING BACKGROUND

### 2.2.1 *Apriori principle and association rules mining*

The *apriori principle* is the property related to the one that is probably the most known application of data mining, the *basket market analysis*, associated to the problem of the mining of association rules. The first definition of the problem and of the algorithm was given by Agrawal, Imielinski and Swami in [19], where a faster and improved implementation is discussed in [20]. In this problem we have a set of  $m$  transactions and  $n$  different items present into the database, and we want to find a set of rules with which we can predict the occurrence of an item based on the occurrences of other items into the transactions [18]. In figure 11 we show an example of association rules elaborated from a database of transactions <sup>1</sup>.

Market-Basket transactions		Example of Association Rules
<b>TID</b>	<b>Items</b>	
1	Bread, Milk	{Diaper} → {Beer},
2	Bread, Diaper, Beer, Eggs	{Milk, Bread} → {Eggs, Coke},
3	Milk, Diaper, Beer, Coke	{Beer, Bread} → {Milk},
4	Bread, Milk, Diaper, Beer	
5	Bread, Milk, Diaper, Coke	

Implication means co-occurrence,  
not causality!

Figure 11: Example of association rules from a database

The algorithm for finding the association rules is structured in two main part:

**GENERATION OF FREQUENT ITEMSETS** in this phase the algorithm try to find all the subsets of items that are frequent in a database made of transactions, the *frequent itemsets*. In the figure 11, for example, one of these subsets is given by {diaper, beer};

<sup>1</sup> all the images about association rules mining has been taken from the support material of [18], at the website <http://www-users.cs.umn.edu/~kumar/dmbook/dmslides/>

GENERATION OF ASSOCIATION RULES once retrieved the set of frequent itemsets, the algorithm elaborate the association rules returning in output only those ones that has confidence over a certain threshold.

The part of the problem that mainly interest us, is the one relative to the generation of the frequent itemsets. This because in this phase is used the apriori property, that we used in the algorithms that we have developed to find the frequent instance patterns and the frequent sequence of interaction patterns. Is quite obvious that, given  $n$  different items, there are at most  $2^n$  possible itemsets that can be generated with a brute force approach. Starting from the null element we can see in figure 12, in which we have five items, the lattice of the possible itemsets of size  $k$  generated by pairs of itemsets of dimension  $k - 1$ , with  $1 \leq k \leq 5$ :

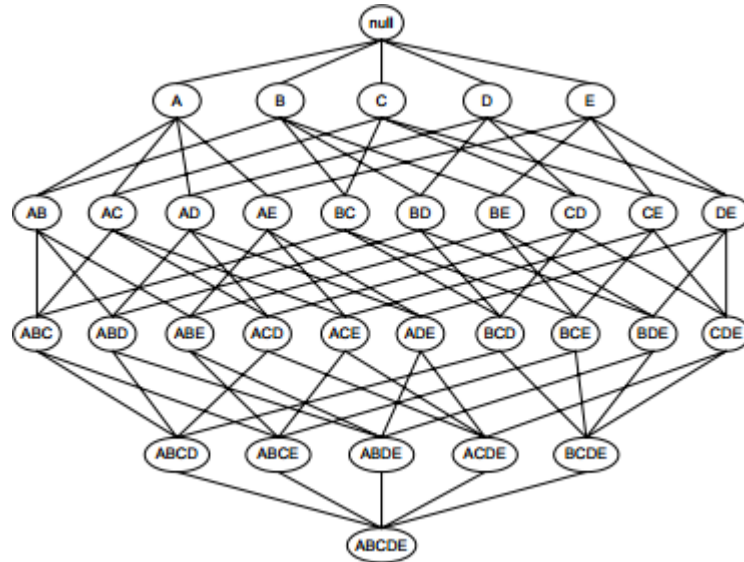


Figure 12: Example of lattice for generate all the subsets for 5 items

The *apriori property* is the one that is used to prune part of this lattice, excluding the ones that generate surely infrequent itemsets. If we indicate with  $\text{support}(\cdot)$  of an itemset  $X$  the fraction of transactions containing the elements of  $X$  over the entire database, we can state the property as:

**Definition 1** (Apriori principle). *Given two itemsets  $X$  and  $Y$ , holds the following:*

$$(X \subseteq Y) \implies \text{support}(X) \geq \text{support}(Y) \quad (3)$$

*that is also known as anti-monotone property of the support.*



The previous principle states that if we have an infrequent itemset, there won't be any superset of that itemset containing it that could be a frequent itemset. So, when the algorithm find that a subset is not frequent into the transaction set, it does not generate any superset that contains the infrequent subset. Taking the previous lattice, and assuming that the subset  $\{A, B\}$  is infrequent, we can see the effect of the pruning followed with the applying of the apriori principle.

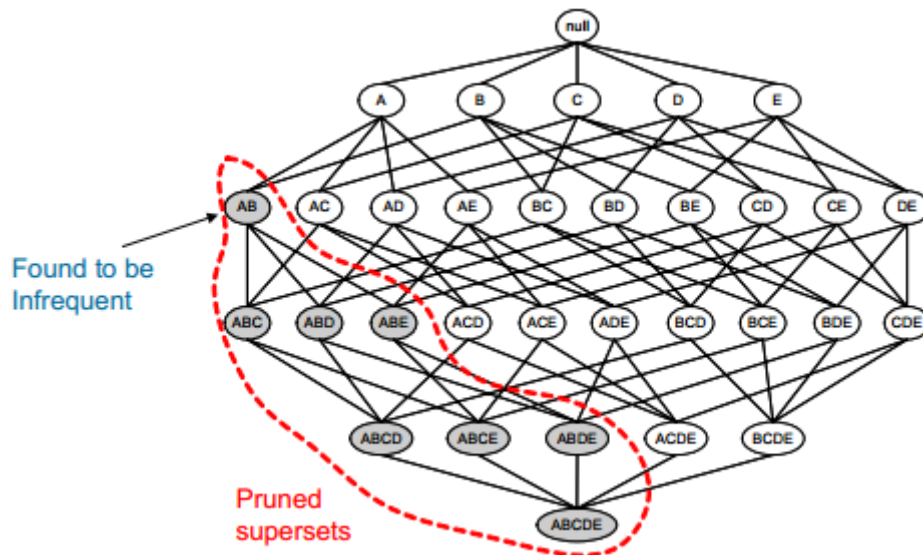


Figure 13: Example of pruned lattice for 5 items

This principle in general can apply a drastic reduction of elements generated by the algorithm during the search of the frequent itemsets. However, even if in real cases it is very improbable, in the worst case (there are no infrequent itemsets) the algorithm still generates  $\binom{n}{k}$  itemsets. In our work we will use this principle in the algorithm that we has developed, because to find all the interaction patterns we will interested to keep, and to grow, only those patterns which are frequent, i.e. that have a support value such that are greater than a decided threshold.

### 2.2.2 Graph mining

Graph mining is the field of data-mining which deals to extract frequent substructures present in datasets represented as graphs [21]. In particular this type of mining can be used in a lot of disciplines, due to the increasing of importance of network models in various scientific fields: apart the just mentioned social networking, we can also describe with graphs the chemical compounds, the web, the protein interactions and so on. Thus work directly with graphs is something that improve the under-

standing of the results, that are given in form of graphs themselves. In the case of frequent substructure mining in chemical compounds, we show an example <sup>2</sup> of graph mining in figure 14.

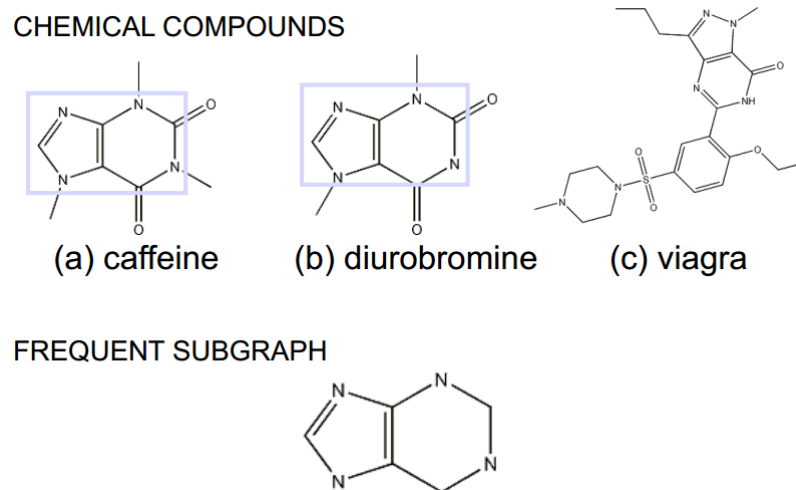


Figure 14: In the dataset of three chemical compounds, given as graphs, is found a frequent structure.

In our work some concepts from the graph mining will be useful, because there will be a moment in which, from a set of elaborated graphs dataset, we would find the ones that in the introduction we have called *interaction patterns*. In particular, to do this, we couldn't use some of the common tools for the graph mining, as the Apriori Graph Mining [22] or the gSpan[23] algorithms, because our dataset will have specific characteristics that those instruments do not take into account, making them unusable. However we will take inspiration from these tools to develop an algorithm that can elaborate our desired form of data.

### 2.3 GRAPH THEORY

Graph theory is the mathematical field that study the graph structures, that are mathematical models which are used to represent objects that are connected with links between them. In our work we will use intensively the graphs, thus we introduce some important concepts about graphs and graphs theory. First of all we give the mathematical definition of a graph.

<sup>2</sup> the image has been taken from the support material about graph mining of the *Interdepartmental Bioinformatics Group - Max Planck Institute for Biological Cybernetics*, at the website <http://agbs.kyb.tuebingen.mpg.de/wikis/bg/BNA-4.pdf>

**Definition 2 (Graph).** A graph  $G$  is a pair  $G = \langle V, E \rangle$ , where  $V$  is the set of vertices and  $E$  is the set of edges. In particular  $E \subseteq V \times V$  identify the set of links that could be directed or undirected.

The graphs can be represented as shown in figure 15. The last of these three examples represents a labelled graph, that we will use to define the framework for the analysis of the interactions between moving agents.

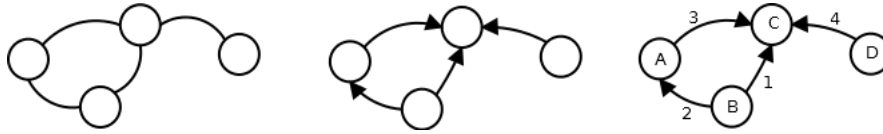


Figure 15: Three examples of graphs: undirected, directed and directed labelled.

We can define formally a labelled graph as:

**Definition 3 (Labelled graph).** Given a set of labels  $\mathcal{L}_V$  for the vertices, a set of labels  $\mathcal{L}_E$  and two functions  $\phi_V : V \rightarrow \mathcal{L}_V$  and  $\phi_E : E \rightarrow \mathcal{L}_E$ , a labelled graph  $G$  is a graph where for each  $v \in V$  :  $label_v = \phi_V(v)$  and for each  $e \in E$  :  $label_e = \phi_E(e)$ .

Each graph, labelled and unlabelled, can be represented with a matrix called *adjacency matrix*: if a graph is such that  $|V| = n$ , then the adjacency matrix is a structure with dimension  $n \times n$  with nodes labels as rows and column. Examples of adjacency matrices for labelled graphs, can be seen in figure 16.

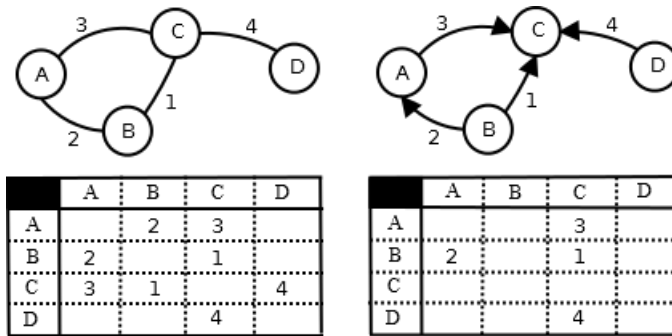


Figure 16: Adjacency matrices associated to an undirected graph and to a directed one.

The concept of adjacency matrix is important because it's the structure usually used to work on matrices and is a compact representation of a graph with its relations; in the case of undirected graph it is necessary only half of the original matrix thanks to the symmetry of the matrices in the undirected case.

### 2.3.1 Multigraphs

In the following chapters we will see that, once elaborated the information connected to the interactions between agents, we will be interested to represent the results in a more simple form to be elaborated by the algorithm that we will develop. In particular we will use the concept of *multigraph*. Multigraphs are a generalization of the simple graphs as we have introduced in the previous section. The difference between the two kind of structures is given by the fact that in simple graphs can exists only one edge between two vertices, while in a multigraph we have not this requirement. In figure 17<sup>3</sup> we show an example of multigraph.

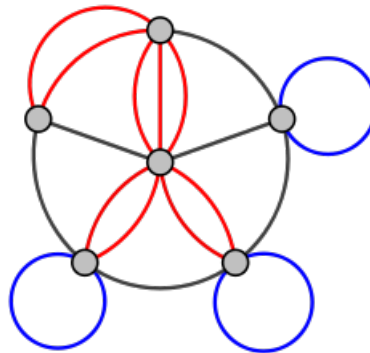


Figure 17: Example of undirected multigraph

With those structures we can easily model multiple relations between nodes of the graph, and this has become very important in the last years where, thanks to the increasing and the diffusion of the social networks, we have access to a lot of data about the connections and the relations between the users that can be analyzed. This because relations can be represented and modelled with labelled multigraphs, as we can see in figure 4<sup>4</sup> 18.

We have introduced multigraphs because we will use them when it will be necessary model in a simpler and blunt way the relations retrieved with the analysis framework (those relations will correspond to ongoing events) between moving agents before starting the search of our interaction patterns.

<sup>3</sup> The image of is taken from the website <http://en.wikipedia.org/wiki/Multigraph>

<sup>4</sup> The image of is taken from [1]

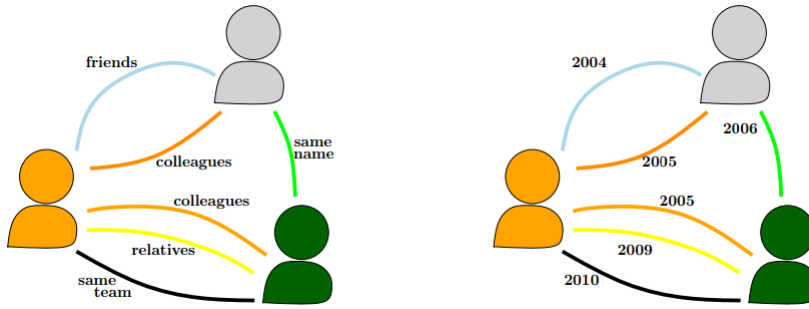


Figure 18: Relations' modelling between a users of a network with a multigraph

2.3.2 Graphs isomorphism problem

The last section about graph theory is the one relative to the *graphs isomorphism problem*. We can state this problem as:

**Definition 4** (Graph isomorphism problem). *Given two graphs  $G = (V_G, E_G)$  and  $H = (V_H, E_H)$ , the graph isomorphism problem is the problem to find a bijection between the sets of vertices  $V_G$  and  $V_H$  which preserves the edges.*

Thus this problem is the one to state if two graphs are the same, unless labels. In figure <sup>5</sup> 19 two isomorphic graphs.

Graph G	Graph H	An isomorphism between G and H
		$f(a) = 1$ $f(b) = 6$ $f(c) = 8$ $f(d) = 3$ $f(g) = 5$ $f(h) = 2$ $f(i) = 4$ $f(j) = 7$

Figure 19: Two isomorphic graphs with the function of translation

This is a well-known problem, that is faced in particular in scientific field as, for example, mathematical chemistry and cheminformatics [14] where is important to identify same chemical compounds or same substructures present in different compounds. Recognize that two graphs are the same, if we do not consider the labelling of the vertices, is something that will be very useful in our work when we will have to decide if two patterns with real agents represents the same generic pattern.

<sup>5</sup> The image of is taken at the website [http://en.wikipedia.org/wiki/Graph\\_isomorphism](http://en.wikipedia.org/wiki/Graph_isomorphism)

However, even if this problem is so important, it is known that it is an NP problem [32], where the best result known in literature for a graph with  $n$  nodes is given by an algorithm of complexity  $2^{O(\sqrt{n \cdot \log n})}$  [33]. The most simple algorithm to recognize if two graphs are isomorphic is the one that we show in figure 20.

```

1: procedure GRAPH ISOMORPHISM(G, H)
2:    $am_G = \text{adjacency\_matrix}(G)$ 
3:    $am_H = \text{adjacency\_matrix}(H)$ 
4:   for all permutation of node in  $am_H$  do
5:     if  $am_G \equiv am_H$  then
6:       return TRUE
7:     end if
8:   end for
9:   return FALSE
10: end procedure

```

Figure 20: Naïve graph isomorphism psuedocode

This algorithm is based on the fact that two isomorphic graphs must have the same adjacency matrix: so it simply permutes the vertices of one of the two graphs until the adjacency matrices become the same. Since the cost of the comparison between two adjacency matrices of graphs with  $n$  vertices is  $O(n^2)$  and since the algorithm generates in the worst case all the permutations of the vertices, with a cost of  $O(n!)$ , we can state that the naive algorithm has an overall cost of  $O(n! \cdot n^2)$ . However we will see how to face this problem in our case, developing an heuristic that in real cases has better performance respect to the worst one.

### 2.3.3 Temporal graphs

Temporal graph [36], or temporal networks [35], are structures that describe a model evolving in time making explicit the temporal component of the model. Take as example the following figure where is reported a table representing the emails send/received from a set of persons. The images are taken from [36].

Obviously, if we want to model the situation of this exchange of emails we could use a directed graph, as reported in figure 22.

It is clear that this representation does not consider an important information given by the data, that has been flattened on a non-temporal representation. Temporal graph has been developed to tackle those situation, in which the temporal informa-

Sender	Recipient	Time
A	B	$t_1=0$
A	C,E	$t_2=1$
E	D	$t_3=3$
B	C	$t_4=5$
B	D	$t_5=9$
D	B	$t_6=14$
A	D	$t_7=20$

Figure 21: Table with the information about the exchange of emails between A, B, C, D and E.

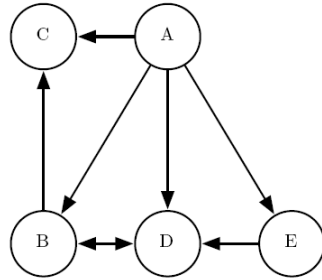


Figure 22: Representation of mail exchanging relation between A, B, C, D and E

tion and the graph representation must be mixed to have the maximal expressiveness to the ones that have to analyse those kind of data. Thus, the data contained in the table of figure 21 can be represented in the following manner.

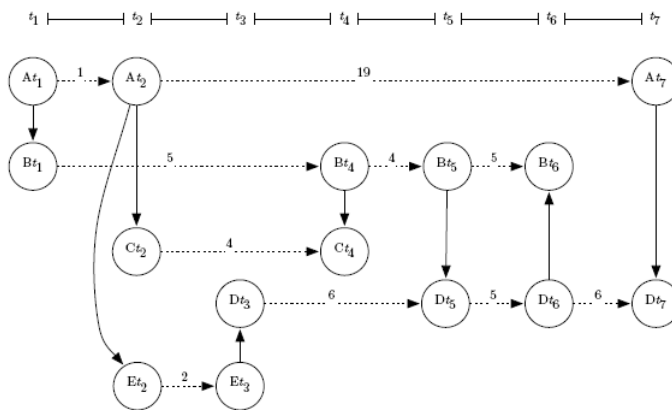


Figure 23: Representation of mail exchanging relation between A, B, C, D and E with a temporal graph. The dashed lines means that the person is waiting.

A lot of things can be modelled with those kind of graphs: in particular we will see how we will use a very similar structure, that we won't use at maximum of its potentiality, to represent relations between moving agents and that will be to the basis of all our work.





## Part II

# MINING DATA WITH INTERACTION PATTERNS

We introduce now a new methodology to analyze mobile data from a point of view based on the concept of *interaction pattern*. First of all we will define the problem, introducing the elements that we need to bring out from the data useful information about the interactions between moving agents. After we develop two algorithms that we will use to extract behavioural patterns from the elaborated data regarding the movement of the agents. Finally we will see the application to the interaction pattern mining on two real cases, showing examples of functioning of ours tools.



## THE INTERACTION PATTERN ANALYSIS FRAMEWORK

In this chapter we will define the framework that is used to abstract and to bring out particular configurations between moving agents. In figure 24 are shown the components we face in this chapter and that we have defined: starting from the trajectories' dataset, called as *raw data* in the example, we want to define a better representation for trajectories that we can use for our computation. This representation is based on a sequence of graphs, each of which represents the neighborhood relations between agents and their interactions, in the form of edges' labels. On this new structure we will search for events between pairs of agents and events on single agents, that are the first abstract objects in output on which we will base the search of the interaction patterns.

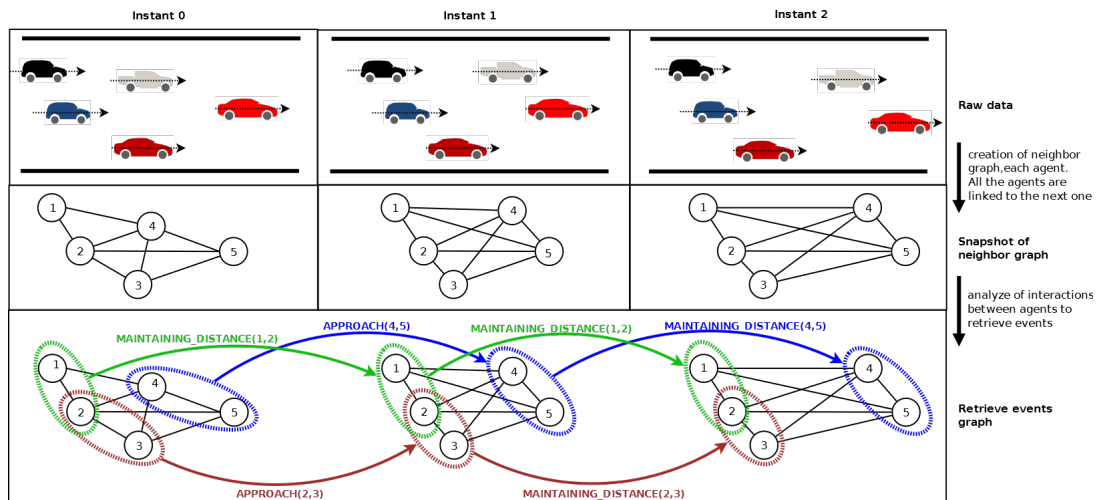


Figure 24: Schematic description of what we tackle in this chapter.

All is constructed around interactions, as we have introduced in chapter 1, of which we gave an informal description but that now we want to define clearly:

**Definition 5 (Interaction).** *We define as interaction any measurable quantity, returned from some defined function, that can potentially influence the behaviour of the agents involved in the measurement or that can describe something that is happening between the agents in their sphere of influence.*

Along this chapter we will define, as first, the theoretical notions on which the framework is based. During the explaining

of that theoretical part, we present also an instantiation of the framework that we have used for our experiments, that will be reported in chapter 6.

### 3.1 PRELIMINARIES

In the previous chapters we have stated that our work is based on trajectories associated to moving agents. So before starting defining the framework and the algorithms we have developed, we formally define the notions of trajectory, trajectories' set and of agent.

**Definition 6** (Trajectories and trajectories' set). *The set of trajectories  $\mathbb{T}$  is the set of the trajectories observed from a set of moving agents in a time scope  $\mathbb{T} = \{t \in \mathbb{N} | 0 \leq t \leq T_{\max}\}$ :*

$$\mathbb{T} = \{\mathbb{T}_1, \dots, \mathbb{T}_n\} \quad (4)$$

where  $n$  is the number of unique agents of the dataset. Each trajectory  $\mathbb{T}_{id}$ , with  $1 \leq id \leq n$ , is the set of points in which the agent with identifier  $id$  has been tracked into an interval  $[s, e] \subseteq \mathbb{T}$ :

$$\mathbb{T}_{id} = \{(x_s, y_s), \dots, (x_e, y_e)\} \quad (5)$$

The trajectory set identify then a set of agents, all unique, each of one has associated a trajectory of the initial dataset.

**Definition 7** (Agent set and agent). *Given a set of trajectories  $\mathbb{T}$  where  $|\mathbb{T}| = n$ , we define the agent set as*

$$\mathcal{A} = \{\mathcal{A}_1, \dots, \mathcal{A}_n\} \quad (6)$$

where a single agent is defined as

$$\mathcal{A}_{id} = \langle \mathbb{T}_{id}, \{(\mathbb{T}_{id}^k, d_{id}^k, v_{id}^k)\}_{k \in \mathbb{T}_{id}} \rangle \quad (7)$$

where  $1 \leq id \leq n$  and  $\mathbb{T}_{id} = [s, e] \subseteq \mathbb{T}$ . The values  $\mathbb{T}_{id}^k = (x_k, y_k)$ ,  $d_{id}^k$  and  $v_{id}^k$  represent respectively the position of the agent  $id$ , its direction and its velocity in the instant  $k$ . For the sake of simplicity we can also refer to the set that represents an agent as

$$\mathcal{A}_{id} = \{\mathcal{A}_{id}^s, \dots, \mathcal{A}_{id}^e\} \quad (8)$$

As above,  $\mathbb{T}_{id} = [s, e]$ .

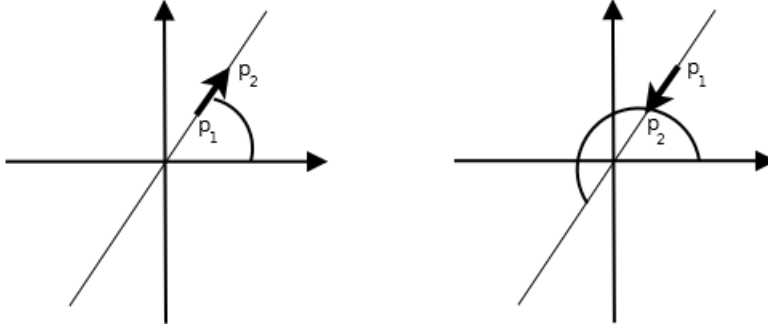


Figure 25: Example of direction's derivation for a moving agent; considering two points in which the agent is positioned in two consecutive instants. The difference between the y values in these instants ( $x$  in the case of direction 0 or 180) determines which angle consider.

Thus, assuming to know the sampling rate of the trajectories, we can start from these points to derive the velocity and the direction of an agent in each instant. We assume as primitive the agent described into definition 7, also containing the derived measures from the associated trajectory. We show in the following how they can be computed.

Given two points  $p_1 = (x_1, y_1)$  and  $p_2 = (x_2, y_2)$  in consecutive instants, the velocity for an agent can be calculated as:

$$v(p_1, p_2) = \frac{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}}{t_u} \quad (9)$$

where  $t_u$  is the temporal sampling unit for the points, that is shared by all trajectories. Given the same points, the direction is calculated as:

$$d(p_1, p_2) = \begin{cases} -1 & (x_1 - x_2) = 0 \wedge (y_1 - y_2) = 0 \\ \text{angle}(p_1, p_2) & \text{otherwise} \end{cases} \quad (10)$$

where  $\text{angle}()$  is the absolute angle of the line on which lies the movement vector respect to the unit circumference. In figure 25 we see how this function must take into account the values of both points to decide if take an angle or its opposite.

### 3.2 THE IPA FRAMEWORK

We start now to develop and discuss a general framework, the *Interaction Pattern Analysis* (IPA) framework, for the analysis and the retrieving of events in which are involved single agents and all the pairs of agents that interact with each others.

### 3.2.1 Representing trajectories with a graph

First of all we want transform the original trajectories dataset  $\mathbb{T}$  in a structure that can describe the relations between agents in a manner for which will be simpler for the framework we will define the seeking process of our atomic components. Since we are putting the basis of all the work, we have chosen the way to represent the trajectories of the dataset agents very carefully. In the introductory examples, into the chapter 1 but also into the works presented in the section 2.1, we have seen how is natural to describe relations and interactions between pairs of objects, instead of representing subsets of agents variable in time. Furthermore we assume that an agent directly or indirectly influence the others into the environment. These considerations have led us to describe trajectories associated to the moving agents with a series of consecutive graphs, where the main relation (the edges) represent a neighborhood relation: each of these graphs is connected with the ones representing the previous instant and the following one, creating so a unique graph structure. We can so define the notion of neighbor graph.

**Definition 8** (Neighbor Graph). *Given a set  $\mathcal{A}$  of agents over the time scope  $\mathbb{T}$  and given a neighborhood function  $\mathcal{N}$  defined between pairs of agents in a time instant, the neighbor graph is a graph  $G_{\mathcal{A},\mathbb{T}}^{\mathcal{N}} = \langle \mathbb{N}, \mathbb{E} \rangle$ , with:*

1. nodes  $\mathbb{N} = \{\mathcal{A}_i^t \mid \mathcal{A}_i \in \mathcal{A} \wedge t \in \mathbb{T}_i\}$

2. edges  $\mathbb{E} = \mathbb{E}_{\text{ego}} \cup \mathbb{E}_{\text{int}}$ , where

$$\mathbb{E}_{\text{ego}} = \{(\mathcal{A}_i^t, \mathcal{A}_i^{t+1}) \in \mathbb{N}^2\}$$

and

$$\mathbb{E}_{\text{int}} = \{(\mathcal{A}_i^t, \mathcal{A}_j^t) \in \mathbb{N}^2 \mid \mathcal{A}_j^t \in \mathcal{N}(\mathcal{A}_i^t) \wedge \mathcal{A}_i^{t-1}, \mathcal{A}_j^{t-1} \in \mathbb{N}\}$$

In figure 26 we show, in the right part of the figure, the representation of the neighbor graph for three snapshots (in the left part) of three consecutive instants, for the agents with identifiers  $\{1,2,3\}$  *Ego* edges represent vertical links that connect the different occurrences of each agent at consecutive time instants, whereas *interaction* edges represent pairs of agents that at specific moments fall within each other's sphere of influence, thus could potentially interact. Notice that  $\mathbb{E}_{\text{int}}$  excludes initial time instants of agents: this is due to the fact that in our framework we will consider only events that involve changes of features between time instants, which are not defined for initial points.

The choice to represent the agents with a graph is the better one, even because this is a representation where the relations

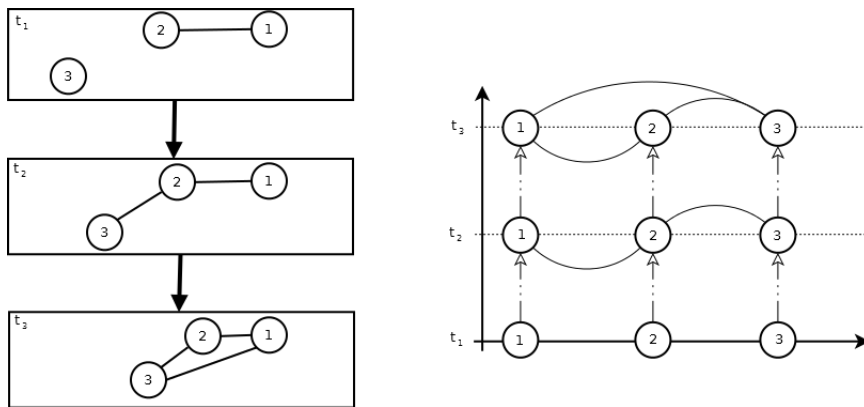


Figure 26: Representation of a neighbor graph

between agents are immediately clear and where the representation of the neighbor relation, the one privileged for us at this level, is the most natural. Furthermore in literature we have other cases in which relations and interactions between moving agents are represented using graphs [11] [30] [31]. This representation, finally, is very close to a temporal graph, presented in the chapter 2 and then can be studied also in this way. However we have mostly used it to base a new kind of work instead of using the techniques studied to analyze the agents and their interactions with this form of representation.

### 3.2.2 Neighborhood creation

In the definition of neighbor graph a fundamental component is the neighborhood function. The type of neighborhood is varying respect to the specific context and also respect to the assumptions that we could make about agents: a particular relevance is given by who can influence the others into the environment. Is intuitive in a mobility context to think that, chosen an agent, agents closest to the one considered have a greater influence on it than the farthest ones. We have chosen to define this relation, in our case, as the one described by all the closer agents to the one considered that are visible by him, and vice versa. We make an example to explain what we mean in figure 27

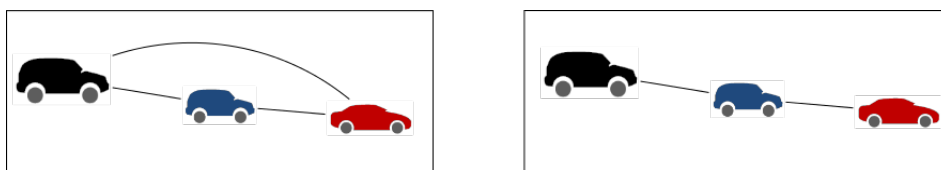


Figure 27: Different possible choices for the neighborhood relation

Both possibilities represent valid neighborhood relations, but the one that is the closest to our approach and to our definition

of interaction is without doubt the one represented in the right part of the figure. This because, if we want that interactions will be the measure indicating potential influences between agents, following the definition 5, in the first case we are assuming that two agents (black and red cars) could have potential interactions even if they are not directly visible each other. We instead prefer the second kind of relation in which interactions, and consequently events, between the three cars can be explained with the pairs of relations  $\langle \text{black car, blue car} \rangle$  and  $\langle \text{blue car, red car} \rangle$ , where they are all directly visible each other.

In other contexts the neighborhood relation can be described in other ways: for example, if we would adapt the framework to manage interactions between non moving agents as financial markets, we could define a function that take into account other types of parameters.

In our case we want to construct a function of neighborhood that works in a similar way to the algorithms that create visibility graphs [28][29]. In figure 28 an example of visibility graph for the points  $s$  and  $t$  and the obstacle's vertices<sup>1</sup>.

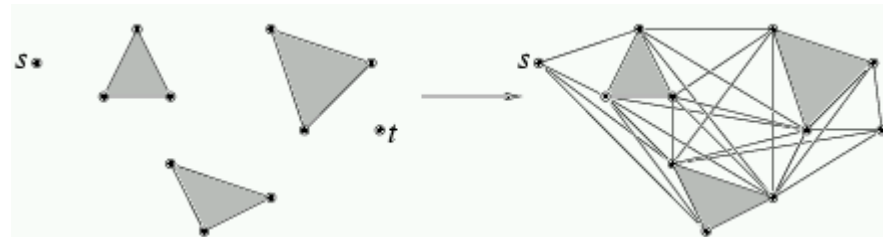


Figure 28: An example of visibility graph for the points  $s$  and  $t$

In those kind of graphs, widely used in robotics, the points are connected to the vertices of the obstacles with an edge iff they are visible each other, and these edges are called *visibility edges*. Our intention is very similar, in fact we have defined our neighborhood function  $\mathcal{N}$  in this way: given a search diameter  $d_{\text{search}}$  and an agent described by a point and a circle of radius  $r_{\text{agent}}$  (fixed for all agents) that describe a radius of visibility having the agent as center, returns all the agents visible to him, i.e. all the agents into the search diameter  $d_{\text{search}}$  for which we can draw a line that has no intersection with any other circles between the two points. In figure 29 we can see an example of behaviour for our implementation of the neighborhood function.

<sup>1</sup> Image taken from the website <http://www.cs.wustl.edu/~pless/546/lectures/l22.html>



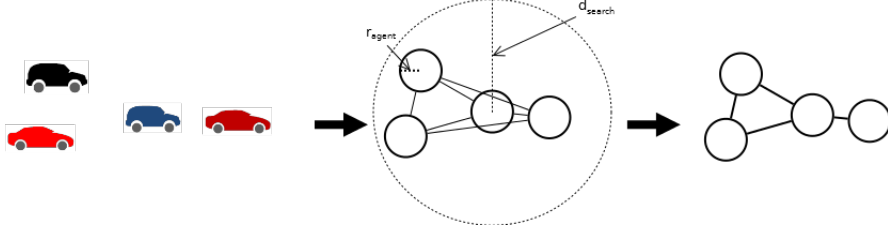


Figure 29: Construction of our neighbor graph for the initial situation in the left part of the figure

### 3.2.3 Measuring interactions

Once defined the structure with which we will represent the trajectories associated to the agents, we are interested to define how to measure interactions. Thus we introduce two kind of function that we can use to label the neighbor graph:

**Definition 9** (Ego and interaction functions). *Given an agent set  $\mathcal{A}$ , a ego function  $f_{\text{ego}}^A$  is defined as a function that associates a predefined number  $n_{\text{ego}}$  of real values to two consecutive instances of an agent, i.e.  $f_{\text{ego}}^A(\mathcal{A}_i^{t-1}, \mathcal{A}_i^t)$  returns values in  $\mathbb{R}^{n_{\text{ego}}}$ .  $f_{\text{ego}}^A$  remains undefined for the first instance of an agent, i.e.  $t-1 \notin T_i \Rightarrow f_{\text{ego}}^A(\mathcal{A}_i^{t-1}, \mathcal{A}_i^t) = \text{undef}$ .*

*Similarly, an interaction function  $f_{\text{int}}^A$  is defined as a function that associates a predefined number  $n_{\text{int}}$  of real values to each pair of contemporary instances of two agents, also involving their previous instances, i.e.  $f_{\text{int}}^A(\mathcal{A}_i^{t-1}, \mathcal{A}_i^t, \mathcal{A}_j^{t-1}, \mathcal{A}_j^t)$  returns values in  $\mathbb{R}^{n_{\text{int}}}$ . As for  $f_{\text{ego}}^A$ ,  $f_{\text{int}}^A$  remains undefined when the first instance of an agent is involved.*

Both ego and interaction functions, so, can be seen as t-uples of respectively,  $n_{\text{ego}}$  and  $n_{\text{int}}$  function:

$$f_{\text{ego}}^A = (f_{\text{ego}_1}^A, \dots, f_{\text{ego}_{n_{\text{ego}}}}^A) \quad (11)$$

$$f_{\text{int}}^A = (f_{\text{int}_1}^A, \dots, f_{\text{ego}_{n_{\text{int}}}}^A) \quad (12)$$

Ego functions are those ones that calculate the variations of internal parameters of an agent in two consecutive instants. Those parameters could be the ones associated to the change in the acceleration, in the direction and so on; however, potentially every measurable change into the state of the agent can be associated to a simpler or more complex ego function. Interaction functions are those ones that calculate a measure regarding the situation between the internal parameters of the involved

agents. These functions consider pairs of agents in two consecutive instants because could be necessary to see the previous state of both agents to return a correct evaluation of the interaction. For instance, take a function that checks the variation of distance: considering only the coordinates of two agents in one instant is not sufficient to determine if the distance between the agents is decreasing or not, because we must have a point of reference to return the correct evaluation. If we consider also the previous distance between them, we can return a value indicating if one of the agent is approaching the other or not.

If we see these values at this point they simply give us an instantaneous of the differences or measures between parameters of different agents. However, if we consider those values in different instants related each other, those simple measurements can tell us something about a behaviour that we can construct to an higher level of abstraction. For example, even if we will see it more deeper after, take again the distancing function. If we see that the returned value for two agents is a value indicating a distance's decrease in an instant, we can state something about the instantaneous situation of the distancing between two agents: on the contrary, if we see that in a consecutive set of instants the function calculates decreasing values for this parameter, we can state that in the considered set of instants those two agents have been interacting. An event is ongoing between them, that we could define as *approach* of the first agent on the second one.

All of those events will be searched into the labelled version of the neighbor graph.

**Definition 10** (Labelled Graph). *Given a set  $\mathcal{A}$  of agents over the time scope  $\mathbb{T}$ , a neighborhood function  $\mathcal{N}$ , a ego function  $f_{\text{ego}}^A$  and a interaction function  $f_{\text{int}}^A$ , the Labelled graph is a graph  $G^L = \langle \mathbb{N}, E, \mathcal{L}_E, \mathcal{L}_J \rangle$ , where:*

1.  $\langle \mathbb{N}, E \rangle$  is the neighbor graph associated to  $\mathcal{A}$  and  $\mathcal{N}$
2. for each edge  $(\mathcal{A}_i^{t-1}, \mathcal{A}_i^t) \in E$ , a label  $\mathcal{L}_E(\mathcal{A}_i^t) = f_{\text{ego}}^A(\mathcal{A}_i^{t-1}, \mathcal{A}_i^t)$  is associated to node  $\mathcal{A}_i^t$
3. for each edge  $(\mathcal{A}_i^t, \mathcal{A}_j^t) \in E$ , a label  $\mathcal{L}_J(\mathcal{A}_i^t, \mathcal{A}_j^t) = f_{\text{int}}^A(\mathcal{A}_i^{t-1}, \mathcal{A}_i^t, \mathcal{A}_j^{t-1}, \mathcal{A}_j^t)$  is associated to edge  $(\mathcal{A}_i^t, \mathcal{A}_j^t)$ .

### 3.2.3.1 Instantiation of interaction functions

We briefly describe now the interaction functions we defined and that we will use in the rest of the thesis. In particular, even

if for completeness we have defined them, we do not use the ego functions and we have not instantiated them, because we focused our attention on the aspects related on the interaction functions, more closest to the basic concept of interaction.

Into the definitions of the functions, in table 1, we denote with  $\mathcal{A}_i$  and  $\mathcal{A}_j$  the pair  $(\mathcal{A}_i^{t-1}, \mathcal{A}_i^t)$  and  $(\mathcal{A}_j^{t-1}, \mathcal{A}_j^t)$  respectively.

Table 1: Complete definition for the interaction functions  $f_{name}(\mathcal{A}_i, \mathcal{A}_j)$  used

name	function definition
distance	$\ \mathbb{T}_i^t - \mathbb{T}_j^t\ _2 - \ \mathbb{T}_i^{t-1} - \mathbb{T}_j^{t-1}\ _2$
velocity	$v_i^t - v_j^t$
direction	$\min\{360 -  d_i^t - d_j^t ,  d_i^t - d_j^t \}$
position	$position(\mathcal{A}_i, \mathcal{A}_j, \epsilon_{lat}, \epsilon_{\parallel}, \epsilon_{move})$
align	$\begin{cases} 1 & \text{if } d(l_i, \mathbb{T}_j^{t-1}) \leq \epsilon_{align} \wedge d(l_i, \mathbb{T}_j^t) \leq \epsilon_{align} \\ 0 & \text{otherwise} \end{cases}$

In the distance function a negative value indicates an instantaneous approach, while a positive non-zero value indicates an instantaneous distancing. In the velocity function we can derive by the returned value if the first agent is faster or slower than the second one. The third function defined calculates the difference of direction between two agents: to note that this value can assume a minimum of 0 degrees (same direction) or 180 (opposite direction). The position function computes the position of the first agent w.r.t. the second one and it returns a code that indicates if the agent  $\mathcal{A}_i$  is behind ( $v_1$ ), ahead of ( $v_2$ ), lateral to ( $v_3$ ), flanked to with same direction ( $v_4$ ), moving in front of ( $v_5$ ), moving behind the other in opposite direction ( $v_6$ ) or not moving as like agent  $\mathcal{A}_j$  ( $v_7$ ): the function analyzes the line generated by the two points pair  $(\mathbb{T}_i^{t-1}, \mathbb{T}_i^t)$  and  $(\mathbb{T}_j^{t-1}, \mathbb{T}_j^t)$ , combining the information about the degree of parallelism of the two lines, the lateral positioning of the agent and the moving indication to return the correct positioning. In figure 30 we show three possible examples in which the position function returns three different values.

In the first case the red car is in front of the blue one, because the blue car is outside the band generated by the perpendicular line to the direction of the red car at distance  $\frac{\epsilon_{lat}}{2}$  from its center. These margins represent the band in which one agent is lateral to the one considered. In the second and third example the blue car is lateral, or flanked, to the red one; in the third case, analyzing the angle between the two directions, we can

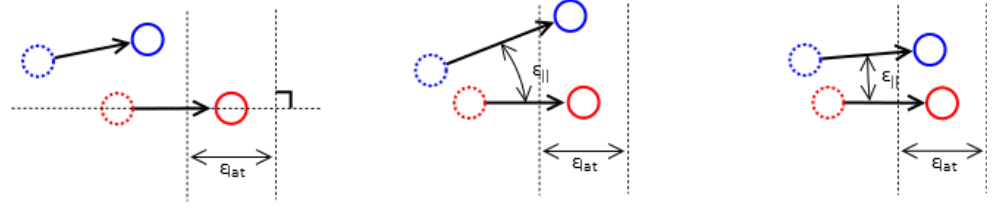


Figure 30: Three examples of position returned from the position function: in the first case a front-back relation, in the second a lateral positioning and in the last a flanking.

guess that the agents are flanked. The last function we have defined is the one that check the alignment between two agents: indicating with  $l_i$  the line  $\frac{y-y_1^i}{y_2^i-y_1^i} = \frac{x-x_1^i}{x_2^i-x_1^i}$ , the function checks if the second agent, in two consecutive instants, is contained into the band generated by the two lines parallel to  $l_i$  at distance  $\epsilon_{align}$  from it.



Figure 31: Two types of alignment

In the left part of figure 31 there isn't any alignment, because the red point in two consecutive instants isn't contained both in the band generated by the parallel lines to the direction of the blue car, each one at distance  $\epsilon_{align}$ . In the second case there is an alignment, the red point in two consecutive instants is contained into the alignment band.

### 3.2.4 IPA Events

Having defined the complete structure that abstracts the trajectories' dataset and where we can find the interactions between pairs of agents, we can finally define the first complex component that will be at the base of our interaction patterns in the following. As we introduced in section 3.2.3, the measuring of an instantaneous interaction is something not interesting itself, and that do not give us any information. We must so find and construct aggregations of interactions that can identify something about the dynamics of the environment. As we have already said in a previous example, a series of decreasing distances can be seen as an event, that we can call approach; this represents an event with a start instant, an end instant, a subject and an object of the event. These events emerge directly from the interactions, because (as introduced in the example) each event can be described by a template that indicates which characteristics, in terms of interactions calculated by the  $f_{int}^A$ , it

should have in a certain interval. In conclusion, the IPA framework primarily search events.

We can define so the notion of event template.

**Definition 11** (Event template). *Given an agent set  $\mathcal{A}$  and a time scope  $\mathbb{T}$ , an **event template** is defined as a predicate  $P : \mathcal{A} \times \mathcal{A} \times \mathbb{T}^2 \rightarrow \mathcal{B}$ .*

*The intended meaning of the predicate  $P(\mathcal{A}_1, \mathcal{A}_2, a, b)$  is that a property  $P$  holds in time interval  $[a, b]$ , involving agents  $\mathcal{A}_1$  and  $\mathcal{A}_2$ .  $\mathcal{A}_1$  is the subject of the event while  $\mathcal{A}_2$  is the object of the event.*

**Example 1.** *Now we can take the example already done and defining it more formally: an approach is an event template  $approach(\mathcal{A}_1, \mathcal{A}_2, t_s, t_e)$  which holds iff for each time instant  $t \in [t_s, t_e)$  is satisfied  $f_{distance}(\mathcal{A}_1^t, \mathcal{A}_2^t) < \epsilon_{dist}$  or  $f_{distance}(\mathcal{A}_1^t, \mathcal{A}_2^t) = \epsilon_{dist} \Rightarrow f_{distance}(\mathcal{A}_1^{t+1}, \mathcal{A}_2^{t+1}) < \epsilon_{dist}$ .*

As the example shows, templates can have a complex form, for instance not limited to a simple monotonic behaviour. In our case shown above, non-strict decreases in the distance are allowed only if isolated.

**Definition 12** (Graph events). *Given a labelled graph  $G^L = \langle \mathbb{N}, E, \mathcal{L}_E, \mathcal{L}_J \rangle$  and a family  $\mathcal{E}\mathcal{T}$  of event templates, the graph events of  $G^L$  are defined as the set  $\mathcal{E}\mathcal{J}$  of all event instances  $P(\mathcal{A}_1, \mathcal{A}_2, a, b)$  such that:*

1.  $\forall t \in [a, b]. \mathcal{A}_1^t, \mathcal{A}_2^t \in \mathbb{N}$
2.  $P(\mathcal{A}_1, \mathcal{A}_2, a, b) = true$
3.  $[a, b]$  is maximal, i.e.:  $[c, d] \supset [a, b] \Rightarrow \neg P(\mathcal{A}_1, \mathcal{A}_2, c, d)$

The events belonging to the  $\mathcal{E}\mathcal{J}$  set, found with a search in the labelled graph, are the objects resulting as output from the framework.

#### 3.2.4.1 Concrete events

The last thing we have to do is to finalize the instantiation of the framework. With the defined interaction functions, in the section 3.2.3.1, we want to create our library of concrete events that we want to search in the datasets that we will have to analyse. First of all we define the concrete event template:

**Definition 13** (Concrete event templates). *We have an event of type  $P(\mathcal{A}_i, \mathcal{A}_j, t_s, t_e)$  iff for each  $t \in [t_s, t_e)$  is satisfied a specific condition or in a  $t < t_e - 1$  the previous condition is not verified, but is verified in the instant  $t + 1$ .*

Table 2: Complete definition for concrete events

<b>p</b>	<b>condition</b>	<b>subject/object</b>
moving_away ←	$f_{\text{distance}}(\mathcal{A}_i, \mathcal{A}_j) > e_{\text{dist}}$	$\mathcal{A}_i$ is the subject if it's faster than $\mathcal{A}_j$
following .....→	$f_{\text{route}}(\mathcal{A}_i, \mathcal{A}_j) \leq e_{\text{route}} \wedge f_{\text{aligned}}(\mathcal{A}_i, \mathcal{A}_j) = 1 \wedge$ $f_{\text{position}}(\mathcal{A}_i, \mathcal{A}_j) = v \wedge -e_{\text{dist}} \leq f_{\text{distance}}(\mathcal{A}_i, \mathcal{A}_j) \leq e_{\text{dist}}$	$v$ indicate that $\mathcal{A}_i$ is behind the agent $\mathcal{A}_j$ , $\mathcal{A}_i$ it's the subject if it follows the $\mathcal{A}_j$
maintaining_distance →	$-e_{\text{dist}} \leq f_{\text{distance}}(\mathcal{A}_i, \mathcal{A}_j) \leq e_{\text{dist}}$ $\wedge (\exists [t_{s'}, t_{e'}] \subseteq [t_{s'}, t_e] \cdot t_{e'} - t_{s'} > 1 \wedge \text{following}(\mathcal{A}_i, \mathcal{A}_j, t_{s'}, t_{e'}))$	$\mathcal{A}_i$ is the subject if it follows $\mathcal{A}_j$
back_aligned_approach .....→	$f_{\text{route}}(\mathcal{A}_i, \mathcal{A}_j) \leq e_{\text{route}} \wedge f_{\text{aligned}}(\mathcal{A}_i, \mathcal{A}_j) = 1 \wedge$ $f_{\text{position}}(\mathcal{A}_i, \mathcal{A}_j) = v \wedge f_{\text{distance}}(\mathcal{A}_i, \mathcal{A}_j) < -e_{\text{dist}}$	$v$ indicate that $\mathcal{A}_i$ is behind the agent $\mathcal{A}_j$ , $\mathcal{A}_i$ it's the subject if it follows the $\mathcal{A}_j$
frontal_approach -...→	$f_{\text{route}}(\mathcal{A}_i, \mathcal{A}_j) \geq (180 - e_{\text{route}}) \wedge f_{\text{aligned}}(\mathcal{A}_i, \mathcal{A}_j) = 1 \wedge$ $f_{\text{position}}(\mathcal{A}_i, \mathcal{A}_j) = v \wedge f_{\text{distance}}(\mathcal{A}_i, \mathcal{A}_j) < -e_{\text{dist}}$	$v$ indicate that $\mathcal{A}_i$ and $\mathcal{A}_j$ are opposite, $\mathcal{A}_i$ it's the subject if it's faster than $\mathcal{A}_j$
opposite_approach -...→	$f_{\text{route}}(\mathcal{A}_i, \mathcal{A}_j) \geq (180 - e_{\text{route}}) \wedge f_{\text{aligned}}(\mathcal{A}_i, \mathcal{A}_j) = 0 \wedge$ $f_{\text{position}}(\mathcal{A}_i, \mathcal{A}_j) = v \wedge f_{\text{distance}}(\mathcal{A}_i, \mathcal{A}_j) < -e_{\text{dist}}$	$v$ indicate that $\mathcal{A}_i$ and $\mathcal{A}_j$ are opposite, $\mathcal{A}_i$ it's the subject if it's faster than $\mathcal{A}_j$
approach →	$f_{\text{distance}}(\mathcal{A}_i, \mathcal{A}_j) < -e_{\text{dist}} \wedge \exists [t_{s'}, t_{e'}] \subseteq [t_{s'}, t_e] \cdot t_{e'} - t_{s'} > 1 \wedge$ $(\text{back\_aligned\_approach}(\mathcal{A}_i, \mathcal{A}_j, t_{s'}, t_{e'}))$	$\mathcal{A}_i$ it's the subject if it's faster than $\mathcal{A}_j$
flanking →→	$\vee \text{frontal\_approach}(\mathcal{A}_i, \mathcal{A}_j, t_{s'}, t_e) \vee \text{opposite\_approach}(\mathcal{A}_i, \mathcal{A}_j, t_{s'}, t_e)$ $f_{\text{route}}(\mathcal{A}_i, \mathcal{A}_j) \leq e_{\text{route}} \wedge f_{\text{position}}(\mathcal{A}_i, \mathcal{A}_j) = v$	$v$ indicate that $\mathcal{A}_i$ is flanking the agent $\mathcal{A}_j$ , $\mathcal{A}_i$ it's the subject if it's faster than $\mathcal{A}_j$
opposite_flanking -...→→	$f_{\text{route}}(\mathcal{A}_i, \mathcal{A}_j) \leq (180 - e_{\text{route}}) \wedge f_{\text{position}}(\mathcal{A}_i, \mathcal{A}_j) = v$	$v$ indicate that $\mathcal{A}_i$ is flanking the agent $\mathcal{A}_j$ , $\mathcal{A}_i$ it's the subject if it's faster than $\mathcal{A}_j$

In the table 2 we report the concrete events we defined: in the first column, with the name we gave to each event, we have indicated a legend of the corresponding graphical representation. This representation will be used in the examples that we will show from now on. To search the events into the labelled graph we have defined a *seeker* that have for each event P the same general skeleton, of which we present the pseudo-code in figure 32. As we can see, the seeker simply annotates when an event starts and then updates the temporal field of the event until the main condition of the template is verified or when the condition in that instant is not verified but is verified in the next one. We can note how this procedure need one visit of the labelled graph to find all the events.

```

1: procedure P_EVENT_SEEKER( $G^L$ )
2:   events_resulting =  $\emptyset$ 
3:   event_repository =  $\emptyset$ 
4:   for all  $t \in T$  do
5:     for all  $(\mathcal{A}_i^t, \mathcal{A}_j^t) \in E$  do
6:       condition_verified = condition for P is verified
or is verified in  $t + 1$ 
7:       event = event_repository( $(\mathcal{A}_i^t, \mathcal{A}_j^t)$ )
8:       if event =  $\emptyset$  then
9:         if condition_verified then
10:          put(event_repository,  $(\mathcal{A}_i^t, \mathcal{A}_j^t)$ , P)
11:        end if
12:       else
13:         if condition_verified then
14:          update(event_repository( $\mathcal{A}_i^t, \mathcal{A}_j^t$ ))
15:        else
16:          remove(event_repository( $\mathcal{A}_i^t, \mathcal{A}_j^t$ ))
17:          events_resulting = events_resulting  $\cup$ 
event
18:        end if
19:       end if
20:     end for
21:   end for
22:   return events_resulting
23: end procedure

```

Figure 32: Generic seeker structure for an event template P





## THE STATIC INTERACTION PATTERN MINING ALGORITHM

---

The framework presented in the previous section can analyse trajectories' data to retrieve some useful information about the events involving the agents of a trajectories dataset. Those data refer to information about the interactions of pairs of agents and do not give directly other information about the presence of some more complex schemes that occur frequently. However we can expect to find some events that appear frequently together: for instance, into a mobility context should be intuitively to find as a frequent pattern a certain number of agents that maintain between them the same distance. This pattern is shown, using the graphical formalism introduced in 3.2.4.1, in the left part of figure 33. Even if these are a more complex objects instead of single pairs of events, these schemes are still too dependant from the specific agents. Thus should be more interesting searching for group of those schemes, trying to find the ones that appears frequently.

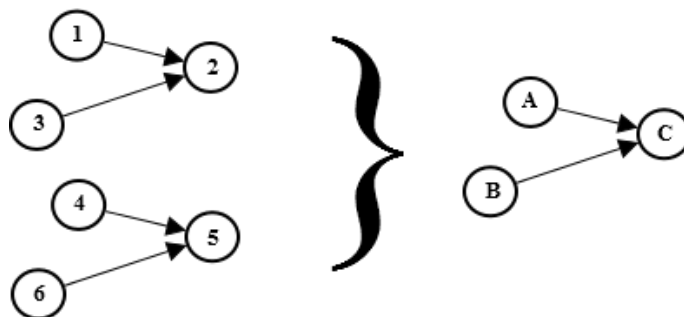


Figure 33: On the left two combinations of contemporary events with different agents, representing the same complex scheme.

On the right, in figure 33, we can observe the generalized scheme of the instances shown on the left, assuming that  $\{A, B, C\}$  are variables and not unique agents' identifiers. We will require that each group of events with concrete agents, like the ones in the left part of the example, must have specific characteristics of:

**PERSISTENCE** groups of contemporary events between agents (*static pattern instances* in figure 34) must have a duration over a certain temporal threshold, to limit our search to the patterns with a significant duration.

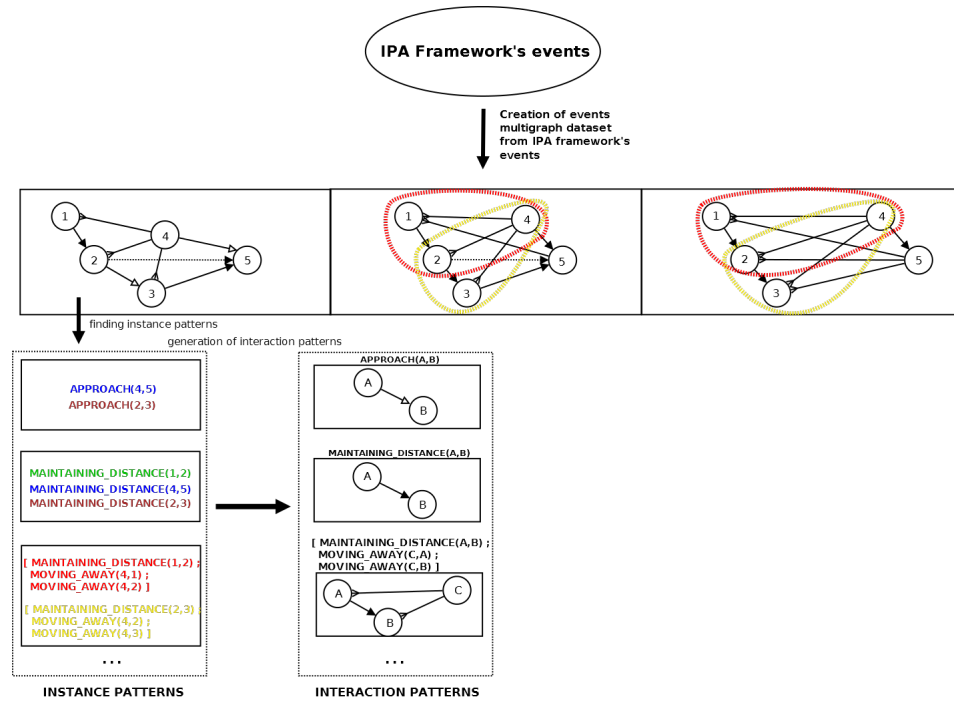


Figure 34: Schematic summary of the work tackle in this chapter.

**MAXIMALITY** group of contemporary events between agents must have the maximal number of persistent and contemporary event.

**TEMPORAL CONTINUITY** the elements of the set containing the same pattern's scheme, but instantiated with different agents (*static interaction patterns* in figure 34), must be seen for an interval with duration over a certain temporal threshold to be considered as frequent.

In figure 34, we show what will be the main topic in this chapter: starting from the events generated by IPA framework we will search for static interaction patterns, that are a generalization of frequently seen contemporary group's events between agents.

#### 4.1 INSTANCE PATTERNS AND INTERACTION PATTERNS

To simplify the search of the objects that we have informally introduced in the summary of the chapter, we define a transformation of the initial dataset: the new dataset will be created starting from the set  $\mathcal{E}\mathcal{J}$  (defined in 12) to make explicit what are the ongoing events between pairs of agents. We have so created, for each instant of the temporal scope  $T$ , a multigraph where exists an edge between two agents for each ongoing events between them in that instant: this information is simply to recover, since we remind that the events are in the form

$P(\mathcal{A}_{id_i}, \mathcal{A}_{id_j}, t_s, t_e)$  with  $[t_s, t_e] \subseteq T$ . So, we can define formally this new structure called *event multigraph set*.

**Definition 14** (Event multigraph set). *Given a set of event instances  $\mathcal{E}\mathcal{J}$ , we can construct a set of consecutive direct multigraph called event multigraph set  $\mathcal{E}\mathcal{M}\mathcal{S} = \{\mathcal{E}\mathcal{M}_i\}_{i \in T}$  where each event multigraph  $\mathcal{E}\mathcal{M}_i = \langle \mathcal{N}_i^{\mathcal{E}\mathcal{M}}, \mathcal{E}_i^{\mathcal{E}\mathcal{M}} \rangle$  is defined as:*

1.  $\mathcal{N}_i^{\mathcal{E}\mathcal{M}} = \{\mathcal{A}_{id} \in \mathcal{A} \mid i \in T_{id}\}$
2.  $\mathcal{E}_i^{\mathcal{E}\mathcal{M}} = \{(\mathcal{A}_{id_1}, \mathcal{A}_{id_2}, P) \mid P(\mathcal{A}_{id_1}, \mathcal{A}_{id_2}, t_s, t_e) \wedge i \in [t_s, t_e]\}$ .

In figure 35 we show an example of graph's construction, taken from the example made in the introduction chapter: in the upper part there are some examples of events found with the IPA framework, that are highlighted in the various multigraphs in the lower part of the figure, where all the events found are visible with the graphical formalism defined in the previous chapter.

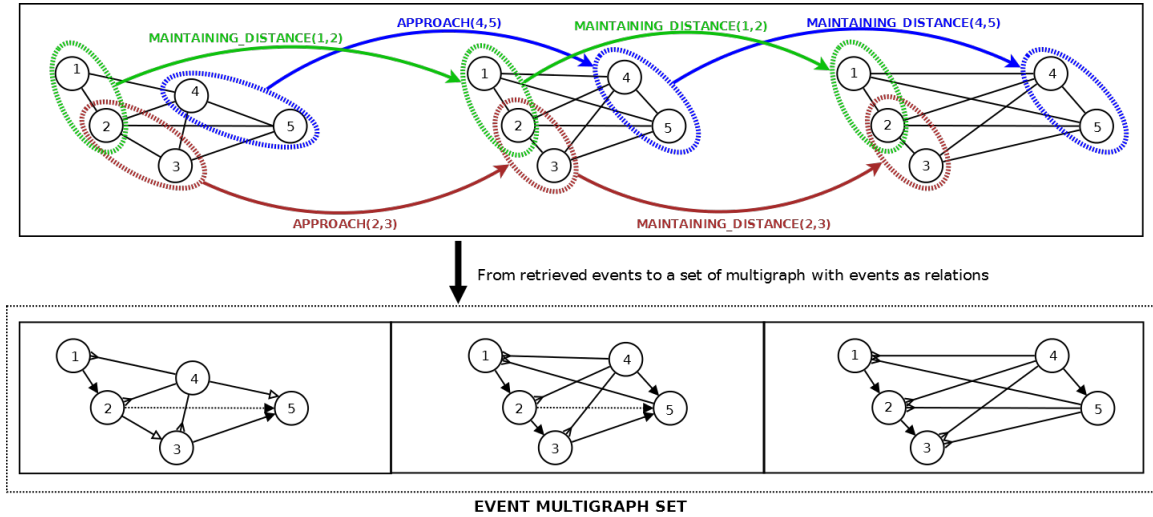


Figure 35: From the events found with the IPA framework we construct a series of multigraphs in which edges represent the ongoing events between neighbors. The meaning of the edges in the multigraphs representation is given in the concrete events table in the previous chapter.

The neighborhood relation in these graphs is implicit, because the presence of an event between a pair of agents means that both are neighbors each other; on the contrary it could not exist an event between them. Moreover each of these graphs gives also immediately the information related to the ongoing events between each pair of agent and also their relation subject/object. We want remark that this is a structure introduced only to simplify the search of the structures that we will define: in a more general definition the problem can be restated using

directly the set  $\mathcal{EJ}$ .

In these graphs we search all the *static instance patterns* defined as:

**Definition 15** (Static Instance Pattern). *Given a minimum interval length  $t_{\min}$ , a set of event instances  $ip = \{e_1, \dots, e_k\} \subseteq \mathcal{EJ}$  is said to be an instance pattern if the following properties are satisfied:*

1. (Duration)  $I_{ip} = \bigcap_{i=1}^k \text{interval}(e_i) \wedge |I_{ip}| \geq t_{\min}$ ;
2. (Connectedness)  $\forall e_i, e_j \in pi, \exists \{e^1, \dots, e^m\} \subseteq pi. e^1 = e_i \wedge e^m = e_j \wedge \forall 1 \leq l < m. \text{agents}(e^l) \cap \text{agents}(e^{l+1}) \neq \emptyset$ ;
3. (Maximality)  $e \in \mathcal{EJ} \wedge |I_{ip} \cap \text{interval}(e)| \geq t_{\min} \wedge \text{agents}(e) \subseteq \bigcup_{i=1}^k \text{agents}(e_i) \Rightarrow e \in ip$ ;

where  $\text{interval}(P(\mathcal{A}_i, \mathcal{A}_j, a, b)) = [a, b]$  and  $\text{agents}(P(\mathcal{A}_i, \mathcal{A}_j, a, b)) = \{\mathcal{A}_i, \mathcal{A}_j\}$ .

In this way we identify substructures which represent persistent and maximal configurations of agents that can be found in at least  $t_{\min}$  consecutive multigraphs of  $\mathcal{EMS}$ . However, those instances are still too dependant from the real agent, while we would recognize the patterns deriving by a generalization of a set of equivalent instance patterns, independently from the identifiers of the agents found in the dataset. To do this we give the definition of isomorphic instances.

**Definition 16** (Isomorphic Instances). *Given two static pattern instances  $pi_1$  and  $pi_2$ , we say that they are isomorphic instances, denoted with  $pi_1 \simeq pi_2$ , if there exists a bijective function  $\phi : \mathcal{A} \rightarrow \mathcal{A}$  such that:  $\forall \mathcal{A}_1, \mathcal{A}_2 \in \mathcal{A}. \forall a, b, a', b' \in T.P(\mathcal{A}_1, \mathcal{A}_2, a, b) \in pi_1 \Leftrightarrow P(\phi(\mathcal{A}_1), \phi(\mathcal{A}_2), a', b') \in pi_2$ .*

Recognize the equivalent instances is the operation necessary to abstract from the concrete pattern found: in this way, different concrete events that represent the same event can be grouped with the others representing the same type. So the set of all equivalent instance patterns is the complex object that we were searching for.

**Definition 17** (Frequent Static Interaction Pattern). *Given the set  $\mathcal{PJ}$  of static pattern instances inferred from all event instances  $\mathcal{EJ}$ , we define the set of static interaction patterns in  $\mathcal{PJ}$  as  $S\mathcal{P} = \{[pi]_{\simeq} \mid pi \in \mathcal{PJ}\}$ , where  $[pi]_{\simeq}$  denotes the equivalence class induced by  $\simeq$  that contains  $pi$ .*

We define the temporal support of static interaction pattern  $sp \in \mathcal{SP}$  as

$$\text{supp}(sp) = \frac{|\bigcup_{pi \in \mathcal{PI} \wedge [pi]_{\simeq} = sp} \text{interval}(pi)|}{|T|}$$

A static interaction pattern  $sp$  is said to be frequent iff  $\text{supp}(sp) \geq t_{\text{supp}}$ , where  $t_{\text{supp}} \in [0, 1]$  is a minimum temporal support threshold. Finally, we denote with  $\mathcal{FS}\mathcal{P}$  the set of all frequent static interaction patterns in  $\mathcal{SP}$ .

An important point in the previous definition is the one relative to the temporal support. This because the property we would with this type of support implies that the important aspect of an interaction pattern is the one to be continuous, instead of having some peak of instances but very limited into the time scope considered. Moreover in this way we are using the temporal support as an application of the Apriori property: in fact we won't use the instances of size  $k$  agents of an infrequent interaction pattern to try to generate instances of size  $k + 1$ , since any instance of size  $k + 1$  containing that instance cannot generate an interaction pattern with greater temporal support. In figure 36 we show a schematic representation of the notions of time persistence for a static pattern instance and of temporal support for a static interaction pattern.

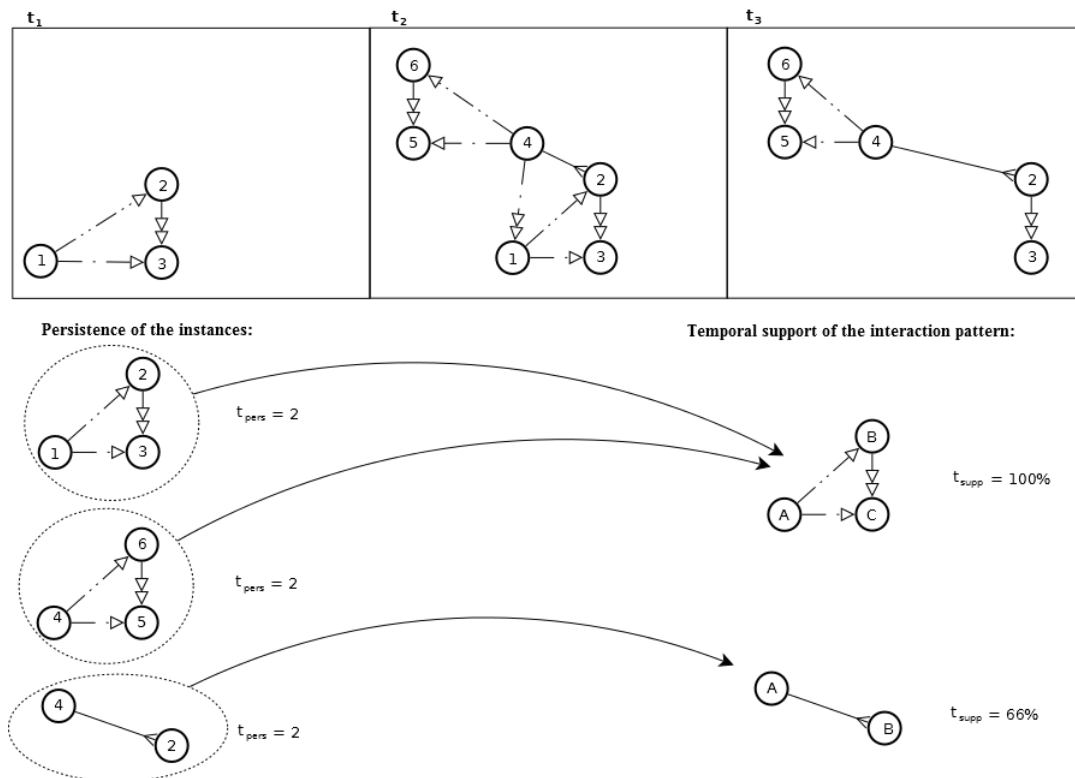


Figure 36: Persistence and temporal coverage of a simple example dataset of three instants.

In the figure we can see clearly the differences between persistence time and temporal support: persistence is related to the number of instants for which we have seen a specific static pattern instance; in particular the ones indicated in the left part in the bottom of the figure have been seen for two consecutive instants ( $t_1$  and  $t_2$  the first,  $t_2$  and  $t_3$  the others). The temporal support is related to the static interaction pattern and is the percentage of instant for which at least one instance of that interaction pattern have been recognized; in the example the first interaction pattern has temporal support of the 100%, since it has at least one instance in every instant, while the second interaction pattern has temporal support of the 66%, since it has at least one instance in the last two instants.

#### 4.2 EQUIVALENCE BETWEEN INTERACTION PATTERNS

Recognizing that two instances are equivalent is a problem amenable to the graph isomorphism problem. In fact if we choose a simple renaming function, as the one shown in figure 37, we could probably create different equivalent groups representing the same generic pattern.

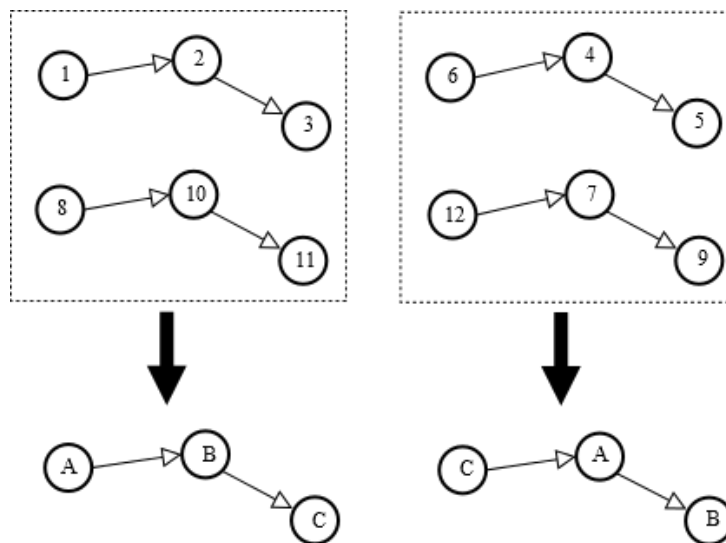


Figure 37: Two static pattern instances belonging to the same static interaction pattern, recognized as different.

However, any problem related to graphs isomorphism is known to be NP-Hard and unfortunately we have to face with this problem to recognize the same instance patterns. Since we can't avoid this problem, we developed an heuristic to try to reduce considerably the number of comparisons necessary to know if two graphs are isomorphic, for this particular case in which the vertices are uniquely identifiable and the edges are labelled. We based our strategy on two main considerations: first of all,

since we can represent our instances as labelled multigraphs, if two vertices have the same number of outgoing edges but have different number of edges with the same label, the two graphs cannot be isomorphic: furthermore, looking at the adjacency matrices of two graphs representing static pattern instances, only the vertices that have the same number of outgoing edges with the same labels and that have the same incoming degree can represent a possible mapping for two vertices. This intuition is shown in the example of figure 38. Then we can use this fact to generate only the necessary subsets of permutations of the adjacency matrix of one of the two matrices to know if the two graph are isomorphic.

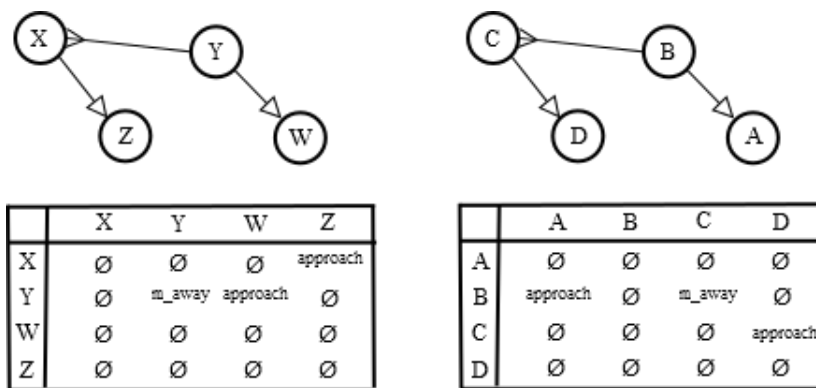


Figure 38: Two isomorphic graph with their adjacency matrices

In figure 38 we have two isomorphic pattern instances with the corresponding adjacency matrices. In this case we are not interested to all the possible permutation of one of the two matrices: in fact we know that if two rows do not contain the same labels' groups we can't have any isomorphism. In this case the heuristic generates only the permutation of the vertices with possible mappings for the assignment of the nodes  $X = C, Y = B, W = A \vee D, Z = A \vee D$ , reducing the number of permutations from  $4! = 16$  to 2. The procedure still costs, in the worst case, as the naive version of the method, but having matrices belonging to real cases we should reduce the method's complexity to have an acceptable computation time. In figure 39 we show the pseudocode of the heuristic we propose. Finally we want to highlight only one aspect: in the procedure we exclude any isomorphism if the nodes are not mappable, that is possible knowing the labels of the edges. Thus the labelling of the edges and the uniqueness of nodes can help us to reduce a lot the number of possible comparison we could have to do, and then the number of times we have to permute the mappings found.

```

1: procedure INP_ISOMORPHISM_HEURISTIC(INP, INPcmp)
2:   if INP and INPcmp have different number of event, or
   same number of event but of different type then
3:     return false
4:   end if
5:   mappings =  $\emptyset$ 
6:   adjacencyINP = get_adjacency_matrix(INP)
7:   adjacencycmp = get_adjacency_matrix(INPcmp)
8:   for all row rcmp  $\in$  adjacencycmp do
9:     for all row rINP  $\in$  adjacencyINP do
10:      if rcmp contains the same labels' groups of rINP
then
11:        maprINP = get_mapped_nodes(mapping, rINP);
12:        maprINP = maprINP  $\cup$  node(rcmp)
13:      end if
14:    end for
15:  end for
16:  if mappings  $\neq \emptyset$  then
17:    for all permutation permutation(mapping,  $\cdot$ ) of
    the mappings do
18:      if adjacencyINP =
      permutation(mapping, adjacencyINP) then
19:        return true
20:      end if
21:    end for
22:  end if
23:  return false
24: end procedure

```

Figure 39: Generalized pattern isomorphism heuristic pseudo-code



### 4.3 THE SIPM ALGORITHM

All the previous elements have been introduced to define the algorithm that works on the event multigraph dataset to extract frequent static interaction patterns. After having introduced the notions at the basis of static interaction patterns, can be clearer the motivation that has led us to develop a new algorithm instead of using one of the already developed for graph mining as, for example, the well-known gSpan[23]. We can summarize those motivations in two points:

**CORRELATION BETWEEN GRAPHS** the structure we used to simplify our search is given by a set of consecutive multigraphs representing consecutive situation between agents; then the order in which they are treated must be considered by the algorithm, to explicit the temporal properties of the instances we are searching for;

**SEARCH FOR GENERAL PATTERNS FROM THEIR INSTANCES** in a graph mining algorithm the dataset is given by a set of  $n$  graphs in which search for frequent structures: we want instead to find not the frequent structures in the event multigraphs set, but to find one more general structure that we can induce from the frequent persistent instances representing it.

In a certain sense the inoperability and the not adaptability of the algorithms proper of the graph mining to find the structures we were searching, are others confirmation of the fact that the use of  $\mathcal{EM}\mathcal{S}$  is only a facilitation of representation of our data. Now we have all the elements to present the Static Interaction Pattern Mining (SIPM) algorithm in figure 40. In the line 4 we extract from the event multigraph dataset all the pairs of vertices that are instance patterns, following the definition 15; in the line 7 we execute the calculus for the equivalence classes of the instance patterns. The cycle starting from 8 is the one that recognize an interaction pattern as frequent, and so all its instances, if the interval set of the instance patterns cover temporally a certain percentage of the entire interval considered. A key step of the algorithm is the generation of the candidates, in the line 14. We include the pseudo-code of this procedure into the figure 42.

Before inspecting the pseudocode of the candidate generation procedure, we show in figure 41 a simplified example of its behaviour, starting from a set of three instance patterns of two agents: if we consider the pair  $(1,2)$ , we start in the first step collecting all the 2-agents instances belonging to frequent interaction patterns that involve the elements of the pair  $(1,2)$ ,

```

1: procedure SIPM( $\mathcal{EMS}$ ,  $t_{supp}$ ,  $t_{min}$ )
2:    $k = 2$ 
3:    $\mathcal{FSP} = \emptyset$ 
4:    $C_k = \text{extract\_2\_ip}(\mathcal{EMS}, t_{min})$ 
5:   while  $C_k \neq \emptyset$  do
6:      $I_k = \emptyset$ 
7:      $\mathcal{SP}_k = \text{merge\_isomorphic\_instances}(C_k)$ 
8:     for all  $sp \in \mathcal{SP}_k$  do
9:       if  $\text{supp}(sp) \geq t_{supp}$  then
10:         $I_k = I_k \cup (\bigcup_{[ip] \simeq sp} ip)$ 
11:         $\mathcal{FSP} = \mathcal{FSP} \cup sp$ 
12:       end if
13:     end for
14:      $C_{k+1} = \text{candidate\_generation}(I_k, t_{min})$ 
15:      $k = k + 1$ ;
16:   end while
17:   return  $\mathcal{FSP}$ 
18: end procedure

```

Figure 40: Static Interaction Pattern Mining pseudo-code

and then we recombine them for the step two. To know if they are the maximal instance patterns we recombine the resulting 3-agents intermediate results to get the final result in step 3. In this case all the instances of step 2 are deleted from the final result, because they are not the maximal ones that can be generated with the agents (1,2,3).

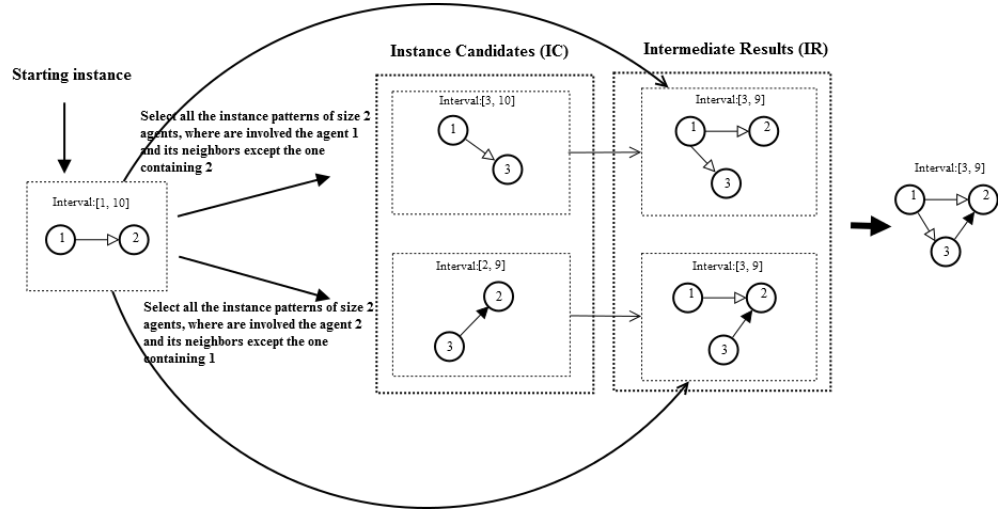


Figure 41: An example of the generation of 3-agents instance patterns starting from 2-agents instance patterns.

We can now explain the procedure for the candidates generation: first of all, given one instance pattern, we retrieve all the instance patterns that are correlated to the one given in this

way: we get all the frequent instance patterns that share with the selected one all the agents minus one, where the last agent is given by one neighbor of one of the other agents remained: this is done for all the agents in the original pattern (line 5), as exemplified in the figure 41. Then we use a set of intermediate results to collect the instance patterns of  $k$  agents that are persistent and that we can merge to create a new candidate of  $k + 1$  agents. These are intermediate results because with the first step of merging we do not found the maximal instance patterns, while into the second recombination, lines 12 - 16, we merge the intermediate results (if they respect the persistence's property) to obtain as final result only the maximal patterns. Finally the generators of the new pattern are removed from the set of the final results (because they are not maximal), if they are used in the recombination phase.

```

1: procedure CANDIDATE_GENERATION( $I_k, t_{\min}$ )
2:    $C = \emptyset$ 
3:   for all  $ip_i \in I_k$  do
4:      $IR = \emptyset, CR = \emptyset$ 
5:      $IC = \{ip_j \in I_k \mid \text{agents}(ip_j) = \{\text{agents}(ip_i) \setminus \{\text{id}_b\}\} \cup$ 
       $\text{id}_c, \{\text{id}_a, \text{id}_b\} \subseteq ip_i, \text{id}_c \in \mathcal{N}(\text{id}_a) \setminus \text{agents}(ip_i)\}$ 
6:     for all  $ip_j \in IC$  do
7:       if  $\text{interval}(ip_i) \cap \text{interval}(ip_j) \geq t_{\min}$  then
8:          $IR = IR \cup \{\text{merge}(ip_i, ip_j)\}$ 
9:          $CR = CR \cup \{\text{merge}(ip_i, ip_j)\},$ 
10:      end if
11:    end for
12:    for all  $ir_i, ir_j \in IR$  s.t.  $\text{agents}(ir_i) = \text{agents}(ir_j)$  do
13:      if  $|\text{interval}(ir_i) \cap \text{interval}(ir_j)| \geq t_{\min}$  then
14:         $CR = \{CR \cup \{\text{merge}(ir_i, ir_j)\}\} \setminus \{\{ir_i\} \cup \{ir_j\}\}$ 
15:      end if
16:    end for
17:     $C = C \cup CR$ 
18:  end for
19:  return  $C$ 
20: end procedure

```

Figure 42: Candidate generation pseudo-code

Analyzing the pseudo-code shown in figure 40, we end this chapter stating the following theorem on the time complexity of the SIPM algorithm.

**Theorem 1** (SIPM complexity). *Given a set of event instances  $\mathcal{E} \mathcal{J}$  observed in a time scope  $T$  and a minimum persistence time for each instance pattern of  $t_{\min}$ , the time complexity of the SIPM algorithm*

is  $O(m \cdot (4n)^n \cdot n^3)$ , where  $n$  is the maximum number of different agents present in one instant and  $m = \lfloor \frac{|T|}{t_{\min}} \rfloor$ .

*Proof.* Indicating with  $n_{ip}^k$  the number of instance pattern found at the  $k$ -th iteration, and indicating the cost in function of the  $k$ -th iteration, we can state that the cost of the algorithm is given by the following sum:

$$c_{SIPM} = \sum_{k=2}^p (c_{mig}^k + c_{cc}^k + c_{cg}^k) \quad (13)$$

where  $p$  is the maximum number of iteration executed by the algorithm. The summands refer to the cost of the methods contained in the main procedure: the cost of the *merge\_isomorphic\_groups* ( $c_{mig}^k$ ) method, the one for the calculus of the frequent interaction patterns ( $c_{cc}^k$ ) and finally the *candidate\_generation* ( $c_{cg}^k$ ). Indicating with  $n = \max\{|N_i^{EM}| \mid i \in T \wedge \langle N_i^{EM}, E_i^{EM} \rangle \in \mathcal{EMS}\}$  then, by definition of instance pattern, the algorithm can generate patterns of size at most  $n$  in the worst case (if all agents would be connected to everyone else), and this is in the worst case the maximum number of iteration of the algorithm. Then we can rewrite the cost as

$$c_{SIPM} = \sum_{k=2}^n (c_{mig}^k + n_{ip}^k + c_{cg}^k) \quad (14)$$

The cycle to check the temporal support is done in linear time in the worst case, the one in which each instance pattern is associated to a different interaction pattern. So we have  $c_{cc}^k = n_{ip}^k$ . To merge the isomorphic group, in the worst case, we have to check for multidimensional directed clique of size  $k$ , so the cycle is composed from a comparison part and the cost of the naive isomorphism between graph,  $c_{mig}^k = (n_{ip}^k)^2 \cdot k! \cdot k^2$ . The part relative to the comparison cycle is  $(n_{ip}^k)^2$  because again we are considering the worst case, the one in which each instance pattern belongs to a different interaction pattern. Indicating with  $n_{ic}^k$  the number of candidates for the generation of instance patterns with size  $k + 1$ , and considering that in the worst case  $n_{ic}^k < n_{ip}^k$ , since  $|IC| < |I_k|$  (because the instance candidates do not contain surely at least the the starting pattern), we have that  $c_{cg}^k = n_{ip}^k \cdot (n_{ic}^k)^3$ . With these consideration about the cost of the various operations we can restate the cost of the algorithm as:

$$c_{SIPM} = \sum_{k=2}^n (n_{ip}^k + (n_{ip}^k)^2 \cdot k! \cdot k^2 + n_{ip}^k \cdot (n_{ic}^k)^3) \quad (15)$$

$$= \sum_{k=2}^n n_{ip}^k + \sum_{k=2}^n (n_{ip}^k)^2 \cdot k! \cdot k^2 + \sum_{k=2}^n n_{ip}^k \cdot (n_{ic}^k)^3 \quad (16)$$

$$\leq \sum_{k=0}^n n_{ip}^k + \sum_{k=0}^n (n_{ip}^k)^2 \cdot k! \cdot k^2 + \sum_{k=0}^n n_{ip}^k \cdot (n_{ic}^k)^3 \quad (17)$$

It remains to give an estimation of  $n_{ip}^k$ : in the worst case at the  $k$ -th iteration we can have as possible patterns all the combination of  $n$  agent of size  $k$ , then  $n_{ip}^k = \binom{n}{k}$ , while the possible instance patterns are given multiplying this quantity for  $m = \lfloor \frac{|T|}{t_{min}} \rfloor$ , that gives the maximum number of instance patterns that can be found with  $n$  agents in a time scope of length  $|T|$  (we are supposing that each instance pattern has the minimum duration, to maximize the number of patterns). Thus, using the binomial properties  $\sum_{k=0}^n \binom{n}{k} = 2^n$  and  $\sum_{k=0}^n \binom{n}{k}^2 = \binom{2n}{n}$  we can state that:

$$c_{SIPM} \leq m \cdot (2^n + n \cdot \binom{2n}{n} \cdot n! \cdot n^2) + n \cdot (2^n \cdot r_n^3) \quad (18)$$

where  $r_n^3 = \max_{k=2 \dots n} \{(n_{ic}^k)^3\}$ . To simplify the formula we can introducing an overestimation of the cost, in fact  $n! < n^n$  and then:

$$c_{SIPM} \leq m \cdot (2^n + n \cdot \frac{2n!}{n! \cdot n!} \cdot n! \cdot n^2) + n \cdot (2^n \cdot r_n^3) \quad (19)$$

$$= m \cdot (2^n + n \cdot \frac{2n!}{n!} \cdot n^2) + n \cdot (2^n \cdot r_n^3) \quad (20)$$

$$= m \cdot 2^n + m \cdot \frac{(2n)^{2n}}{n^n} \cdot n^3 + m \cdot n \cdot 2^n \cdot r_n^3 \quad (21)$$

Then, with algebraic passages we can write:

$$c_{SIPM} = m \cdot 2^n + m \cdot (4n)^n \cdot n^3 + m \cdot n \cdot 2^n \cdot r_n^3 \quad (22)$$

and then the cost, with the growing of  $n$ , is dominated by the middle factor; so we can state that:

$$c_{SIPM} = O(m \cdot (4n)^n \cdot n^3) \quad (23)$$

□

The previous theorem gives us surely an overestimation respect to the real cases in which the algorithm is used: however we can note how the cost of the algorithm is somehow a mix of the components given by the use of a variant of the Apriori algorithm (that has a complexity of  $O(m \cdot 2^n)$  with  $m$  transactions and  $n$  items) and the cost of the naive graphs isomorphism algorithm (that has a complexity of  $O(n! \cdot n^2)$ , with graphs of  $n$  vertices). Our heuristic for the isomorphism and the pruning technique based on the temporal support of the interaction patterns, in union with the difficulty of finding instances patterns with the minimum persistence's threshold with the increasing of the iterations, act on the factor  $(4n)^n$  and , how we will see in the experiments chapter, seems to work well in real cases.

## THE EVOLVING INTERACTION PATTERNS MINING ALGORITHM

In the previous chapter we have introduced an algorithm to find all the frequent interaction patterns, deriving from the interaction analysis of agents' pairs. Those data indicate frequent behaviours seen between agents; even if we can already extrapolate some useful information about the characteristics of those patterns, we could ask if they have some consequentiality. For example if we are analysing data from a dataset of moving vehicles, we should not be surprised to find some sequences like the one represented in figure 43.

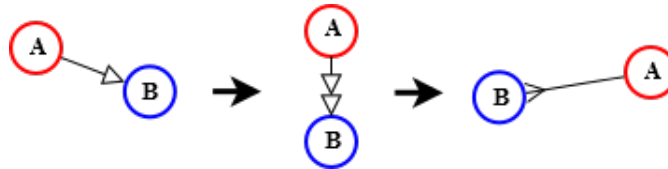


Figure 43: An example of sequence of static interaction patterns

The previous figure represents an overtaking: in fact we have a first phase in which an agent A approaches the agent B, a second phase in which the agent A is flanking the agent B and in the third and last phase the agent A that is moving away from B. Our goal, in this chapter, is to use the results of the SIPM algorithm to find objects like this. So, if in the previous chapter we found frequent behaviour between agents, we will now make explicit their natural evolution in time, still maintaining the generalization of the agents.

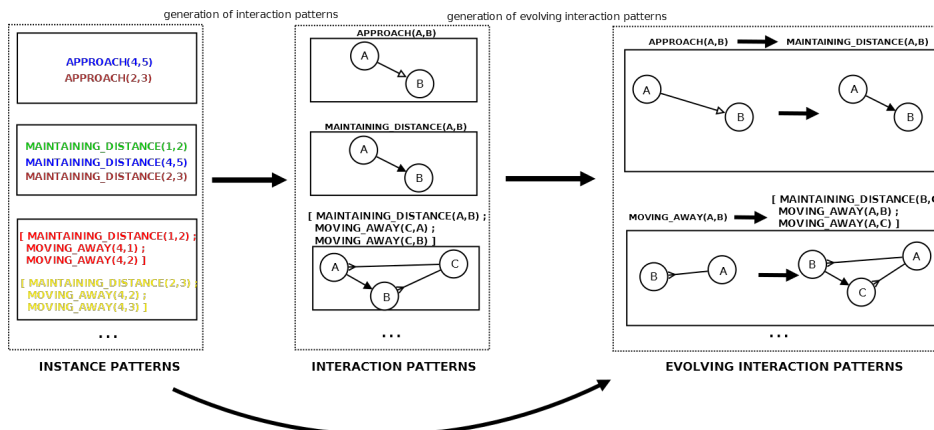


Figure 44: Schematic summary of the work tackled throughout this chapter.

In figure we can see schematically the aim of this section: from interaction patterns and their instances, generate sequence of interaction patterns correlated in time.

### 5.1 EVOLVING INTERACTION PATTERNS

The main question now is: how to use static interaction patterns and their instances to find sequences of frequent interaction patterns?

We explain our idea following the schematic example in figure 45.

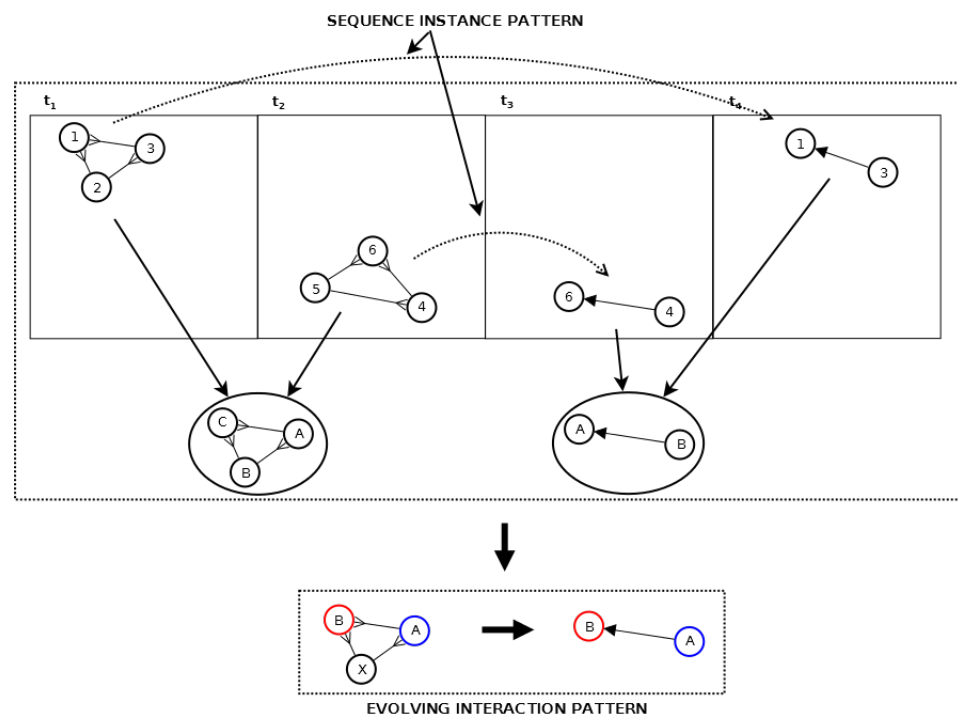


Figure 45: Schematic example of creation of an evolving interaction pattern

We exemplified how we want to proceed with some simplification to be clearer: suppose to have a situation like the one represented in the previous figure where, in four consecutive instants, we find four static pattern instances belonging to the interaction patterns close into the ellipses. We can see how the indicated pairs of instances are related between them, because share a subset of the agents: in the first case the agents with identifier 1 and 3, and in the other case the agents 4 and 6. If we suppose that the time window in which we have found that relation be enough significant for this context, we could state that the second instances patterns, in instants  $t_3$  and  $t_4$ , are evolutions of the instances found respectively in the instants  $t_2$  and  $t_1$ . Furthermore, the two sequences share another feature:



in each sequence the first instance pattern and the second belong to the same static interaction pattern. We can conclude that from data emerge a frequent evolution of the interaction pattern from the first state into the second one. It is interesting that the resulting *evolving interaction pattern* has been constructed starting from instance patterns, to recognize sequences present in the data, and using the information given by the labelling of the corresponding static interaction patterns to create a sequence in which we want to highlight the agents protagonists of the evolution. The situation represented in the example involves only two instance patterns, but we can extend the search, into a specified temporal window, for sequences of  $m$  patterns. Anyhow we are interested to sequences with particular characteristics about agents' persistence and continuity. All these features are explained with the following definition.

**Definition 18** (Sequence pattern instances). *Given a set of frequent static patterns  $\mathcal{FSP}$ , a temporal window  $t_{win} \in \mathbb{N}$  and a  $min_{jc} \in [0, 1]$ , a sequence of instances  $spi = (ip_1, ip_2, \dots, ip_k)$ , where  $\forall 1 \leq j \leq k. [ip_j]_{\simeq} \in \mathcal{FSP}$ , is said to be a sequence pattern instance if the following conditions are verified:*

1.  $jc(ip_1, ip_2) \geq min_{jc} \vee agents(ip_1) \subseteq agents(ip_2) \vee agents(ip_2) \subseteq agents(ip_1)$ .
2.  $\forall 1 < i < k : jc(ip_i, ip_{i+1}) \geq min_{jc}$
3.  $\forall 1 \leq i < k, t_{start}^i < t_{start}^{i+1}$
4.  $(t_{start}^k - t_{start}^1) \leq t_{win}$
5.  $\forall 1 \leq i \leq k : kernel(spi) \subseteq agents(ip_i)$

where  $jc(a, b)$  denotes the Jaccard coefficient between  $agents(a)$  and  $agents(b)$ . The set of agents  $kernel(spi) = agents(ip_1) \cap agents(ip_2)$  is called the kernel of the sequence, while the interval of the sequence is defined as  $interval(ip_1, \dots, ip_k) = \bigcup_{i=1 \dots k} interval(ip_i)$ . We denote with  $SPJ$  the set of all the sequence pattern instances of this form.

Some comments on the previous definition: first of all we can see how the first two instances of the sequence have particular characteristics respect to the others of the sequence. In fact they determine the start of the sequence because in two instants, not immediately after but within the time window, we have that:

- A. the set of the agent in the first pattern is contained in the second one (independently from the Jaccard's coefficient value) or viceversa;

- B. we have a significant amount of agents shared from both instances, where the significant amount is given by a threshold indicated with the Jaccard's coefficient value.

This because the evolution of a sequence can be interesting if we see the evolution of an hard core of agents (first case) or of a significant amount of agents that persist through the sequence's patterns.

We only miss the second phase of the process we exemplified in the figure 45: the generalization of the sequence to create a labelling that give us a general sequence to which belong all the sequences with the same form. For this purpose, in a similar manner to what we done in the previous chapter, we define the equivalence between two sequences.

**Definition 19** (Isomorphic sequence instances). *Given two sequence pattern instances  $spi_1 = (pi_1^1, \dots, pi_1^n)$  and  $spi_2 = (pi_2^1, \dots, pi_2^m)$ , we say that they are isomorphic sequence instances, denoted with  $spi_1 \simeq spi_2$ , if (i)  $n = m$ , and (ii) there exists a bijective function  $\phi : \mathcal{A} \rightarrow \mathcal{A}$  such that  $\forall 1 \leq i \leq n. pi_1^i \simeq_\phi pi_2^i$ .*

In the definition above, we notice that the name mapping function  $\phi$  is the same for all static pattern instances involved, meaning that if an agent appears in more than one static instance within a sequence, the *new identities* it will have after the name mapping will be the same across the whole sequence. An example of behaviour of the renaming process is shown in figure 46.

Thus we assign the same label to the agents of  $agents_{kernel}$  for all the instances in the sequence, while the other agents are labelled following an ordering suggested by the mapping between the label of the agent into the corresponding static interaction pattern and the identifiers of the agents themselves. When considering the equivalence, we must take into account also the fact that two instances can be overlapped and, following the definition of sequence pattern, that can exist a sequence in which appear the same agents in more than one interaction pattern but that are not recognized as belonging to the kernel (this fact derives from the condition  $\forall 1 \leq i < k : jc(ip_i, ip_{i+1}) \geq \min_{jc}$ ). An example of those two cases are shown in figure 47.

Since we can recognize all the equivalent sequences, we can define the notions of *evolving interaction pattern* and *frequent evolving interaction pattern*.

**Definition 20** (Evolving interaction patterns). *Given the set  $\mathcal{SPJ}$  of all sequence pattern instances, we define the set of evolving interaction patterns in  $\mathcal{SPJ}$  as the set  $\mathcal{EP} = \{[spi]_{\simeq} \mid spi \in \mathcal{SPJ}\}$  where  $[spi]_{\simeq}$  denotes the equivalence class induced by  $\simeq$  that contains  $spi$ .*

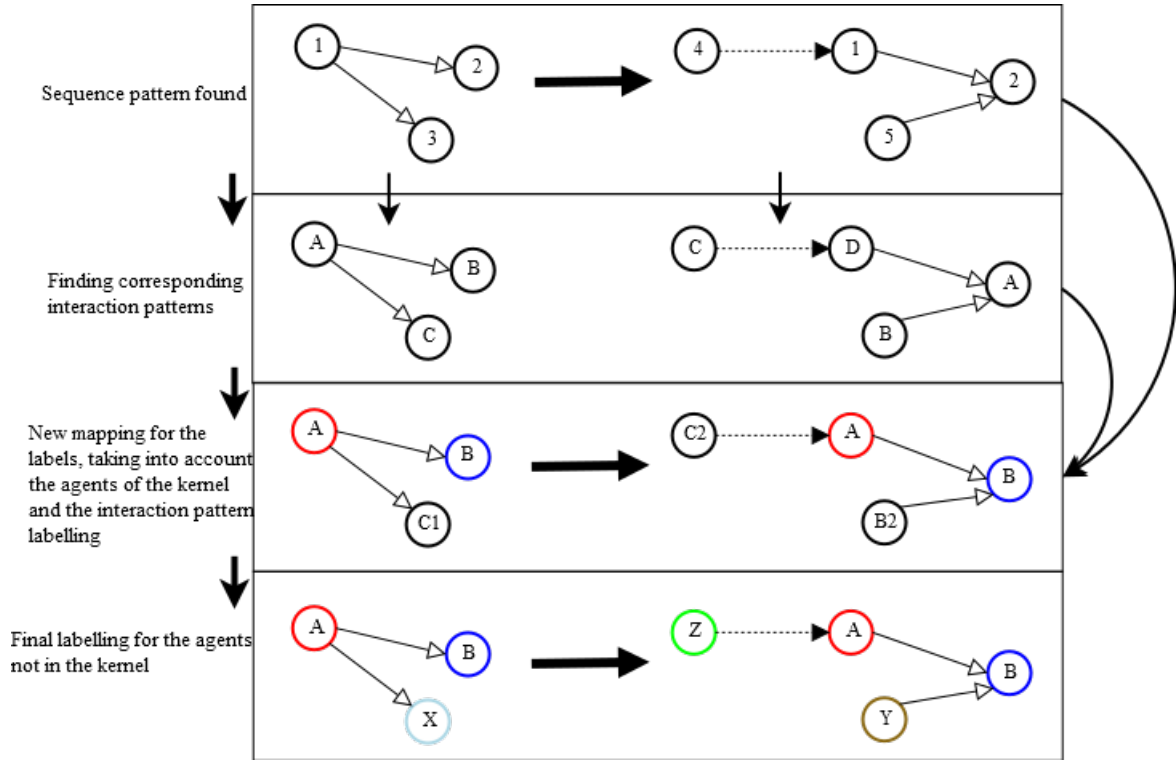


Figure 46: Example of behaviour of the renaming for the sequence of instance patterns with their corresponding interaction patterns.

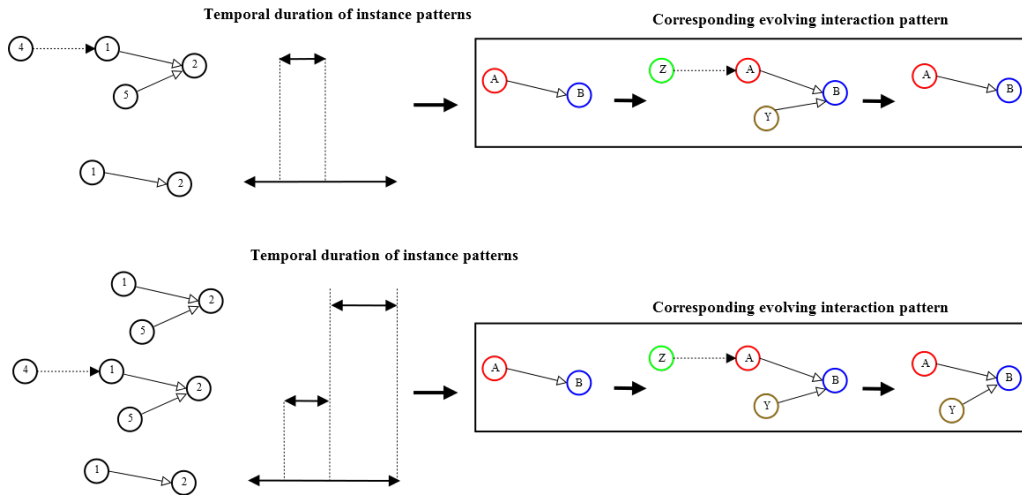


Figure 47: In the first case two subsequent pattern generate an evolving pattern of three elements because the one with three agents is overlapped to the one of two. In the second case the label mapping for the agent Y is maintained in the second and third part of the sequence because the generic agent found in the second part, even if is not part of the kernel of the sequence, still appears in the last part.

We define the temporal support of an evolving interaction pattern  $ep \in \mathcal{EP}$  as

$$\text{supp}(ep) = \frac{\left| \bigcup_{spi \in \mathcal{SPJ} \wedge [spi] \simeq ep} \text{interval}(spi) \right|}{|T|}$$

```

1: procedure EvIPM( $\mathcal{FSP}, \mathcal{PJ}, t_{win}, \min_{jc}, t_{e-supp}$ )
2:    $k = 2$ 
3:    $\mathcal{SPJ}_2^* = \{(p_{i_1}, p_{i_2}) \in \mathcal{PJ}^2 \mid [p_{i_1}]_{\simeq} \in \mathcal{FSP} \wedge [p_{i_2}]_{\simeq} \in \mathcal{FSP} \wedge \text{interval}(p_{i_1}) = [t_s^1, t_e^1] \wedge \text{interval}(p_{i_2}) = [t_s^2, t_e^2] \wedge t_s^1 < t_s^2 \leq t_s^1 + t_{win} \wedge (\text{agents}(p_{i_1}) \subseteq \text{agents}(p_{i_2}) \vee \text{agents}(p_{i_2}) \subseteq \text{agents}(p_{i_1}) \vee \text{jc}(p_{i_1}, p_{i_2}) \geq \min_{jc})\}$ 
4:    $\mathcal{FEP} = \{[spi]_{\simeq} \mid spi \in \mathcal{SPJ}_2^* \wedge \text{supp}([spi]_{\simeq}) \geq t_{e-supp}\}$ 
5:    $\mathcal{SPJ}_2 = \{spi \in \mathcal{SPJ}_2^* \mid [spi]_{\simeq} \in \mathcal{FEP}\}$ 
6:   while  $\mathcal{SPJ}_k \neq \emptyset$  do
7:     for all  $spi = (p_{i_1}, \dots, p_{i_k}) \in \mathcal{SPJ}_k$  do
8:        $[t_s, t_e] = \text{interval}(p_{i_1})$ 
9:        $\mathcal{PJ}_{tail} = \{pi \mid \text{interval}(pi) = [t_1, t_2] \wedge t_1 \in [t_s, t_s + t_{win}] \wedge \text{kernel}(spi) \subseteq \text{agents}(ip) \wedge \text{jc}(p_{i_k}, pi) \geq \min_{jc}\}$ 
10:       $\mathcal{SPJ}_{k+1}^{spi} = \{(p_{i_1}, \dots, p_{i_k}, pi) \mid pi \in \mathcal{PJ}_{tail}\}$ 
11:      end for
12:       $\mathcal{SPJ}_{k+1}^* = \bigcup \mathcal{SPJ}_{k+1}^{spi}$ 
13:       $\mathcal{FEP} = \mathcal{FEP} \cup \{[spi]_{\simeq} \mid spi \in \mathcal{SPJ}_{k+1}^* \wedge \text{supp}([spi]_{\simeq}) \geq t_{e-supp}\}$ 
14:       $\mathcal{SPJ}_{k+1} = \{spi \in \mathcal{SPJ}_{k+1}^* \mid [spi]_{\simeq} \in \mathcal{FEP}\}$ 
15:       $k = k + 1$ 
16:   end while
17:   return  $\mathcal{FEP}$ 
18: end procedure

```

Figure 48: Evolving Interaction Pattern Mining pseudo-code

An evolving interaction pattern  $ep$  is said to be frequent iff  $\text{supp}(ep) \geq t_{e-supp}$ , where  $t_{e-supp} \in [0, 1]$  is the minimum temporal support threshold for evolving patterns. Finally, we denote the set of all frequent evolving patterns with  $\mathcal{FEP}$ .

Even in this definition we have an use of the Apriori principle, given by the temporal threshold that the evolving interaction pattern must verify to be recognized as a frequent one.

## 5.2 THE EVIPM ALGORITHM

Now we can describe the *sequential interaction pattern mining*, shown in figure 48.

The algorithm starts with the creation of sequence instances composed by two static pattern instances (step 3), by following the special case conditions described in Definition 18; then, we keep only those forming frequent evolving patterns (lines 4-5). The main cycle in lines 6-16 performs the same operations for the general case, trying to append a new static pattern (instance) to existing sequences at each step: in line 9 we find all the static pattern instances with starting time into the temporal window, and then we construct all the possible sequences of size  $k + 1$ , creating a new sequence that is composed by the current  $spi$  and one of the possible patterns found in 9. Again we keep only those sequences belonging to frequent evolving patterns, lines 13-14. From the pseudocode of the SeqIPM algorithm we can derive the following theorem:

**Theorem 2** (EvIPM complexity). *Given  $n$  the number of instance patterns contained into  $\mathcal{FSP}$ ,  $t$  the dimension of the temporal window, and  $m$  the maximum number of instance patterns that start in*

the same instant  $i \in T$ , the computational complexity of the EvIPM algorithm is given by  $O(n \cdot m^t)$ .

*Proof.* Indicating with  $m = \max_{i \in T} |\{p_i | p_i \in \mathcal{P} \wedge [i, t_e] = \text{interval}(p_i)\}|$ , in the worst case each combination of patterns in sequence into the chosen temporal window  $t = t_{win}$  of length  $k$  with  $0 \leq k \leq t$  is a valid sequence pattern. If we assume that in each instant start the maximum number of instance patterns  $m$ , we have  $m^t$  possible sequence patterns for one starting sequence: this means that for  $n$  instances ( $n$  possible starting sequences) the procedure cost in the worst case

$$c_{EvIPM} = n \cdot m^t \quad (24)$$

□



## EXPERIMENTS

In this section we present the results of the experiments obtained analyzing two dataset with the IPA framework and the SIPM/EvIPM algorithms. In the last section we will discuss also the performances of the algorithms, to compare the real performances with their theoretical complexities. The system that we used for all the runs is equipped with an Intel i7-3517U 1.9/2.4 Ghz and 8GB of memory, while the IPA framework and the SIPM/EvIPM algorithms are written in Java 7. The values used for the parameters of the interaction functions, defined in section 3.2.3.1, are reported in table 3.

Table 3: IPA framework used values

dataset	$r_{\text{agent}}$	$d_{\text{search}}$	$\epsilon_{\text{move}}$	$\epsilon_{\text{dir}}$	$\epsilon_{\text{align}}$	$\epsilon_{\parallel}$	$\epsilon_{\text{dist}}$	$\epsilon_{\text{route}}$	$\epsilon_{\text{vel}}$	$\epsilon_{\text{lat}}$
NGSIM	1.83m	25m	0.1m	5°	2m	4°	0.1m	4°	0.15 km/h	3.25m
Campus	0.6m	10m	10 <sup>-4</sup> m	13°	1m	13°	0.01m	13°	0.005 m/s	1.2m

We want to do a comment on the choice of these values: as one can image, all the events resulting from the analysis of the IPA framework are very related to the choice of the interaction functions' parameters. Small variations in those parameters could present, in some cases, significant variations in the results. To tune them we have done several tests on both datasets, trying to validate their effectiveness and appropriateness on a sample of a group of events (for both datasets) and searching for a visual confirmation into the videos from which data are taken. Without doubt this part has required a lot of time, even for some problems linked to the Campus dataset which are explained in the section 6.2. After testing and analyzing the contexts in which the datasets are inserted, we think that those can be good values to represent appropriately the dataset conditions.

## 6.1 ANALYSIS ON NGSIM VEHICLE DATASET

The first dataset on which we have experimented our tools contains data about 15 minutes of car that are moving on 2100 feet of the US Highway 101<sup>1</sup>.

In figure 49 we can see a photo with the area considered, that is substantially a straight road with five lanes of the highway

<sup>1</sup> the original dataset is available here: <http://ngsim-community.org/>



Figure 49: Image of the part of highway considered by the NGSIM dataset.

with an entering and an exit ramp. The period considered in this dataset is described, from the preliminary analysis done by the owners of data, as a *transational period of traffic in the built up of a congestion period*<sup>2</sup>. The vehicles' trajectories have been tracked with an automatic analysis of the video recording of eight cameras, each of which was used to record a section of the highway, as evidenced in figure 49. These raw data was manual adjusted in a second moment, where needed. Each car moving on it has been tracked every one-tenth of a second.

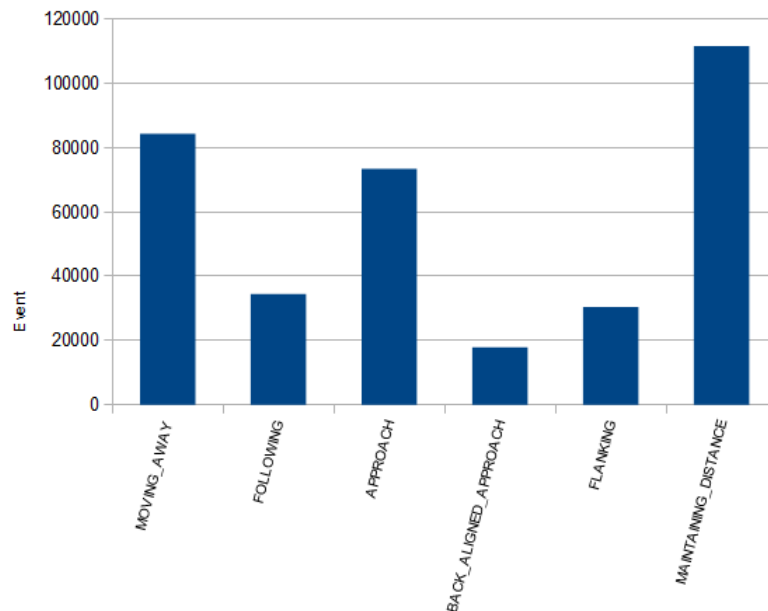


Figure 50: Events' distribution for the NGSIM dataset

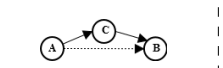

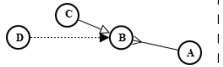

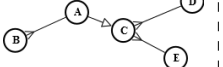
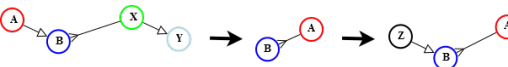
<sup>2</sup> The preliminar analysis on the area can be retrieved from <http://ngsim-community.org/>



With the analysis of the dataset with the IPA framework for the NGSIM dataset we have found 343441 different events of length at least one-tenth of second, divided as shown in figure 50. How we can expect, the dominating events between agents are the maintaining\_distance, the moving\_away and the approach ones. However, we have also found a significant presence following events, flanking and back aligned approaches. It is not surprising that opposite's type of events, as the opposite approach, were not found since we consider in this dataset vehicles that are moving on a straight road with the same mandatory direction.

Once retrieved these events, we have set up the SIPM algorithm for finding all the interaction patterns with instance patterns of minimum duration  $t_{min} = 20$  tenth of seconds and temporal support of  $t_{supp} = 0.8$ . The algorithm has retrieved 98 interaction patterns and did not found any interaction pattern with more than six agents per pattern: of these ones, we report some examples in the table 4 in the first column.

Table 4: Examples of interaction patterns and evolving patterns found for the NGSIM dataset: the symbols for the events are specified in table 2

INTERACTION PATTERNS	EVOLVING INTERACTION PATTERNS
	
	
	

We can see in this table how some patterns are intuitively recognizable, like the one in the grid in the first column at the first row, in which an agent A is following an agent B and at the same time a third agent C is in the middle maintaining the same distance from both A and B: this pattern represents a group moving together. However not all the patterns can be so simple to recognize with a visual analysis, as the one that state that while two agents D and E are moving away from an agent C, it is approached by an agent A which is moving away from a fifth agent B: the graphical representation of this pattern is given by the figure in the table 4, in the last row, first column.

Once retrieved the interaction patterns we have used the results of the algorithm to test the EvIPM algorithm, to find into

the interaction patterns possible schemes of evolution of the corresponding instance patterns. The algorithm has been configured to find all the sequences of pattern instances into windows of length  $t_{win} = 150$  tenth of seconds, with Jaccard's coefficient value of at least  $min_{jc} = 0.6$  and temporal support of  $t_{e-supp} = 0.9$ : furthermore we have limited our search to the sequences of instances of at most 4 consecutive instances. With these parameters the algorithm has found 950352 sequences divided in 786 frequent evolving interaction patterns. The most frequent evolving pattern is shown in the first row in table 4 in the second column; this sequence has been seen over 6079 times and indicates an overtaking in which the flanking phase is under the 2 second necessary to recognize a flanking between the approach phase and the moving away phase. In fact, even if is not reported into the examples, also the sequence  $approach(A, B) \rightarrow flanking(A, B) \rightarrow moving\_away(A, B)$  is a frequent pattern, but with a lower counting respect to the considered in the example: this because it is more simpler to find a fast overtaking in this context respect to one in which all the phases are slower.

Another example of frequent evolving pattern found is the evolution of a small group that is moving together, as shown in the second row, second column in table 4.

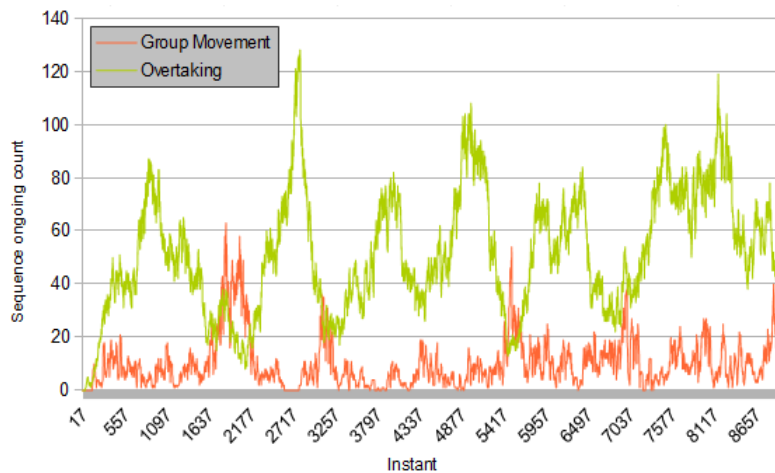


Figure 51: Ongoing events count for the fast overtaking and the group's movement along time scope T.

We take as example those two sequences to evidence some analysis that could be done with the retrieved data. We specify that in the following figures related to NGSIM, the flow of the vehicles is left to right, where the ramps are situated in the upper part of the plots while instead the external lane is in the bottom part. First of all we can analyze the time distribution of the sequences, considering the ongoing sequences' instances for each instant. From the plot in figure 51 we can note, for

instance, that when is increasing the group movement the fast overtaking tend to decrease and vice versa, that is a reasonable fact. Thus, we could see from the time distribution of the ongoing sequences if can emerge some hidden temporal correlations between sequences.

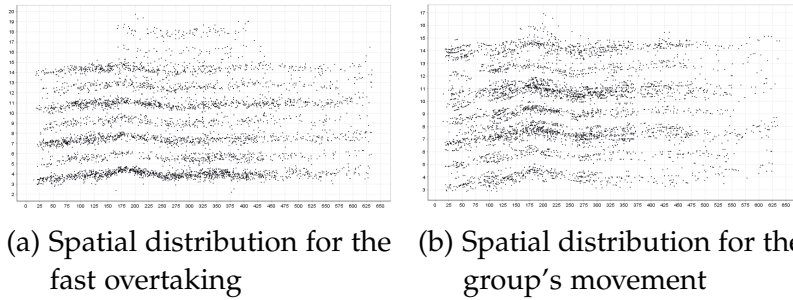


Figure 52: Spatial distribution for the sequences taken as example.

Another interesting analysis that we can do is the one related to the spatial distribution of the sequences: in fact, having the information about the trajectories of the agents, we can track each point in which a sequence has been recognized. To do that we have taken the middle point of the bounding box containing all the agents involved in each sequence of instance pattern. In this way we can produce a graphical view of the points in which the sequences has been found, as we can see in figure 52. Seeing the distributions we can immediately recognize some areas in which the sequences are most concentrated. However, even if we will see that this kind of plot can be significant in some cases, in this context we can have more significant information using these points with a kernel density estimation. In this way we can have an immediate visualization of the characterization of a sequence respect to the area studied. In figure 53 we report the computed density for the sequences we considered.

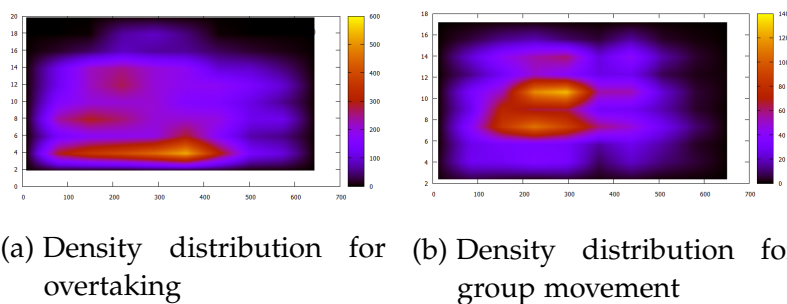


Figure 53: Density distribution for the sequences taken as example.

In figure 53a we can see how the fast overtaking is an event that characterizes the external lane of the highway, and the

most dense area is curiously the one that, in air line, is after the entrance ramp. The group movement is instead most concentrated into the central lanes of the road, in particular in proximity of the entering ramp, going gradually disappearing towards the end of the section considered, indicating as motorists near the entrance ramp maintain safer behaviour, maybe suggested by the flow of vehicles entering the road. In fact, this interpretation is supported by the fact that in the external lane this behaviour is less frequent.

Until now we have seen two behaviours that one can expect to find on a road. The last example we show, in the last row of the second column in table 4, is the one that we call for simplicity *complex overtaking*. If we analyze it, we can see that it has the principal components of the already considered fast overtaking even if in this case we have different interactions. While an agent A is approaching an agent B both are moved away by other two agents in the same condition; then the agent A, once made the overtaking on B, is moving away from it and, finally, when A is moving away from B this last one is approached by another agent. This is without doubt a more difficult behaviour to recognize even with a prior knowledge that one can have on the mobility context, however this behaviour emerge from data and has been seen, with different groups of agents, for at least 13 minutes. Then it must be something that could characterize the area in the period of time considered.

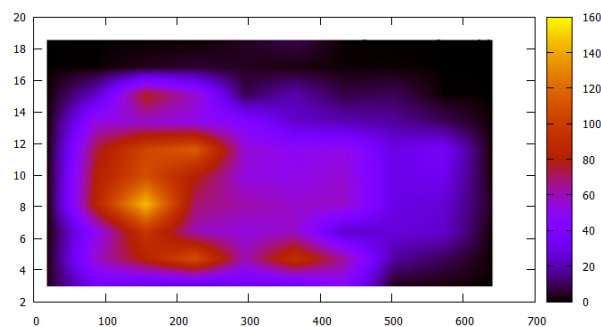


Figure 54: Density distribution for the complex overtaking. We can note how this event has a significance density also in proximity of the input junction.

In figure 54 we can see also how the distribution is very dissimilar from the one found for the fast overtaking, identifying its own different spatial characterization of the road.

## 6.2 ANALYSIS ON A CAMPUS SQUARE DATASET

The second dataset represent a set of people moving in a small square of an university campus <sup>3</sup>.



Figure 55: Image of the square considered into the Campus dataset.

In figure 55 we show a frame taken from the original video from which has been retrieved the data. The main reason for which we used this dataset is primarily for testing the framework and the algorithms in an environment very different from the previous one, and where we have a different type of moving agents, but using the same functions and events used for the NGSIM dataset; this is useful also to test also the ductility of the instruments in different contexts. This dataset is however more gaunt than the one downloaded from NGSIM: in fact contains data only for 3 minutes and 31 seconds.

A problem given by this dataset is that the agents and their trajectories weren't presented in the same format as the one given by the NGSIM dataset. In particular, for each agent, in this dataset has been provided a set of tracking point with associated the instants in which they have been tracked: furthermore the point has been tracked respect to an angle of the camera, so were not presented in any standard unit of measurement for distance. Thus we have tried to reconstruct the trajectories and to convert the points to be amenable to a standard unit of measurement in an empirical way. This procedure, even if we have made our best to be much precise as possible, has surely introduced some errors: however, for our testing purposes, this is an acceptable compromise. We have not chosen another dataset for the difficulty to find one that had been available freely.

After the preprocessing phase we have reconstructed the trajec-

<sup>3</sup> the original campus dataset is available here: <https://graphics.cs.ucy.ac.cy/research/downloads/crowd-data> in the section University Student

tories to have data relating to the agents with a sampling rate  $\frac{1}{25}$  of a second.

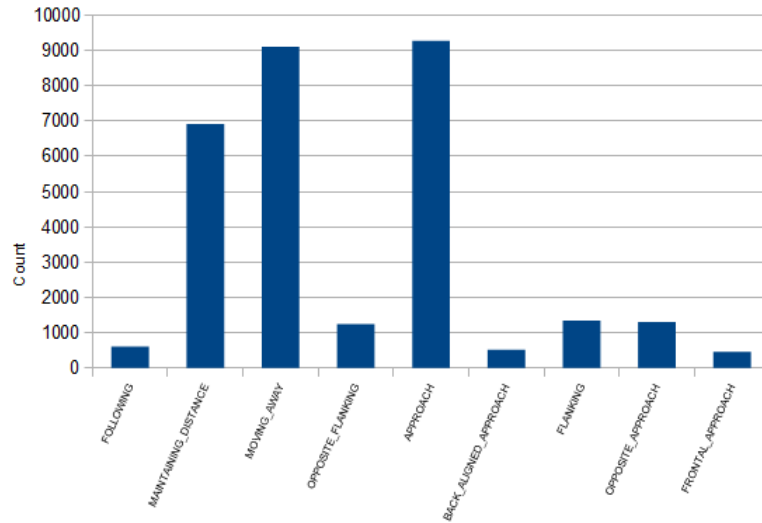
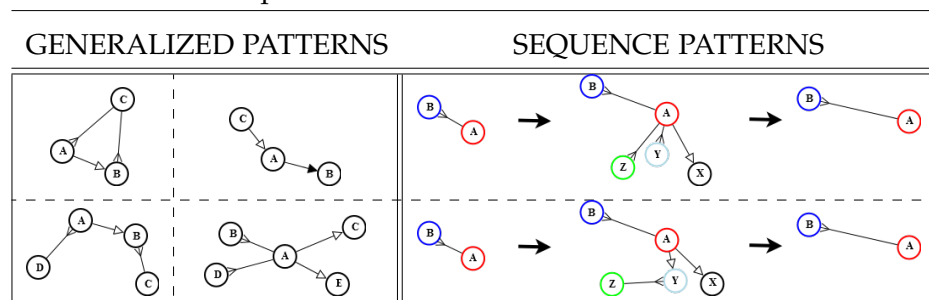


Figure 56: Events' distribution for the Campus dataset

The analysis with the IPA framework has retrieved 30772 events between pairs of agents, divided as shown in figure 56: even in this case, as the one presented in the previous section, we have that the events most present are the ones related to the maintaining of the distance, the approaching and the moving away. In this context they are also predominating on the others that has a lower counting; however we can note how here are present events that involve opposite agents, that in the previous dataset we could not find. After we have launched the SIPM algorithm with parameters  $t_{\min} = 25$  to identify all the interaction patterns with duration of at least a second and with time support of  $t_{\text{supp}} = 0.8$  of the dataset. The SIPM algorithm has returned 48 interaction patterns and did not find any interaction pattern with more than five agents. Some examples of these patterns retrieved are shown in the first column of table 5:

Table 5: Examples of interaction patterns and evolving interaction patterns found for the Campus dataset: the symbols for the events are specified in table 2



It is evident that, in only three minutes of video, the most frequent patterns involve the approach and the moving away as most frequent interaction patterns with instances of at least one second. This could be intuitive from the fact that a low number of other events respect to the maintaining of distance, the approaching and the moving away cannot be sufficient to find them in some frequent interaction patterns. Even for this dataset we have used the SIPM results as input for the EvIPM algorithm, this time with parameters  $t_{win} = 12$  seconds, Jaccard's coefficient value of at least  $\min_{j_c} = 0.6$  and temporal support of  $t_{e-supp} = 0.9$ : again we have limited the search of the sequences to only sequences of length at most 4 instances. The algorithm with these parameters has retrieved 147839 sequences grouped into 171 evolving interaction pattern. The sequences found for this dataset may be less obvious than the ones found into the NGSIM dataset; however, even if these events can't be named with a well-known macro-event (as the overtaking), they express plausible interactions' schemes into a place in which moving people are involved. An example of sequence found, of which we report the density distribution in figure 57, is the second in the table 5 in the column of evolving interaction patterns for the Campus dataset.

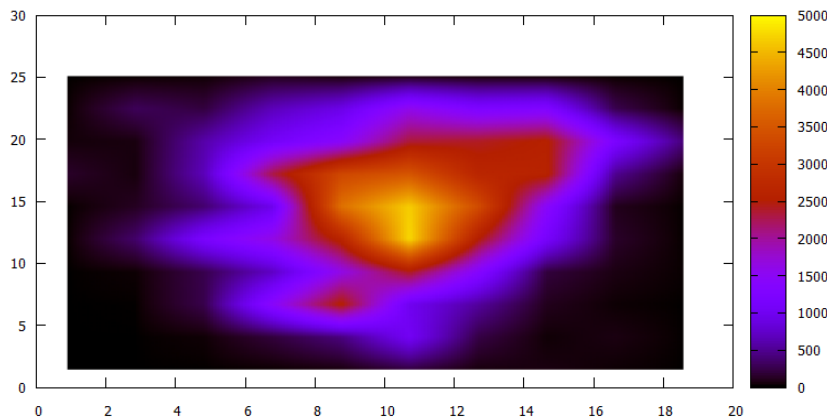
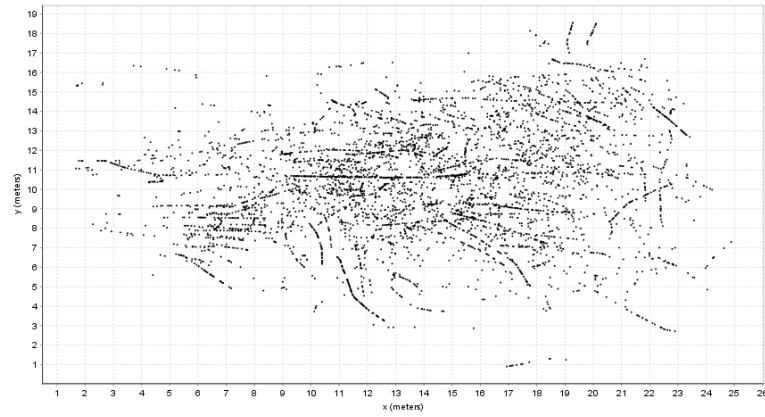


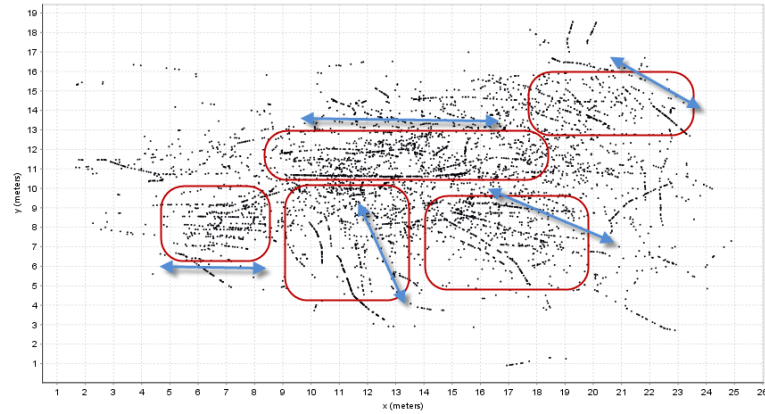
Figure 57: Density distribution of the evolving interaction pattern in the first row, second column of table 5

How we can see from the density distribution in figure 57, and how one can expect, the most dense area is the center of the square. For this sequence we also provide, in figure 58, the tracked points in which the sequence has been recognized.

We show the tracked point also because we can note something interesting that in the other dataset did not emerge as here. In fact, in figure 58, we can clearly see well-defined trajectories deriving from the tracked sequences: in figure 58b we have highlighted some groups of those derived trajectories to



(a) Spatial distribution of the sequence.



(b) Schematic ways in which develops the sequence.

Figure 58: In the figures we highlight some evident flows of movement for the evolving interaction pattern taking as example.

evidence how we could find information regarding the flow behaviour given by the agents that are involved in this sequence. This can be used to find not only the area associated to some behaviour, but also to inspect how they evolve in a spatial manner.

### 6.3 SIPM AND EVIPM PERFORMANCES

In this section we report the performances test for the SIPM and the SeqIPM algorithms. In particular, in the light of the complexities' theorems stated for both algorithms, we have tested the effectiveness of the heuristic that we have proposed to face the graph isomorphism in our context and of the pruning techniques used to find only frequent interaction patterns and sequences. In fact in the worst case both algorithm have a complexity that would make them unusable for a non toy-example. However, how happens for other algorithms theoretically unusable, we have seen how those algorithms provide results in a



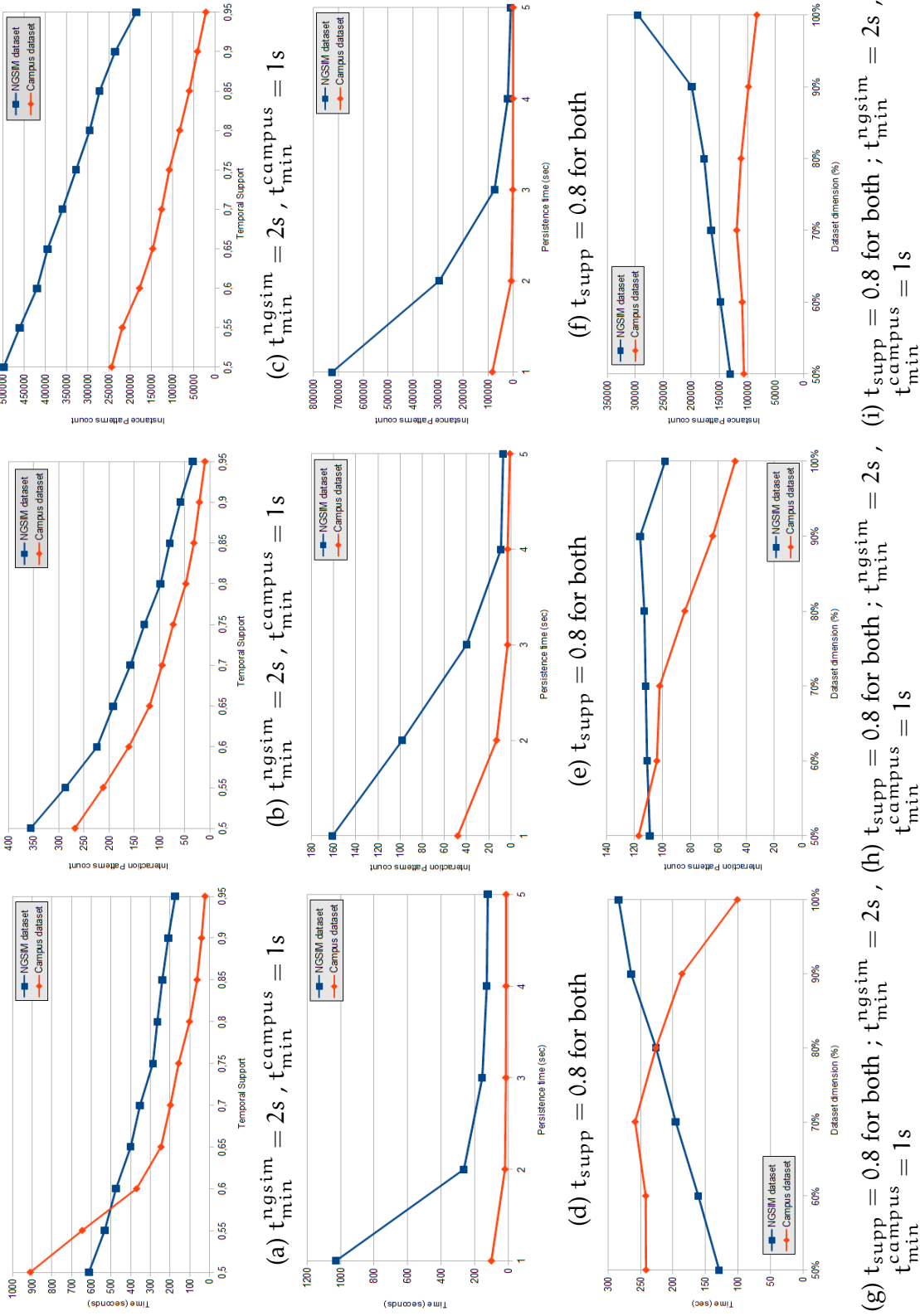


Figure 59: SIPM performances varying time support, persistence time and dimension of dataset. In the first column we have the completion time, in the second the count of generalized pattern and in the last column the count of all instance patterns found.

reasonable amount of time.

In figure 59 we report the charts with the performance tests for the algorithm SIPM. Those charts are related to the completion time, the number of interaction patterns and the number of instance patterns found varying the persistence time, the temporal support and the size of the dataset. In the first two rows we can see the performances of the SIPM having more stringent time parameters: in the first case, varying the persistence time of a group of events that happen contemporary, we can see how is difficult to find persistent instance patterns with the increase of this parameter. This fact as a consequence tell us that an higher persistence time, apart to be useful for finding interesting instances instead of the volatile ones, has an important impact on the growing of instances candidates. Thus it is a first passage of pruning, followed by the one done with the temporal support. The performances related to the variation of this last parameter are shown in the second row: again we see how is important the reduction of instances and interaction patterns found with the increase of this parameter. These consideration, keeping apart the fact that the given heuristic to face the isomorphism problem seems to work in real cases, highlights the importance of the choice of time persistence and temporal support respect to the problem faced, since it can determine a significant reduction or growing of the patterns found, that can be led to include some useless pattern or to exclude some useful one. In the third row we show the performances varying the dimension of the datasets. Here we can see a substantial difference between the dataset, in which is evidenced how the worst quality of the Campus dataset influences the final results. In fact, while the amount of interaction patterns are maintained at the same level with the NGSIM dataset and the instances increase with the increase of the dataset's dimension, this does not happen for the Campus dataset where the increase of the dataset' dimension corresponds to a decrease of patterns found: this can be explained with the fact that increasing the dimension of the dataset (and then the amount of time considered) the first dataset consolidates the frequent pattern found, while in the second case the ones that in a smaller interval was considered frequent are not really frequent (characteristics of the referenced context), and so disappear.

In the figure 60 we show the result of the tests done for the algorithm EvIPM on a percentage of the dataset of the 10% for both cases. In the first row we are varying the Jaccard's coefficient value, and we can see that in the campus dataset this

impact on all the values we are considering (completion time, sequence instances found, evolving interaction patterns found), while in the NGSIM dataset this does not lead a significant decreasing of the calculated values. Effects which instead are visible on both dataset varying the temporal support and the length of the searching window. Here again we obtain results in a reasonable amount of time, but is important to note how the pruning based on the temporal support for the evolving interaction patterns is less influent on the performances respect to the same effect that the similar parameter has on the SIPM algorithm: in this case in fact, even using strict parameters as only the 10% of the dataset and limiting the search of the parameters to 4 interaction patterns per sequence at most, we can see how the completion time is in the order of the ten of minutes. This because we can only act on the surely infrequent sequences, but for those ones found as frequent, we can only proceed to search all the combination that we can find into the defined window length with all the instance patterns that can make grow the sequence.

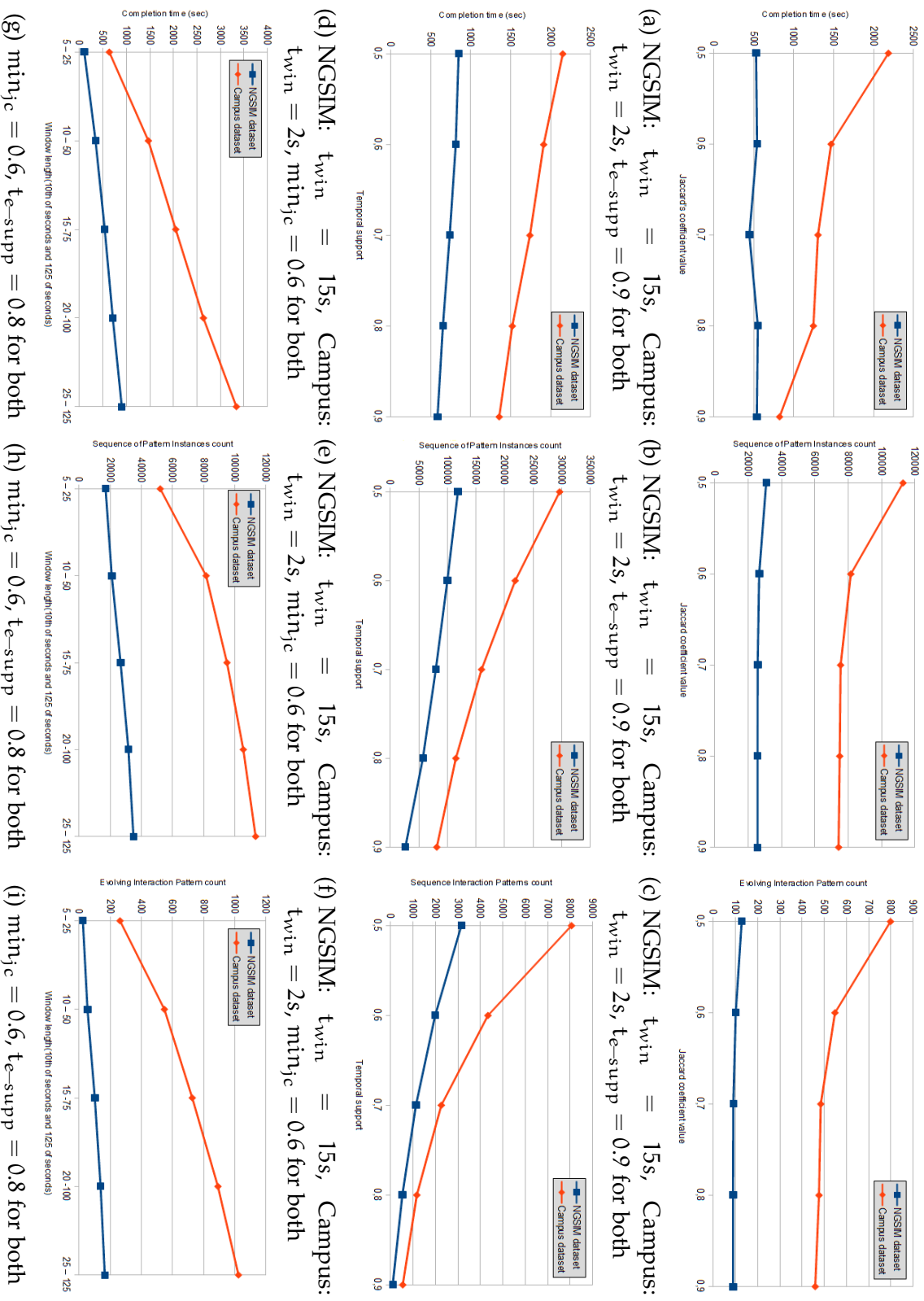


Figure 60: EVIPM performances varying time support, Jaccard values and window length. In the first column we have the completion time, in the second the count of frequent sequence group found and in the last all the sequences found that belong to an evolving pattern. All the test has been done on the 10% of the dataset of both, with at most 4 instances per sequence.

## Part III

### CONCLUSIONS AND FUTURE WORK

We have introduced a kind of mobile data analysis based on the idea that data could tell something, bringing out patterns that can describe the behaviour of the moving agents bringing out those patterns from the interactions between agents. In this last part we conclude our work, summarizing our results respect to the initial idea and possibles improvements and future work following this new approach.



## CONCLUSIONS

---

### 7.1 FUTURE WORKS

The work we have done till now cannot be considered conclusive: a lot of work and improvements can be done to strengthen the new methodology we introduced and to integrate it with the existent literature. As future works that can be done to improve what we started with this thesis, we suggest the following:

**NEW EXPERIMENTS** The datasets analyzed in this thesis are only two examples that we have used to define the problem and study a data mining approach for the extraction of patterns based on the interactions between agents. However, we could think to analyze other situations involving mobile agents, as areas with dense crowd, malls or other contexts involving vehicles. In this sense the NGSIM community provides other datasets, with complex environments as streets with crossroads or wider urban areas. We could also think, if we should plan an experiment, to record the moving agents of an area for different days to check if the patterns found present the same temporal and spatial trends, or be used as a starting point to see in which conditions an interaction pattern (or a sequence of interaction patterns) is frequent and with what distribution over time and space.

**INTEGRATE OBJECTS OF THE SPECIFIED CONTEXT** In the presented work we have left apart the role of the environment and its possible passive influence on the active agents; we do that because in the presented datasets we preferred to focus our attention on basic interactions, i.e. those between active agents. However, we could define some interactions, in the NGSIM dataset, for example regarding the input/output ramp; we should not underestimate the influence that an object can passively have on the active agents, as a velocity limit signboard or the weather.

INTEGRATE THE EXISTENT WORKS IN SIMULATION In section 2.1 we have seen the equation for the social force model applied to the simulation of a person's movement:

$$m_i \frac{d\mathbf{v}_i}{dt} = m_i \frac{v_i^0(t) \mathbf{e}_i^0 - \mathbf{v}_i(t)}{\tau_i} + \sum_{j(\neq i)} \mathbf{f}_{ij} + \sum_W \mathbf{f}_{iW} \quad (25)$$

As already said, we based our work in some way on assumptions that are reported in this equation: the resulting movement of an agent can be described as a sum of influences between the pairs constituted by the current agent and the ones interacting with it; in that case those values are given by a set of functions that model social interactions as physical forces, but we could think to use the information that we can retrieve from the IPA framework, in some way, to improve this equation, since we retrieve information about all the events that involve pairs of agents. In this way we could insert a data-driven social component that can improve the realism of the simulation of moving agents.

OPEN THE STUDY OF INTERACTIONS IN OTHER CONTEXTS The work we have done, and the problem we have defined, is used for a mobility context; however there is no limitation to the possible application of this methodology in different context, adjusting some concepts as the notion of neighborhood or the specific events that need to be used. Since we stated that we continuously interact with the external world, we could apply the mining of interaction patterns in any context where there exist a set of measurable interactions between agents: in economy, where financial markets influence each other; in medicine, where patients and drugs in a certain sense interact and influence each other; in games (either on computers or in the physical world), where players interact with each others and so on;

Then it becomes fundamental to understand, on one side, the potential benefits that the interaction patterns can give in other fields beyond mobility; on the other hand, we must understand which is the best way to integrate the information about frequent behaviours between agents discovered with our work and how they can be inserted into the existent tools to improve them.



## 7.2 SUMMARY OF RESULTS

We started this work with a question: can hidden interactions between agents tell something important or difficult to see that we might overlook when analyzing data? Through our work, we aimed at letting data “speak about themselves”: the main goal we have selected since we started, was to find a new approach that could exploit the common behaviours and their characteristics analyzing the most common activities in our life: the interaction with the others. To do that, we have defined a set of tools and mechanisms to explicit those interactions, in function of visible events of which interactions are the atomic components.

The experiments we made on the datasets have been very encouraging: on one side the algorithms we developed seemed to react well, in terms of computation time, on real cases with big amounts of data; on the other hand, the results obtained are very significant. For instance, think about the results obtained on the NGSIM dataset; we have found patterns that one initially can expect as frequent, as the overtaking or the group movements; however, this fact in a certain sense validates the results found, because an instrument that cannot find the obvious cannot surely hope to find what is not obvious or immediate. Thus we can state that less obvious results we found can really highlight those hidden behaviours we were searching.

In conclusion, we believe that the approach we proposed for the extraction of frequent patterns from mobility data in mobile contexts is a good start to a new way of exploiting the hidden information contained in the data: even if we have only posed the basis of the approach we think that our work, with deeper studies, could have a very useful impact on the scientists/analysts/tools that work on simulation, and on the studies of big datasets from an analytical and/or sociological point of view.



## BIBLIOGRAPHY

---

- [1] M. Berlingerio, M. Coscia, F. Giannotti, A. Monreale, D. Pedreschi "Foundations of multidimensional network analysis", International Conference on Advances in Social Networks Analysis and Mining (ASONAM), pp. 485-489, 2011
- [2] D. Helbing, P. Molnar, "Social Force Model for Pedestrian Dynamics", Physical Review E, Volume 51, Issue 5, pp. 4282-4287, 1995
- [3] D. Helbing, I. Farkas, T. Vicsek, "Simulating dynamical features of escape panic", Nature, Volume 407, pp. 487-490, 2000
- [4] T. I. Lakoba, N. M. Finkelstein, "Modifications of the Helbing-Molnar-Farkas-Vicsek social force model for pedestrian evolution", Simulation: Transactions of the Society for Modelling and Simulation International, Volume 81, Issue 5, pp. 339-352, 2005
- [5] R. L. Hughes, "A continuum theory of flow pedestrian motion", Transportation Research, Volume 36, Part B, pp. 507-535, 2002
- [6] P. M. Torrens, "Moving agent pedestrians through space and time", Annals of the Association of American Geographers, Volume 102, Issue 1, pp. 35-66, 2012
- [7] E. Bonabeau, "Agent-based modeling: Methods and techniques for simulating human systems", Proceedings of the National Academy of Sciences of the United States of America (National Academy of Sciences), Volume 99, 2002
- [8] A. V. Moudon, P. M. Hess, M. C. Snyder, K. Stanilov, "Effects of site design on pedestrian travel in mixed-use, medium-density environments", Transportation Research, Record 1578, pp. 48-55, 1997
- [9] T. Hägerstrand, "Space-time and human conditions", Dynamic allocation of urban space, ed. Karlqvist-Lundkvist-Snickars, pp. 3-12, 1975
- [10] M. Moussaid, D. Helbing, S. Garnier, A. Johansson, M. Combe, G. Theraulaz, "Experimental study of the behavioural mechanism underlying self-organization in human crowds", Proceedings of the Royal Society B: Biological Sciences, Volume 276, pp. 2755-2762, 2009

- [11] P. Hidas, "Modelling vehicle interactions in microscopic simulation of merging and weaving", *Transportation Research Part C: Emerging Technologies*, Volume 13, No. 1, pp. 37-62, 2005
- [12] W. Quattrociocchi, D. Latorre, E. Lodi, M. Nanni, "Dealing with interaction for complex system modelling and prediction", *International Journal of Artificial Life Research*, Volume 1 No.1, pp.1-11, 2010
- [13] C. Castelfranchi, "The theory of social functions: challenges for computational social science and multi-agent learning", *Journal of Cognitive Systems Research*, Number 2, pp.5-38, 2001
- [14] J. W. Raymond, P. Willett, "Maximum common subgraph isomorphism algorithms for the matching of chemical structures", *Journal of Computer-Aided Molecular Design*, Volume 16, pp.521-533, 2002
- [15] N.Eagle, A. Pentland, "Reality mining: sensing complex social systems", *Personal and Ubiquitous Computing*, Volume 10, Issue 4, pp 255-268, 2006
- [16] F. Giannotti, D. Pedreschi, "Mobility, Data Mining and Privacy: A Vision of Convergence", ed. Springer Berlin Heidelberg, 2008
- [17] M. Nanni, B. Kuijpers, C. Körner, M. May, D. Pedreschi, "Spatio-temporal Data Mining", in *Mobility, Data Mining and Privacy*, ed. Springer Berlin Heidelberg, Chapter 10, pp 267-296, 2008
- [18] P.N. Tan, M. Steinbach, V. Kumar, "Introduction to data mining", Chapter 6, ed. Pearson, 2005
- [19] R. Agrawal, T. Imielinski, A. Swami, "Mining association rules between sets of items in large databases", *Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, pp. 207-216, 1993
- [20] R. Agrawal, R. Srikant, "Fast algorithms for mining association rules", *Proceedings of the 20th International Conference on Very Large Databases*, pp. 487-499 , 1994
- [21] D. J. Cook, L. B. Holder, "Mining graph data", ed. John Wiley and Sons, 2006
- [22] A. Inokuchi, T. Washio, H. Motoda, "An apriori-based algorithm for mining frequent substructures from graph

- data", Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery, pp. 13-23, 2000
- [23] X. Yan, J. Han, "gSpan: graph-based substructure pattern mining", Proceedings of the 2002 IEEE International Conference on Data Mining, pp. 721-725, 2002
- [24] O. Arıkan, D. A. Forsyth, "Interactive motion generation from examples", Proceedings of the 29th annual conference on Computer graphics and interactive techniques, pp. 483-490, 2002
- [25] K. H. Lee, M. G. Choi, Q. Hong, J. Lee, "Group behaviour from video: a data-driven approach to crowd simulation", Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation, pp. 109-118, 2007
- [26] J. Dijkstra, J. Jessurun, H. Timmermans, "A cellular automata model for pedestrian behaviour", in *Pedestrian and evacuation dynamics*, ed. Springer-Verlag, pp. 173-181, 2001
- [27] D. Helbing, "A fluid-dynamic model for the movement of pedestrians", *Complex Systems*, Volume 6, pp. 391-415, 1992
- [28] M. de Berg, O. Cheong, M. van Kreveld, M. Overmars, "Visibility graphs - Finding the shortest route", in *Computational Geometry: algorithms and applications*, ed. Springer, Chapter 15, pp. 323-333, 2008
- [29] S. K. Ghosh, "Visibility algorithms in the plane", ed. Cambridge University Press, 2007
- [30] A. Sebastian, M. Tang, Y. Feng, M. Looi, "Multi-vehicles interaction graph model for cooperative collision warning system", Proceeding of the IEEE Intelligent Vehicles Symposium, pp.929-34, 2009
- [31] C. Oh, T. Kim, "Estimation of rear-end crash potential using vehicle trajectory data", *Accident Analysis and Prevention*, ed. Elsevier, Volume 42, Issue 6, pp.1888-1893, 2010
- [32] J. Torán, "On the hardness of graph isomorphism", *SIAM Journal on Computing*, Volume 33 Issue 5, pp.1093-1108, 2004
- [33] V. Arvind, J. Köbler "Graph isomorphism is low for ZPP(NP) and other lowness results", Proceedings of the 17th Annual Symposium on Theoretical Aspects of Computer Science, pp.431-442, 2000

- [34] DIVA project website, <http://www-kdd.isti.cnr.it/project/diva>
- [35] P. Holme, J Saramäki "Temporal networks", Physics Reports, Volume 519, Issue 3, pp.97-125, 2012
- [36] V. Kostakos "Temporal Graph", Journal of Physics A, Number 388, pp.1007-1023, 2009