

**UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA**

Areli Andreia dos Santos

**TOWARDS MOVING OBJECTS RELATIONSHIP
INFERENCE FROM ENCOUNTER PATTERNS**

Florianópolis

2016

Areli Andreia dos Santos

**TOWARDS MOVING OBJECTS RELATIONSHIP
INFERENCE FROM ENCOUNTER PATTERNS**

Dissertação submetida ao Programa
de Pós-Graduação em Ciência da Com-
putação para a obtenção do Grau de
Mestre.

Orientador

Universidade Federal de Santa Cata-
rina: Prof. Dr. Vania Bogorny

Florianópolis

2016

Ficha de identificação da obra elaborada pelo autor através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Santos, Areli Andreia dos

Towards Moving Objects Relationship Inference from Encounter Patterns / Areli Andreia dos Santos ; orientadora, Vania Bogorny - Florianópolis, SC, 2016. 60 p.

Dissertação (mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico. Programa de Pós-Graduação em Ciência da Computação.

Inclui referências

1. Ciência da Computação. 2. Relationship Degree. 3. Encounters. 4. Moving Object Trajectories. I. Bogorny, Vania. II. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Ciência da Computação. III. Título.

Areli Andreia dos Santos

**TOWARDS MOVING OBJECTS RELATIONSHIP
INFERENCE FROM ENCOUNTER PATTERNS**

Esta Dissertação foi julgada aprovada para a obtenção do Título de “Mestre”, e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

Florianópolis, 22 de março 2016.

Prof. Dr. Carina Friedrich Dorneles
Coordenador
Universidade Federal de Santa Catarina

Banca Examinadora:

Prof. Vania Bogorny, Dr.
Universidade Federal de Santa Catarina

Prof. Renato Fileto, Dr.
Universidade Federal de Santa Catarina

Prof. Ronaldo dos Santo Mello, Dr.
Universidade Federal de Santa Catarina

Prof. José Antonio Fernandes de Macêdo, Dr.
Universidade Federal do Ceará

I would like to dedicate this work to my
mother Andreia Edesia da Silva

ACKNOWLEDGEMENT

I would like to thank God for all the strength and patience that was given to me during this work. Thanks to my parents, that always supported me in my decisions. To my advisor Prof. Dr. Vania Bogorny for encouraging me and for being so careful in each detail of this work. Also to Prof. Dr. Luis Otavio Alvares that contributed in discussions, ideas, and that was a guide in this work. Thanks to my boyfriend André Salvaro Furtado for encouraging me to start this thesis and keep working, for all the ideas, support and patience with me.

I must thank CAPES to support the development of this work through the scholarship. To UFSC, specially PPGCC, for the amazing opportunity of being part of this institution during these years. To the Marie Curie project and the European Union which made it possible for me to have a great experience at UPRC (University of Piraeus Research Centre) in Piraeus/Greece through the SEEK project. This was a singular opportunity in my life, having the pleasure of work with great researchers. I would like to thank Prof. Dr. Nikos Pelekis, Prof. Dr. Yannis Theodoridis, Panagiotis Tampakis, Despina Kopanaki, and Marios Vodas, participants in the SEEK project, which have contributed in the work and have helped me a lot during my stay in Piraeus.

Finally, I would like to say thanks for the colleagues that helped in the discussions of this work: Carlos Andres Ferrero, Lucas André de Alencar and Marco Beber. To everyone from LISA lab: Fernando de Lucca Siqueira, Mateus Barragana, Cleto May, Geomar Schreiner, Douglas Klein, Juarez Piazza Sacenti. And all the others that helped collecting data, without them this work would not be possible.

If you can't give me poetry, can't you give
me poetical science?

(Ada Lovelace)

RESUMO

Dispositivos como smartphones e navegadores GPS são muito populares. Estes equipamentos podem armazenar a localização de um objeto associada ao respectivo tempo, gerando um novo tipo de dado, chamado de trajetórias de objetos móveis. Com esses dados é possível descobrir vários padrões interessantes, entre eles o padrão encontro, permitindo inferir o grau de relacionamento entre os objetos que se encontram. Este trabalho propõe um método para inferir o grau de relacionamento entre objetos através dos encontros das suas trajetórias. Primeiramente, os encontros são detectados pelo algoritmo *BeingTogether* com base em uma nova definição, que considera todo o tempo em que os objetos estão próximos no tempo e no espaço, independentemente se estão parados ou em movimento. A partir de medidas baseadas na duração, frequência e área dos encontros são propostos os algoritmos *MORE* e *MORE++*. Os algoritmos foram avaliados em experimentos com dados reais de trajetórias e mostram que os encontros e os respectivos relacionamentos são detectados corretamente.

Palavras-chave: Grau de Realacionamento, Padrões de Encontro, Trajetórias de Objetos Móveis, Encontros em Trajetórias

ABSTRACT

Devices like smart phones and GPS navigators are very popular nowadays. These equipments can save the location of an object with an associated time, generating a new kind of data, called trajectories of moving objects. With these data it is possible to discover several interesting patterns, among which is the interaction between individuals, allowing to infer their relationship. This work proposes a method to infer the relationship degree between moving objects from their encounters. First, the encounters are detected by the algorithm *BeingTogether* based on a new definition that considers the entire time that objects are close in time and space, regardless of whether they are stopped or moving. The relationship degree is calculated using measures based on the duration, frequency and area of encounters, through the algorithms *MORE* and *MORE++*. The algorithms were evaluated in experiments with real datasets, and show that the meetings and their relationships are correctly detected.

Keywords: Relationship Degree, Encounter Patterns, Moving Object Trajectories, Trajectory Encounters

LIST OF FIGURES

Figure 1	Temporal Representation of encounters between the same individuals during 3 days.....	22
Figure 2	Encounter Area.....	35
Figure 3	Different Areas.....	37
Figure 4	Third case of addOrMergeEncounter() function.....	42
Figure 5	(left) every encounter and (right) encounters grouped by objects.....	43
Figure 6	Trajectories and Encounters at UFSC.....	46
Figure 7	Amsterdam database color teams.....	48
Figure 8	Encounters at Amsterdam database considering $minTime = 15$ minutes.....	49
Figure 9	Encounters at Amsterdam database with $minTime = 30$ minutes.....	51
Figure 10	Number of Detected Encounters for different values of Δ_t	53
Figure 11	Number of Detected Encounters for different values of Δ_d	53
Figure 12	Computation Time in minutes over different values of Δ_t	54
Figure 13	Computation Time in minutes over different values of Δ_d	54
Figure 14	Computation Time in seconds over different values of Δ_t in the original dataset.....	54
Figure 15	Computation Time in seconds over different values of Δ_d in the original dataset.....	54
Figure 16	Temporal Representation of encounters between the same individuals.....	58

LIST OF TABLES

Table 1	Summary of Related Works.	31
Table 2	Relationship Measures for the running example	44
Table 3	Scheduled Encounters at UFSC.	46
Table 4	Encounters detected by <i>BeingTogether</i> at database UFSC	47
Table 5	Relationship Degrees between objects at UFSC	47
Table 6	Relationship Degrees between objects at Amsterdam database with <i>minTime</i> = 15 minutes	50
Table 7	Relationship Degrees between objects at Amsterdam database with <i>minTime</i> = 30 minutes	52
Table 8	Relationship Measures for the running example <i>MORE+</i> +	62
Table 9	Scheduled Encounters at UFSC during 3 days	63
Table 10	Relationship Degrees between objects at UFSC	64

SUMMARY

1	INTRODUCTION AND MOTIVATION	21
1.1	OBJECTIVE	23
1.2	METHODOLOGY AND STRUCTURE	23
2	RELATED WORK	25
2.1	ENCOUNTERS	25
2.2	RELATIONSHIP IN TRAJECTORIES OF MOVING OBJECTS	26
2.3	RELATIONSHIP INFERENCE FROM GSM AND SOCIAL NETWORKS	27
3	ALGORITHMS BEINGTOGETHER AND MORE	33
3.1	MAIN CONCEPTS	33
3.2	BEINGTOGETHER: AN ALGORITHM TO COMPUTE ENCOUNTER PATTERNS	38
3.3	MORE: AN ALGORITHM TO COMPUTE THE RELATIONSHIP AMONG OBJECTS	42
3.4	RUNNING EXAMPLE	44
4	EXPERIMENTAL EVALUATION	45
4.1	UFSC DATASET	45
4.2	AMSTERDAM DATASET	48
4.3	PARAMETER ANALYSIS FOR THE AMSTERDAM DATASET	52
4.4	DISCUSSION	55
5	MORE++	57
5.1	MAIN DEFINITIONS	57
5.2	MORE ++ ALGORITHM	60
5.3	RUNNING EXAMPLE	62
5.4	EXPERIMENTAL EVALUATION	63
6	CONCLUSION	67
	REFERENCES	69

1 INTRODUCTION AND MOTIVATION

The price reduction of mobile devices such as GPS and mobile phones, as well as advances in satellite and wireless sensor technologies, has significantly increased the use of these mechanisms. Mobile devices allow recording the movement of an individual. Accordingly, any individual who carries a mobile device, while moving, generates a trace, in which each time point corresponds to a location in space. This trace is called *trajectory of the moving object*. There are several works dealing with such data, as the one that describes avoidance of trajectories (ALVARES et al., 2011) (LETTICH et al., 2016), chasing (SIQUEIRA; BOGORNY, 2011), flocks, leadership, convergence, and encounter (LAUBE; KREVELD; IMFELD, 2005). A summary of different works on trajectory analysis is presented in (BRAZ; BOGORNY, 2012).

Although there are numerous works on patterns in trajectories, only a few address the encounter/meeting patterns, and even less infer friendship relationships from trajectories. The first work to deal with encounters was (LAUBE; KREVELD; IMFELD, 2005), defining encounters as a set of objects that have points inside a specific given radius at similar time. Gudmundsson, in (GUDMUNDSSON; KREVELD; SPECKMANN, 2007), defined the encounter pattern with a minimum number of entities inside a given minimum radius at the same time. Bak, in (BAK et al., 2012), proposed an algorithm to detect encounters between two trajectories, where all points that are close in space and time are connected forming a line. This work focuses on visual analysis of encounters. The most formal definition of encounter is given by (DODGE; WEIBEL; LAUTENSCHÜTZ, 2008), which defines encounter as a convergence where the objects arrive at a place at the same time.

Existing works only define the concept of encounter, and have neither go deeper in the encounter pattern analysis nor use them for relationship inference among moving objects. The inference of relationships is an important issue for several application domains. In biology, for example, we can discover how much time the pandas A and B stayed together in the last summer, and which areas they visited together. For investigative applications, one may analyze the total amount of time that a group of individuals stayed close in the last month, how much time objects Y and Z of this group stayed together, and which areas they visited with a larger group of objects.

We strongly believe that the relationship of objects is directly related to the amount of time they spend together, or in other words,

how frequently they meet. For instance, a married couple may stay more time together than a couple that is dating, or a couple of friends.

To infer relationships from encounter patterns is not a trivial task. Let us consider the example shown in Figure 1, where Louis met Marie and then both met John, and afterwards they met Susan, at day 5th of May. At this day, there are four encounters, each one with duration of one hour. The first encounter is between Louis and Marie (from 9 to 10). The second encounter is between Louis, Marie and John (from 10 to 11). The third encounter is between Louis and Marie alone (from 11 to 12), and the fourth encounter is between Louis, Marie and Susan (from 12 to 13).

The example of Figure 1 illustrates three important things when we reason about measuring the relationship degree based on encounters: the number of objects at each encounter, the frequency of the encounters, and their duration. In this example, still considering 5th of May, the duration of the encounter is higher for Louis and Marie, corresponding to 4 hours in total. So we can consider that Louis and Marie have a stronger relationship degree among each other than with John and Susan. When considering the encounter of Louis and Marie alone, their encounter has 2 hours of duration. Considering the whole time they stayed together at 5th of May the duration is 4 hours. This example shows that it is very complex to analyze encounters and relationships between moving objects, and that we must analyze every different encounter and with different number of objects.

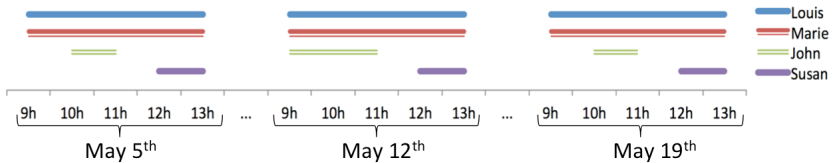


Figure 1 – Temporal Representation of encounters between the same individuals during 3 days

In this thesis we propose a new algorithm to compute level-wise encounter patterns from trajectories, called *beingTogether*. From these encounters we propose two algorithms to infer relationships, called *MORE* (*Moving Objects Relationship degree based on Encounters*) and *MORE++*. The main contribution of our approach is the computation of level-wise encounter patterns and the inference of moving object relationships. In this work we go one step further to existing approaches which do only detect encounters from trajectory data and

do not infer objects relationships, making the following contributions: (i) define encounter patterns between level-wise objects as well as the encounter area from trajectory data; (ii) define different criteria to measure the relationship of individuals based on their encounters; (iii) propose efficient algorithms, one to compute encounters and, two to compute the relationship degree of multiple individuals.

1.1 OBJECTIVE

The main objective of this work is to infer the relationship degree between moving objects considering their encounter patterns. To achieve this goal, the following specific objectives must be fulfilled:

- Formally define encounter and encounter area;
- Formally define the relationship degree between moving objects, and define different measures to infer the relationship degree;
- Propose efficient algorithms to compute encounters and infer the relationship degree of individuals from their trajectories.
- Evaluate the proposed methods with real trajectory data.

1.2 METHODOLOGY AND STRUCTURE

The following tasks will be performed to achieve the objectives of this thesis:

1. Review of the state-of-the-art on encounter detection and relationship degree inference over different types of data, such as: phone calls, bluetooth records, social networks, and GPS trajectory data;
2. Formally define encounter, encounter area, and relationship degree based on encounters;
3. Define and implement an algorithm to detect encounters between a group of objects and an algorithm to infer the relationship degree of groups of objects;
4. Generate datasets with ground-truth to evaluate the algorithms.

5. Perform experiments using a running example, the generated dataset, and a real trajectory database (RAESSENS, 2007).

The remaining of this document is organized as follows: Chapter 2 summarizes the related work. In Chapter 3 we present the main definitions and the algorithms *BeingTogether* and *MORE*. In Chapter 4 we present the experiments to evaluate the algorithms. In Chapter 5 we propose a second algorithm that overcomes some limitations of the *MORE* algorithm and, finally, in Chapter 6 we conclude the thesis.

2 RELATED WORK

In this chapter we present related works that define encounters and flocks, and since there are only a few works that infer relationships based on GPS trajectories, we present some works that infer relationships from other types of spatio-temporal data as phone logs/calls and social networks.

2.1 ENCOUNTERS

Among the few existing studies in the literature that directly address the encounter between trajectories we can highlight the work of (LAUBE; KREVELD; IMFELD, 2005), which defines a set of patterns considering both geographic and temporal aspects. Between the different patterns are flocks and encounters. The flock pattern can be detected from the REMO (Relative Motion Pattern) matrix, which relates the time and the direction of objects movements. When the objects are moving in the same direction at the same time, and within a certain area, they present a flock behavior. The encounter pattern cannot be detected from the REMO matrix. To detect encounters, (LAUBE; KREVELD; IMFELD, 2005) proposed a division of the space in cells of a given size, and the trajectories that intersect the same cells in a similar time have an encounter pattern.

Gudmundsson in (GUDMUNDSSON; KREVELD; SPECKMANN, 2007) defines encounter as a minimal number of objects m that "stay" inside an area of size r at a certain time. A flock is defined as a set of m objects that "move" inside a cylinder of size r for a certain time, and objects may leave or enter the flock. In this work both encounter and flock are considered as different patterns, while in our proposal we assume that objects that are together, either stopped or moving, are having an encounter. Indeed, while in (GUDMUNDSSON; KREVELD; SPECKMANN, 2007) objects enter and leave a flock, we compute a *different* encounter every time the group changes.

A taxonomy of movement patterns was proposed by Dodge in (DODGE; WEIBEL; LAUTENSCHÜTZ, 2008). Different types of patterns that could be interesting for mining the interaction between objects is presented in this work, including *Meet* and *Movingcluster(orFlock)*. *Meet* is defined as a set of objects that stay in a cylinder of a given radius in space-time, they stay within a stationary disk of specific radius

in a certain time interval. A *movingcluster* occurs when a set of objects stay close to each other while taking the same path during a specific time. (BAK et al., 2012) proposed an algorithm to detect encounters focused on visual analysis. The main idea of the algorithm is to connect with a line two points of different objects that are close in space and time. The user can vary the space and time threshold, visualizing the connected dots and evaluating the results.

In our work, we consider as encounter the whole time that the objects remain together, does not matter if they are stopped or moving. Usually an encounter is split in two parts to fit in the definitions of encounter and flock. For example, a walk with a friend followed by a visit to a restaurant will be separated in two parts: the walk is a flock pattern and the staying in a restaurant, an encounter. Gudmundsson, in (GUDMUNDSSON; KREVELD; SPECKMANN, 2007), defines the encounter pattern with a minimum number of entities and a minimum radius, and a flock pattern has the same parameters of the encounter, but the objects must be moving in the same direction. These works, however, do neither provide any analysis of the detected encounters nor infer any relationship degree, as we propose in this thesis.

The works about encounters and flocks focused only on detecting the pattern, not on the inference of relationships between multiple objects, which is the objective of our work. Indeed, the area of the encounters and flocks is neither defined nor computed.

2.2 RELATIONSHIP IN TRAJECTORIES OF MOVING OBJECTS

Only a few works use GPS data to infer relationships, like (BRILHANTE et al., 2012) that infers relationships among places and (MA et al., 2014) among people. Because GPS data are more complex and lack in relationship information, both works ((BRILHANTE et al., 2012) and (MA et al., 2014)) use summarized trajectories, i.e., stops or stay points to reduce the complexity. In our approach we use raw trajectories, first computing encounters/flocks and from these encounters propose a method to infer and rank relationships.

It is important to mention that the inference of relationships either from phone calls or social media is quite trivial, since information of who is connected with who is available. In GPS traces, on the other hand, the data is a set of points that correspond to the position of an object in space and time.

Brilhante (BRILHANTE et al., 2012) proposed a methodology to

discover communities of interesting places, using as input the trajectories of moving objects and known POIs (Points of Interest). The first step is to detect stops at the given POIs. If a group of trajectories has short stops on a pair of POIs, these POIs are connected. This work does not infer relationships among users, only between points of interest.

Ma (MA et al., 2014) proposed a model to infer the strength of social relationships among pairs of objects in GPS trajectories. The input of the method are stay points (stops) of different objects, while we use the raw trajectories to infer relationships. Similarly to (PHAM; SHAHABI; LIU, 2013), it uses the entropy of places and users. The entropy of the user is higher when a user visits different places. The entropy of the places is based on the number of users that visit the place. Entropy is a measure of disorder, so, the higher the entropy is, the lower is the relationship strength. If an object visits a place that is visited by many other objects, the entropy of the place increases. In our proposal we do not consider the entropy of the place because we believe that although more populated places may increase the chance of objects being close by coincidence, it does not mean that they do not know each other. Therefore, we will consider in our proposal that the more different areas two or more objects have encounters, the higher is the probability to know each other, and this will increase the relationship degree. The result of the method is a matrix that relates each pair of objects, and because of the user entropy, the relationship from object A to object B can be different from object B to A.

None of the existing works infer a relationship degree based on encounters of groups of objects, and none of them consider area, duration and frequency as measures to determine the relationship degree, as we propose in this thesis.

2.3 RELATIONSHIP INFERENCE FROM GSM AND SOCIAL NETWORKS

Before going into detail of works on social networks, we must highlight that the task of relationship inference in social networks is only an affine topic, and it is more trivial since much information about friendship is available in the data. In the domain of social network analysis there are several works that try to infer the relationship between objects. They use the information of place check-in, or shared geo-tagged photos. (CRANDALL et al., 2010), (PHAM; SHAHABI;

LIU, 2013) and (WANG; LI; LEE, 2014). Other works use information of phone logs and calls like (EAGLE; PENTLAND; LAZER, 2009).

A friendship network structure based on mobile phone data was proposed in (EAGLE; PENTLAND; LAZER, 2009). A group of students answered if another member of the group is a friend or not. In a first step, each pair of objects is classified as "Reciprocal Friends", "Non-Reciprocal Friends" and "Reciprocal Non-Friends". Then, the friendship relations are inferred using proximity information of the mobile phones, based on phone logs, calls, and Bluetooth connections. Finally, the proximity of these people is evaluated based on the day of the week and the kind of place the objects met. The output is a friendship network, where each node is an individual and the edge corresponds to a score based on the proximity of the work.

Cho, in (CHO; MYERS; LESKOVEC, 2011), proposed a model to predict future locations of moving objects based on relationships extracted from social networks and periodic movement patterns extracted from cell phone tower location logs. With the proposed model, he shows that social relationships can explain about 10% to 30% of all human movement, while periodic behavior explains 50% to 70%.

Crandall, in (CRANDALL et al., 2010), proposed a model to infer social ties between users, where spatio-temporal co-occurrences are detected based on shared photos of the photo-sharing site Flickr. First, the space is divided in cells, then if both users shared a photo within t days in the same cell, a co-occurrence is detected. Finally, the number of different cells visited by the same pair is counted, and based on the number of different co-locations between the pair of objects a probabilistic model is used to compute the friendship.

A model to infer social strength called EBM (Entropy-Based Model) was proposed by Pham in (PHAM; SHAHABI; LIU, 2013). It analyses information from the social network *Gowalla*, that allows the users to make check-ins. First, the co-occurrences are computed for each pair of objects. The co-occurrences are coincident check-ins of the objects, considering space and time. Finally, the relationship is calculated considering the entropy of the places and the frequency of the co-occurrences. The entropy of a visited place is used to avoid casual meetings based on the idea that a person that meets another person at a popular place, is probably in a casual meeting. We avoid this problem in a different way, considering the number of different encounter areas that objects meet. For example, suppose that a group of students has encounters at the University, at a cafe, and at the cinema. All these places are visited by a lot of people and the entropy of all pla-

ces would be very high, making the group of students having a lower relationship strength. However, we claim that if they have encounters at different places with a considerable duration, they probably have a higher relationship degree.

Another model to infer relationship strength between pairs of users based on check-ins from location-based social networks was recently proposed by Wang in (WANG; LI; LEE, 2014). This approach is based on personal, global, and temporal factors. The personal factor considers an individual user probability to visit a certain location. The global factor captures the popularity of a location to the general public. The temporal factor considers the time gaps between consecutive meeting events.

Check-ins from social networks are very interesting to infer relationships between people, but when we want to measure the relationship among animals or people which do not use social networks, GPS trajectories are a possible way to measure these relationships.

Table 1 summarizes the related works. The first four works ((LAUBE; KREVELD; IMFELD, 2005), (BAK et al., 2012), (GUDMUNDSSON; KREVELD; SPECKMANN, 2007) and (DODGE; WEIBEL; LAUTENSCHÜTZ, 2008)) only define the encounter pattern, but do not try to infer the relationship degree. None of them compute level-wise encounters, between two, three, or more objects. They do only verify if an encounter occurred, given a pre-determined region, a set of trajectories and a minimum number of objects. In our proposal we compute all encounters that happened between a set of objects, giving the *begin* and *end* time of these encounters, and we also calculate the area where the encounter happened, based on the trajectories that had an encounter.

Some works uses different types of data (as social network and cell phone logs) to infer the relationship degree among objects as (CRANDALL et al., 2010), (PHAM; SHAHABI; LIU, 2013), (WANG; LI; LEE, 2014), (CHO; MYERS; LESKOVEC, 2011), (EAGLE; PENTLAND; LAZER, 2009). In this thesis we is focus in computing the relationship among moving objects based on GPS trajectories.

The most related works to our proposal are (MA et al., 2014) and (BRILHANTE et al., 2012). Both works use GPS data to infer relationship degree. However, both works use only stops to infer the relationship degree. Ma in (MA et al., 2014) computes the relationship only between pairs of objects, and Brilhante, in (BRILHANTE et al., 2012), suggests communities of interesting places, that can be considered as a relationship between places.

The methods proposed in this thesis (*MORE* and *MORE++*),

are the only ones that compute the relationship degree among moving objects, based on their GPS trajectories, considering the duration, frequency, and distinct areas of their level-wise encounters. The method *MORE++* also uses the encounter and collecting days to compute the relationship degree.

3 ALGORITHMS BEINGTOGETHER AND MORE

In this chapter we first present the main concepts to define an encounter pattern and the relationship degree between moving objects (Section 3.1). In Section 3.2 we present two algorithms: the first one is *BeingTogether*, developed for efficiently detecting level-wise encounters between moving objects; the second one is *MORE* (Moving Objects Relationship inference from Encounters), an algorithm to infer the relationship degree between a group of two or more objects.

3.1 MAIN CONCEPTS

We start our definitions with the well known concepts of *point*, *trajectory*, and *subtrajectory*, inspired by the definitions presented in (SIQUEIRA; BOGORNY, 2011) and (BOGORNY et al., 2014).

Definition 1. *Point.* A point p is a tuple (x, y, t) , where x and y are geographic coordinates that represent a position in space and t is the timestamp in which the point was collected.

A trajectory is an ordered list of points that corresponds to the position of the object in space at a time, as presented in Definition 2. In this work, the historical data of an object is pre-processed in order to be represented in only one trajectory.

Definition 2. *Trajectory.* A trajectory $T_o = \langle p_1, p_2, p_3, \dots, p_n \rangle$ is an ordered list, where o is the object identifier, $p_j = (x_j, y_j, t_j)$ and $t_1 < t_2 < t_3 < \dots < t_n$.

It is well known that several trajectory patterns do not hold for an entire trajectory, but only in a trajectory part. For the encounter pattern it is not different. Two objects may not be together during the complete movement, but only in parts of their movements. These parts are called subtrajectories. The definition of subtrajectory is given in Definition 3.

Definition 3. *Subtrajectory.* A subtrajectory s of T is a list of consecutive points $\langle p_k, p_{k+1}, \dots, p_{k+l} \rangle$, where $p_i \in T$ and $k + l \leq n$.

(LAUBE; KREVELD; IMFELD, 2005), (GUDMUNDSSON; KREVELD; SPECKMANN, 2007) and (DODGE; WEIBEL; LAUTENSCHÜTZ, 2008) define an encounter as a set of objects that are close in space and time.

Our definition of encounter is a bit different, where we compute encounters of every object in the database with all other objects which stay close in space and time, for a minimal amount for time. For example, in Figure 1, we compute the encounters between Louis and Marie, between Louis, Marie and John, and between Louis, Marie and Susan. We do not require a minimal number of objects, since we are, in fact, interested in all possible encounters of any number of two or more objects. We are interested in encounters of two people, three, four, five, and so on, since we are going to analyze each of these encounters.

In this work we are not interested in distinguishing stationary and moving encounters, because we want to know when two or more objects stay together. Definition 4 presents the concept of encounter. For the sake of simplicity, in the following we will restrict the definitions for two moving objects, but note that the generalization to more than two objects is straightforward.

Definition 4. *Encounter.* Let $T_1 = \langle p_1, p_2, p_3, \dots, p_n \rangle$ and $T_2 = \langle q_1, q_2, q_3, \dots, q_m \rangle$ be two trajectories. Let $s_1 = \langle p_a, p_{a+1}, \dots, p_{a+u} \rangle$ and $s_2 = \langle q_b, q_{b+1}, \dots, q_{b+v} \rangle$ be two subtrajectories of T_1 and T_2 , respectively. T_1 and T_2 have an encounter at two maximal subtrajectories s_1 and s_2 w.r.t a spatial distance Δ_d , a temporal tolerance Δ_t and a minimum duration *minTime* IIF the following conditions hold:

- $\forall p_i \in s_1, \exists q_j \in s_2 \mid \text{spatialDist}(p_i, q_j) < \Delta_d \wedge \text{temporalDist}(p_i, q_j) < \Delta_t$
- $\forall q_j \in s_2, \exists p_i \in s_1 \mid \text{spatialDist}(q_j, p_i) < \Delta_d \wedge \text{temporalDist}(q_j, p_i) < \Delta_t$
- $(\min(p_{a+u}.t, q_{b+v}.t) - \max(p_a.t, q_b.t)) > \text{minTime}$

where the functions *spatialDist()* and *temporalDist()* compute, respectively, the Euclidean distance and the temporal distance between the points p_i and q_j .

To the best of our knowledge, there are no works in the literature which take into account the area where two or more objects have an encounter. In this work, as we want to measure the relationship degree of all combinations of objects, just assuming that the objects stay together (close) in space for a certain amount of time is not enough. However, if we consider that two objects have meetings at different places, we reduce the fact that two objects meet only by coincidence. For instance, two objects that leave in a nearby area and work at a nearby place (e.g. in a shopping center), will be detected as having encounters, even if these encounters represent a coincidence. However, if we consider that these two objects that have frequent encounters at

the same places (e.g. nearby homes and working place) but have encounters also in different areas, the probability of coincidence will be reduced, and the confidence that these objects know each other will increase the relationship degree. Therefore, we introduce the definition of encounter area, in order to evaluate all different places (areas) where two or more objects meet.

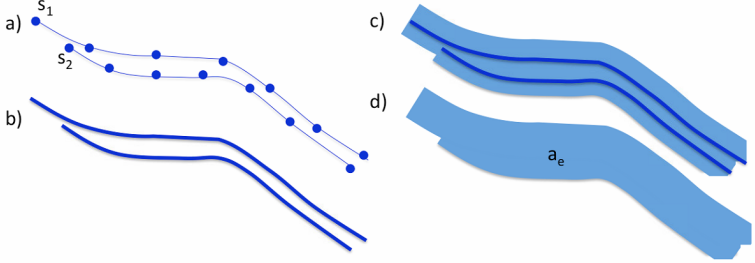


Figure 2 – Encounter Area

We define encounter area as a buffer with radius $\frac{\Delta_d}{2}$ around the union of the subtrajectories of all objects involved in the encounter, with each subtrajectory represented as a line. In Figure 2 (a), the subtrajectories s_1 and s_2 are having an encounter e . In Figure 2 (b) each subtrajectory is transformed to a line. In Figure 2 (c) each line is transformed into a polygon, adding a buffer of size $\frac{\Delta_d}{2}$. In Figure 2 (d) the polygons are unified, forming the encounter area a_e . More formally,

Definition 5. *Encounter area.* Let e be an encounter between the subtrajectories s_1 and s_2 , w.r.t Δ_d , Δ_t and $minTime$. The encounter area a_e is given by the formula:

$$a_e = buffer \left(makeLine(s_1), \frac{\Delta_d}{2} \right) \cup buffer \left(makeLine(s_2), \frac{\Delta_d}{2} \right) \quad (3.1)$$

where *makeline* is a function that transforms a set of points of a subtrajectory s in a line, and *buffer* is a function that builds a polygon of size $\frac{\Delta_d}{2}$ around s .

In order to take into account the area of the encounter, hereafter we refer to encounter as a tuple $e = (O, beginTime, endTime, a)$, where O is the set of objects involved in the encounter, *beginTime* and *endTime* are, respectively, the start and end time of the encounter, and a is the encounter area.

To define the relationship degree between two or more objects based on their encounters, we consider three main criteria: the *frequency* of the encounters, the *duration*, and the *different areas* where the encounters take place.

The frequency reveals how many times two or more objects meet.

Definition 6. *Frequency-Based Relationship Degree.* Let $DB = e_1, e_2, \dots, e_n$ be a set of encounters w.r.t. Δ_d, Δ_t and $minTime$, of all sets of moving objects in a trajectory database. Let $E(o_i, o_j)$ denote the set of all encounters between any objects o_i and o_j . The frequency-based relationship degree between a pair (o_1, o_2) is given by:

$$R_f(o_1, o_2) = \frac{|E(o_1, o_2)|}{\max(|E(o_i, o_j)|)} \quad (3.2)$$

where $|X|$ represents the cardinality of X and, $\max(X)$ is the highest number of encounters that any other pair of objects had in the same database.

The division by the maximum number is used to normalize the results with values from 0 to 1. This normalization is useful in the comparison of the results over different parameter values.

The duration of an encounter tells how much time two objects spend together. We assume that the higher the duration of an encounter is, the higher will be the relationship between the objects. The duration of an encounter is given by the subtraction of the *endTime* and *beginTime* of an encounter. When we sum the duration of all the encounters between a pair of objects, we have all the time this group stayed together. The duration-based relationship degree is given in Definition 7.

Definition 7. *Duration-Based Relationship Degree.* Let $DB = e_1, e_2, \dots, e_n$ be a set of encounters w.r.t. Δ_d, Δ_t and $minTime$, of all sets of moving objects in a trajectory database. Let $E(o_i, o_j)$ denote the set of all encounters between o_1 and o_2 . The duration-based relationship degree between o_1 and o_2 is:

$$R_d(o_1, o_2) = \frac{\sum_{z=1}^{|E(o_1, o_2)|} (endTime_z - beginTime_z)}{\max\left(\sum_{z=1}^{|E(o_i, o_j)|} (endTime_z - beginTime_z)\right)} \quad (3.3)$$

where z represents each encounter between o_1 and o_2 .

The first idea when we think about defining a relationship degree

between a group of several objects is to use the duration and frequency. However, in downtown areas we can find different objects close to each other in space and time. In these cases, people who take the same bus together everyday could have a strong relationship, when they not even know each other. To reduce this problem we define that a group that has encounters in different areas has a higher relationship degree. The more different areas two or more objects meet, the higher is the probability for the objects to know each other. Therefore, we define the area-based relationship according to Definition 8.

Definition 8. *Area-Based Relationship Degree.* Let $DB = e_1, e_2, \dots, e_n$ be a set of encounters w.r.t. Δ_d, Δ_t and $minTime$, of all sets of moving objects in a trajectory database. Let $E(o_1, o_2)$ denote the set of all encounters between o_1 and o_2 . Let $A(o_1, o_2) = \{a_1, a_2, \dots, a_r\} | a_1 \cap a_2 \cap \dots \cap a_r = \emptyset$ be the set of different geometric encounter areas between o_1 and o_2 . The area-based relationship degree between o_1 and o_2 is:

$$R_a(o_1, o_2) = \frac{|A(o_1, o_2)|}{\max(|A(o_i, o_j)|)} \quad (3.4)$$

For a better understanding of the function $A(o_i, o_j)$, in Figure 3 we show four examples of encounter areas. In Figure 3 (a) the result of the function that returns the number of different areas is 2. For all other examples in Figure 3 (b), (c) and (d) the number of different areas is 1, since all the areas are connected.

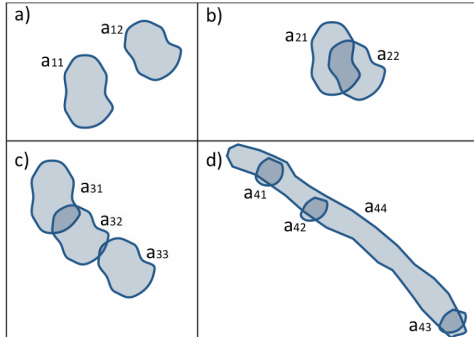


Figure 3 – Different Areas

Since the Area-Based Relationship Degree considers the number of *different* areas where a group of objects had encounters. When a set of objects walking together along an avenue, and also had encounters

in several different places in this same avenue, only *one encounter area* will be considered for this group.

Considering duration, frequency, and encounter area, the final relationship degree between two or more objects is computed by the sum of the degrees multiplied by a respective weight, as shown in Definition 9.

Definition 9. *Relationship Degree.* Let $DB = e_1, e_2, \dots, e_n$ be a set of encounters w.r.t Δ_d , Δ_t and $minTime$ of all sets of moving objects in a trajectory database. Let $E(o_i, o_j)$ denote the set of all encounters between o_1 and o_2 . The final relationship degree between o_1 and o_2 is computed as:

$$R(o_1, o_2) = R_f(o_1, o_2).w_{R_f} + R_d(o_1, o_2).w_{R_d} + R_a(o_1, o_2).w_{R_a} \quad (3.5)$$

with $w_{R_f} + w_{R_d} + w_{R_a} = 1$

We balance the frequency, duration, and encounter area with the weight, in order to generalize our approach to different applications. For example, investigative applications may give a higher weight to duration and frequency, and a lower to the area, since all objects are suspicious, every meeting independent of the area is important. For applications that try to infer friendship, the area is important to avoid casual meetings and should have its weight increased. In applications that try to measure the strength of the relationship between animals, one may balance the weights, because all the aspects could have the same importance.

In the following section we present the algorithms to compute encounters and relationships.

3.2 BEINGTOGETHER: AN ALGORITHM TO COMPUTE ENCOUNTER PATTERNS

In this section we first present the algorithm to detect encounters between multiple objects (Listing 3.1), which is the most time consuming step. The input of the algorithm *BeingTogether* is: a set of trajectories T , the time tolerance Δ_t , the spatial distance Δ_d , and $minTime$, the minimum time for detecting an encounter. The output is the set of detected encounters E .

Listing 3.1 – BeingTogether Algorithm

```

1  Algorithm BeingTogether
2  Input:  $T$  //set of trajectories
3            $\Delta_t$  //Time Tolerance
4            $\Delta_d$  //Distance Threshold
5            $minTime$  //Minimal time duration for an encounter
6  Output:  $E$  //set of detected encounters
7
8   $P$  = empty set
9   $C$  = empty set
10  $P$  = getAllPoints( $T$ ).sortByTime()
11 for ( $i = 0; i < P.size() - 1; i = i + 1$ ) do
12      $p = P.get(i)$ 
13      $temporalWindow = true$ 
14      $j = i + 1$ 
15     while ( $temporalWindow$  and  $j < P.size()$ ) //satisfies  $\Delta_t$ 
16          $pNext = P.get(j)$ 
17         if ( $p.oid \neq pNext.oid$ )
18             if ( $|p.t - pNext.t| < \Delta_t$ )
19                 if ( $spatialDistance(p, pNext) < \Delta_d$ )
20                      $C = addOrMergeEncounter(C, p, pNext, getMaxEncounter(p))$ 
21                 end if
22             else
23                  $temporalWindow = false$ 
24             end if
25         end if
26     end while
27 end for
28  $E = excludeShortEncounters(C, minTime);$ 
29 forall  $encounters$  in  $E$ 
30      $computeEncounterArea(E, \Delta_d)$ 
31 end forall
32 return  $E$ 

```

To avoid the comparison of each point of a trajectory with all points of all trajectories in the database, we order by time all trajectory points in one list of points (line 10). Sorting this list by time, a point is compared only with the ones whose time is close, i.e., those which are in a temporal window of size Δ_t . Only if the time constraint is satisfied the spatial distance is tested, so reducing the algorithm processing time, since the spatial operation is the most expensive.

The first loop (line 11) compares every point in P with the next (except the last). For this comparison, the variable *temporalWindow* (line 13) controls the time of the points of the list, which allows comparing a point only with the ones that satisfy the time tolerance Δ_t .

While the time of the point p minus the time of the point $pNext$ is less than Δ_t , the variable *temporalWindow* is true, and the next point is verified (line 15). If the *oid* of points p and $pNext$ are different (line 17), then the algorithm tests if they are close in space (line 19). If this is the case, an encounter candidate is generated (line 20). Notice that when both time and space constraints are satisfied we can-

not simply generate a new encounter, but we have to check if these two points are already in an existing encounter or not. To verify this is not a trivial task, so this verification is performed by the function *addOrMergeEncounter* (line 20), detailed in Listing 3.2, and explained later. As the encounter must have a minimum duration of *minTime*, we exclude the candidates that do not have a minimum time duration, transforming each other candidate in an encounter (line 28).

The next step is to compute the area of each encounter (lines 29 to 31), according to Definition 5.

The input of the function *addOrMergeEncounter* (Listing 3.2) is: a set of encounter candidates *C*, the points *p* and *pNext*, the encounter candidate *maxC* (that has the maximum time and the same object identifier of *p*). The output is the updated set of encounter candidates *C*.

Listing 3.2 – addOrMergeEncounter

```

1  Algorithm addOrMergeEncounter
2  Input: C //set of encounter candidates
3          p //point
4          pNext //point
5          maxC // encounter candidate with the maximum time
6  Output: C //set of detected encounter candidates updated
7
8  newEncounter = true
9  if (maxC != null)
10     if (maxC.getOIDs().contains(pNext.getOid()))
11         if (maxC.getOIDs().size() == 2) //first case
12             maxC.addPoint(pNext)
13             newEncounter = false
14         else //second case
15             nearPoints = findNearPoints(maxC, pNext)
16             if (nearPoints.size() == (maxC.getOIDs().size() - 1))
17                 maxC.addPoint(pNext)
18                 newEncounter = false
19             end if
20         end if
21     else //third case
22         nearPoints = findNearPoints(maxC, pNext)
23         if (nearPoints.size() > 1)
24             newCandidateEncounter = createNewCandidateEncounter(pNext,
25                 nearPoints)
26             C.add(newCandidateEncounter)
27             newEncounter = false
28         end if
29     end if
30 end if
31 if (newEncounter)
32     newCandidateEncounter = createNewCandidateEncounter(p, pNext)
33     C.add(newCandidateEncounter)
34 end if
35 return C

```

The variable *newEncounter* is used to create a new encounter

when the point $pNext$ is not added to an existent encounter candidate (line 8). Comparing the new point $pNext$ with an existing encounter candidate $maxC$ we evaluate three cases. The first case (line 11) is when the encounter candidate already contains the object identifier of point $pNext$ and there are only two trajectories in the encounter candidate.

The second case (line 14) is when the encounter candidate already contains the object id of point $pNext$, i.e., $pNext$ is already involved in an encounter candidate, and the encounter candidate involves a group of more than two objects. If this is the case, the point $pNext$ must be compared to the maximum point (the point with the highest time) of all objects in the current encounter candidate (line 15 - function *findNearPoints*). If $pNext$ is near in space and time of all objects inside the encounter candidate (line 16), $pNext$ is added to the encounter candidate.

The third case (line 21) is when the encounter candidate does not contain the object identifier of $pNext$. When this is the case, the point $pNext$ must be compared with the maximum point (the point with the highest time) of all objects in the current encounter candidate. The points close in space and time to $pNext$ are held in the variable *nearPoints* (line 22). If there are at least two objects close to $pNext$ (line 23), a new encounter candidate is created among $pNext$ and *nearPoints* (line 24).

Figure 4 illustrates an example of the third case. The two points connected with the red line are p (circle) and $pNext$ (diamond). The encounter candidate c_1 is the $maxC$ (the last encounter candidate that contains the same object identifier of point p). In this case, the object with the trajectory represented by diamonds was not part of encounter candidate c_1 so far. So, it is necessary to verify if the points of the trajectory represented with stars is also close to $pNext$ (diamond). Since there are points of the star trajectory close to p and $pNext$, a new encounter candidate is created, involving the three objects.

When $pNext$ did not fit in any of three cases, a new encounter candidate is created between p and $pNext$ (lines 31 to 33).

The complexity of the algorithm *beingTogether* is given by the number of points of all trajectories (line 11), multiplied by the number of points that each point will be compared (line 15). In the worst case, the complexity is $O(n^2)$, if all points are compared to each other, which is very unlikely since all points must be inside Δ_t .

The sort step has a complexity of $O(n \log(n))$. All other functions inside the loops of the *BeingTogether* algorithm have complexity $O(1)$.

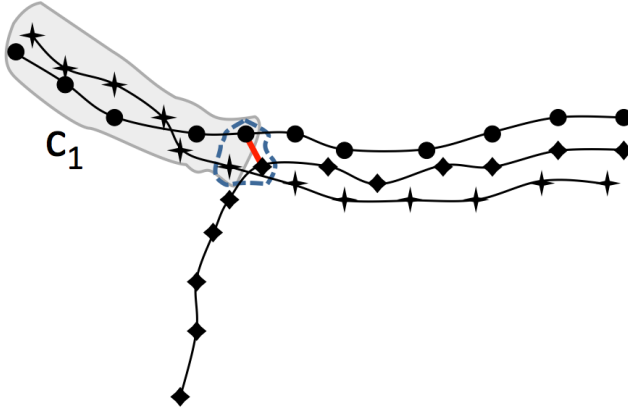


Figure 4 – Third case of `addOrMergeEncounter()` function

The function `getMaxEncounter()` is implemented using an index, so its complexity is $O(1)$. The function `addOrMergeEncounter()`, as can be seen in Listing 3.2 does not have any loop inside, and also has complexity $O(1)$.

In this section we presented the algorithm to detect the encounters, called *beingTogether*. The next section presents the algorithm to infer the relationship degree between groups of moving objects based on their encounters.

3.3 MORE: AN ALGORITHM TO COMPUTE THE RELATIONSHIP AMONG OBJECTS

The input of the algorithm *MORE* (Moving Objects Relationship inference from Encounters) (SANTOS et al., 2015), shown in Listing 3.3, is: the set of encounters E , the weights w_f , w_d and w_a . The output is the list of objects R with their respective relationships.

To compute the frequency, the duration, and the distinct areas of the encounters, the algorithm transforms the set of all encounters with their start and end time, into a list that contains each different group of objects with their respective encounters (line 8). Figure 5 shows this transformation for the encounters between Marie (*oid1*), Louis (*oid2*), John (*oid3*) and Susan (*oid4*), according to Figure 1.


Listing 3.3 – MORE Algorithm

```

1 Algorithm MORE
2 Input:  $E$  //set of encounters
3            $w_f$  //weight of Frequency
4            $w_d$  //weight of the Duration
5            $w_a$  //weight of the Area
6 Output:  $R$  //list of objects, with their relationship degree
7
8 encountersPerObjects = retrieveEncountersPerObjects( $E$ )
9  $max_f$  = getMaxFrequency(encountersPerObjects)
10  $max_d$  = getMaxDuration(encountersPerObjects)
11  $max_a$  = getMaxDiffAreasCount(encountersPerObjects)
12 for each set of objects  $O \in$  encountersPerObjects.values do
13      $R_f$  = getFrequencyOf(encountersPerObjects.get(o)) /  $max_f$ 
14      $R_d$  = sumDurationsOf(encountersPerObjects.get(o)) /  $max_d$ 
15      $R_a$  = getDistinctAreasOf(encountersPerObjects.get(o)) /  $max_a$ 
16      $result.R$  =  $R_f * w_f + R_d * w_d + R_a * w_a$ 
17      $R.put(O, result)$ 
18 end for
19 return  $R$ 

```

id	O	area	beginTime	endTime
e ₁	{1,2}	a ₁	5 th May 9h	5 th May 10h
e ₂	{1,2,3}	a ₂	5 th May 10h	5 th May 11h
e ₃	{1,2}	a ₃	5 th May 11h	5 th May 12h
e ₄	{1,2,4}	a ₄	5 th May 12h	5 th May 13h
e ₅	{1,2,3}	a ₅	12 th May 9h	12 th May 11h
e ₆	{1,2}	a ₆	12 th May 11h	12 th May 12h
e ₇	{1,2,4}	a ₇	12 th May 12h	12 th May 13h
e ₈	{1,2}	a ₈	19 th May 9h	19 th May 10h
e ₉	{1,2,3}	a ₉	19 th May 10h	19 th May 11h
e ₁₀	{1,2}	a ₁₀	19 th May 11h	19 th May 12h
e ₁₁	{1,2,4}	a ₁₁	19 th May 12h	19 th May 13h



oids	encounters
{1,2}	{e ₁ , e ₂ , e ₃ , e ₄ , e ₅ , e ₆ , e ₇ , e ₈ , e ₉ , e ₁₀ , e ₁₁ }
{1,2,3}	{e ₂ , e ₅ , e ₉ }
{1,2,4}	{e ₄ , e ₇ , e ₁₁ }

Figure 5 – (left) every encounter and (right) encounters grouped by objects

Since we have the list of objects with their respective encounters, the first step of the algorithm is to get the maximum values for the frequency (line 9), the duration (line 10) and the area of the encounters (line 11). Then, for each group of objects (line 12) the algorithm computes the frequency (line 13), the duration (line 14) and the area of the encounters (line 15) according to definitions 6, 7 and 8, respectively. Finally, the relationship degree is computed (line 16) and is added to a list R (line 17).

The complexity of the algorithm *MORE* is $O(g)$, where g is the number of groups of objects that have encounters, in the algorithm represented by the list *encountersPerObjects*.

3.4 RUNNING EXAMPLE

For a better understanding of the relationship inference we apply the *MORE* algorithm over the example shown in Figure 1. The output is illustrated on Table 2, which is sorted in descending order by R , forming a relationship degree rank. Table 2 shows the frequency, the duration, and the number of different encounter areas between Marie, Louis, John and Susan.

Table 2 – Relationship Measures for the running example

O	$freq.$	$dur.$	$area$	R_f	R_d	R_a	R
{Marie, Louis}	3	12	3	1	1	1	1
{Marie, Louis, John}	3	4	3	1	0.333	1	0.778
{Marie, Louis, Susan}	3	3	3	1	0.25	1	0.75

According to Figure 1, Marie and Louis stayed together from 9h to 13h at each one of the three days. Therefore, the frequency of this group is equal to 3, and the duration is 12h. Marie, Louis and Susan, stayed together for one hour on 5th of May (from 12h to 13h), one hour on 12th of May and another hour on 19th of May, so the total duration is 3 hours. Assuming that, in this example, the objects met at three different places, each group has 3 different encounter areas.

On observing the example in Figure 1, the group with the highest relationship during the period between 5th of May to 19th of May was Marie and Louis. Notice that the relationship degree between the objects Marie, Louis and John is a bit higher than between the objects Marie, Louis and Susan. This is because there was one encounter (Figure 1) at 12th of May between Marie, Louis, and John with duration of 2 hours (from 9h to 11h), while all others had the same one hour of duration.

4 EXPERIMENTAL EVALUATION

We performed experiments with two GPS trajectory datasets. The first dataset contains 11 trajectories with simulated encounters, where the encounters are known. The second dataset has 466 trajectories of students in the city of Amsterdam.

Since GPS devices may lose signal, the datasets were pre-processed in order to complete existing gaps in the original trajectories. We performed linear interpolation. When two sequential points of the same trajectory had a time difference higher than a given threshold, we placed points between the two original ones.

The experiments were performed in a Intel Core i5 2.4 Ghz processor, with 8Gb of RAM and a 256Gb SSD, using PostgreSQL and PostGIS extension. The algorithms were developed using the Java language.

4.1 UFSC DATASET

On August 13th, 2015, a group of eleven volunteers walked around the UFSC campus to simulate the encounters described in Table 3, generating a dataset with 29.329 points. This experiment has the intent to evaluate the correctness of the algorithms *BeingTogether* and *MORE*.

The participants received a smartphone, a map, and the instructions to visit three different places during the time described in Table 3. In this experiment, each participant had to visit 3 places and stay at each place for around 10 minutes. For instance, object 1 visited the *Place₁*, then *Place₇*, and finally *Place₉* (see first line of Table 3).

When two or more participants stayed at the same place at the same time, for at least 10 minutes, they had encounters. For instance, objects 1, 2, and 3 have only one encounter alone, at *Place₁*, and in the path to *Place₇*, they met object 4, generating a new encounter with four objects. At *Place₉*, objects 5 and 6 joined the group, forming a new encounter, now with 6 objects.

Objects 7 and 8, for instance, have two different encounters at *Place₃* (from 17:40 to 17:50 and 18:10 to 18:20). Object 11 did not meet any other object, so not having any encounter.

Table 3 – Scheduled Encounters at UFSC

oid	1st place	⇒	2nd place	⇒	3rd place
	[17:40, 17:50]	⇒	[17:55, 18:05]	⇒	[18:10, 18:20]
1	Place ₁	⇒	Place ₇	⇒	Place ₉
2	Place ₁	⇒	Place ₇	⇒	Place ₉
3	Place ₁	⇒	Place ₇	⇒	Place ₉
4	Place ₆	⇒	Place ₇	⇒	Place ₉
5	Place ₂	⇒	Place ₄	⇒	Place ₉
6	Place ₂	⇒	Place ₄	⇒	Place ₉
7	Place ₃	⇒	Place ₈	⇒	Place ₃
8	Place ₃	⇒	Place ₆	⇒	Place ₃
9	Place ₄	⇒	Place ₉	⇒	Place ₆
10	Place ₄	⇒	Place ₉	⇒	Place ₅
11	Place ₅	⇒	Place ₃	⇒	Place ₄

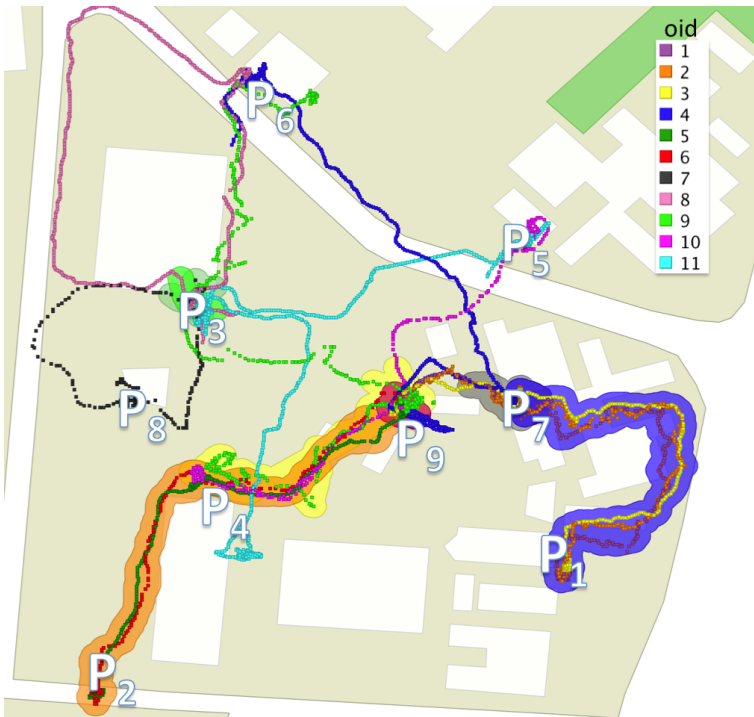


Figure 6 – Trajectories and Encounters at UFSC

In short, the students simulated 9 encounters, which are represented by rectangles in different colors in Table 3, and are visually represented in Figure 6.

Running the algorithm *BeingTogether* with parameters $\Delta_t = 30$ s, $\Delta_d = 15$ m and $minTime = 5$ minutes, it correctly detected all encounters, as shown in Table 4. Notice that there are encounters with two, three, four and six objects.

Table 4 – Encounters detected by *BeingTogether* at database UFSC

e	O	$beginTime$	$endTime$
e_1	{1,2,3}	17:40:00	17:55:17
e_2	{5,6}	17:40:00	18:06:58
e_3	{7,8}	17:40:00	17:50:21
e_4	{9,10}	17:40:00	18:06:10
e_5	{1,2,3,4}	17:54:20	18:05:48
e_6	{1,2,3,4,5,6}	18:12:24	18:20:48
e_7	{7,8}	18:09:25	18:20:46

Table 5 shows the relationship degree for each group of objects, and is sorted in descending order of R . As can be seen in Table 5, objects 1, 2 and 3 have the highest relationship degree (0.831). Those objects stayed together during all the experiment, and their encounters happened at three different areas. Objects 5 and 6 also stayed together during all the experiment. However, they are the second in the rank because their encounters happened at only two different areas. As can be seen in Table 3, objects 5 and 6 visited together $Place_2$ and $Place_4$, generating only one encounter in this case.

Although objects 7 and 8 have the highest frequency (2) of the experiment, they are only the third group of the rank. Because although they had two encounters, they were at the same place ($Place_3$), so

Table 5 – Relationship Degrees between objects at UFSC

O	$frequency$	$duration$	$area$	R_f	R_d	R_a	R
{1,2,3}	1	35.1	3	0.5	0.992	1	0.831
{5,6}	1	35.4	2	0.5	1	0.667	0.722
{7,8}	2	24.3	1	1	0.687	0.333	0.673
{1,2,3,4}	1	19.9	2	0.5	0.562	0.667	0.576
{9,10}	1	27.5	1	0.5	0.779	0.333	0.537
{1,2,3,4,5,6}	1	8.4	1	0.5	0.238	0.333	0.357

having only one encounter area.

The group of objects 1, 2, 3, 4, 5, and 6 had the lowest relationship degree (0.357), because this group had only one encounter, and this encounter had the lowest duration of the experiment (8.4 minutes). Object 11 (see Table 3) did not have any encounter during the experiment, so it has no relationship.

Notice that independently of the size of the group, if they had at least one encounter, their relationship degree is calculated. Even in a small dataset it is possible to understand the relevance of this information. The knowledge about relationships allows the understanding of which objects are the most related inside a trajectory database.

4.2 AMSTERDAM DATASET

This dataset was collected between 7 and 9 February 2005 in a mobile play game organized by Waag Society (RAESSENS, 2007). The game was held in the city of Amsterdam, the Netherlands, where students of 12-14 years old received GPS devices and a map with directions to locate historical places and were allocated in different teams, named by colors (yellow, orange, green, blue, red, and purple). In some historical places should occur encounters. We strongly believe that members of the same team should have several encounters and a high relationship. This dataset has 466 trajectories and, after interpolation, 1,942,851 points. Those points were plotted in the color of the team, as shown in Figure 7.

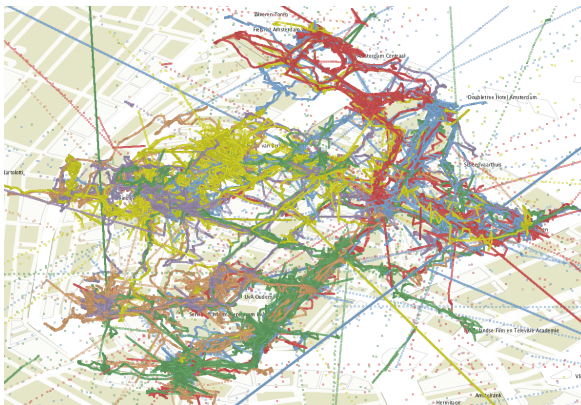


Figure 7 – Amsterdam database color teams

The 27 encounters detected on the Amsterdam database for parameters $\Delta_t = 30s$, $\Delta_d = 15m$ and $minTime = 15$ minutes are illustrated in Figure 8. Since this database does not have ground-truth for encounters, we used the known parameters from previous experiments. The 15m distance threshold was used due to the GPS devices poor accuracy. Similar values were used in related work to detect encounters (BAK et al., 2012). We also do not have information about the time synchronization of the GPS devices and the value of 30s was enough to detect encounters in this dataset. First, we evaluate the algorithm for $minTime = 15$ minutes considering that the objects must spend some time together in order to find the clues of the game.

Figure 8 shows the encounters detected with these parameters.



Figure 8 – Encounters at Amsterdam database considering $minTime = 15$ minutes

The result of *MORE* is illustrated in Table 6. When the value of $minTime$ is 15 minutes, from the total of 27 different groups that have a relationship, 16 were detected between objects of the same team. Every group of objects that has at least two objects from the same team are shown with the team color in Table 6. There are six groups

with more than two objects, and those have at least two objects of the same team. Among the objects with a relationship, 6 encounters are between objects of team *blue*, 4 for team *green*, 3 for team *orange*, and 1 for *red*, *yellow* and *purple*.

Table 6 – Relationship Degrees between objects at Amsterdam database with $minTime = 15$ minutes

<i>O</i>	R
{192933(red), 192962(blue)}	0.99
{192642(green), 193301(green)}	0.989
{192949(blue), 192962(blue)}	0.98
{193041(orange), 193537(orange)}	0.978
{193363(red), 193537(orange)}	0.939
{192634(green), 193301(green)}	0.922
{193174(orange), 193485(purple)}	0.833
{193109(blue), 193392(red), 193528(red)}	0.821
{192594(yellow), 192834(red)}	0.815
{192722(orange), 193418(orange)}	0.813
{192598(blue), 192962(blue), 193531(yellow), 193583(blue)}	0.812
{192682(orange), 193454(purple)}	0.81
{192674(green), 193166(green), 193298(green)}	0.81
{192727(orange), 192763(blue)}	0.789
{192623(blue), 193359(blue)}	0.7
{192873(blue), 192965(blue)}	0.694
{193041(orange), 193363(red), 193537(orange)}	0.684
{192933(red), 192949(blue), 192962(blue)}	0.669
{192681(red), 192883(yellow)}	0.669
{192980(yellow), 192988(purple)}	0.656
{192826(red), 193275(green)}	0.645
{192802(yellow), 193452(yellow)}	0.642
{193231(yellow), 193485(purple)}	0.639
{192634(green), 192642(green), 193301(green)}	0.628
{192557(purple), 192990(purple)}	0.612
{193062(orange), 193436(purple)}	0.582
{192949(blue), 193007(blue)}	0.577

We also evaluate our method increasing the value of $minTime$ to 30 minutes. The higher is the value of $minTime$, the lower is the

number of encounters. Then, 20 encounters were detected (see Figure 9 and Table 7).

When the minimal encounter duration is 30 minutes, 11 encounters were detected between objects of the same team. 5 encounters between objects of team *blue*, 2 for teams *green* and *orange*, and 1 for *red* and *yellow* teams. There are five groups with more than two objects, and again these groups have at least two objects of the same team. All the groups of the same team detected when the value of *minTime* was 30 minutes (Table 7), were also detected when *minTime* was 15 minutes (Table 6).

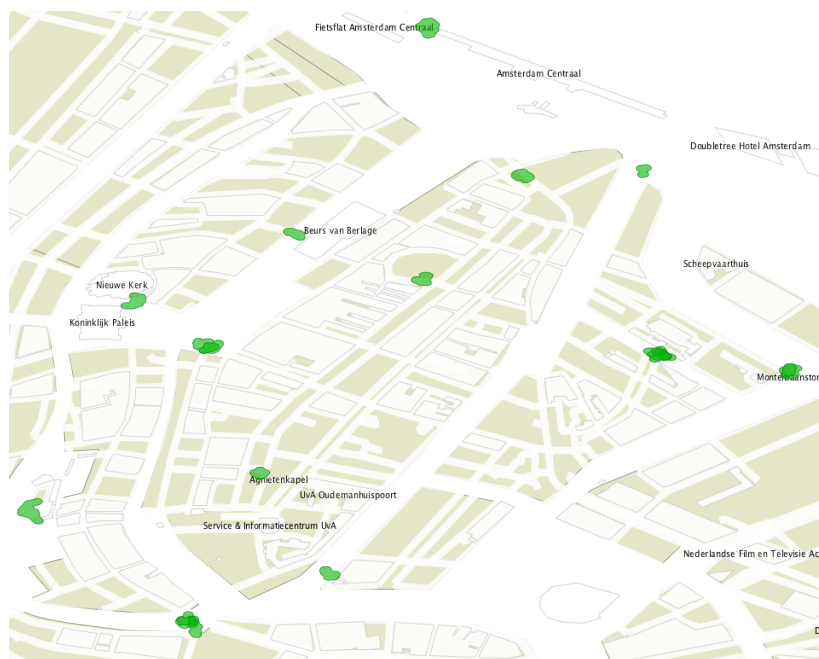


Figure 9 – Encounters at Amsterdam database with *minTime* = 30 minutes

Table 7 – Relationship Degrees between objects at Amsterdam database with $minTime = 30$ minutes

O	R
{192933(red), 192962(blue)}	0.99
{192949(blue), 192962(blue)}	0.98
{193174(orange), 193485(purple)}	0.833
{193109(blue), 193392(red), 193528(red)}	0.821
{192594(yellow), 192834(red)}	0.815
{192722(orange), 193418(orange)}	0.813
{192598(blue), 192962(blue), 193531(yellow), 193583(blue)}	0.812
{192682(orange), 193454(purple)}	0.81
{192674(green), 193166(green), 193298(green)}	0.81
{192727(orange), 192763(blue)}	0.789
{192623(blue), 193359(blue)}	0.7
{192873(blue), 192965(blue)}	0.694
{192642(green), 193301(green)}	0.694
{193041(orange), 193363(red), 193537(orange)}	0.684
{192933(red), 192949(blue), 192962(blue)}	0.669
{192681(red), 192883(yellow)}	0.669
{192980(yellow), 192988(purple)}	0.656
{192826(red), 193275(green)}	0.645
{192802(yellow), 193452(yellow)}	0.642
{193231(yellow), 193485(purple)}	0.639

4.3 PARAMETER ANALYSIS FOR THE AMSTERDAM DATASET

In order to evaluate the number of encounters and the computing time over the parameters Δ_t and Δ_d , we ran the *BeingTogether* algorithm several times using different combinations of both parameters. The tested values for Δ_t were 10s, 20s, 30s, 40s, 50s and 60s. The different values for Δ_d were 10m, 15m, 20m and 25m, what is reasonable for detecting encounters. These values were chosen after several initial tests.

Figure 10 shows the number of detected encounters for different values of Δ_t when the value of Δ_d is fixed in 15 meters. When the value of Δ_t is 10s, 20s, or 30s the number of encounters is less than 30. When the value of Δ_t is 60s, almost 50 encounters were detected. The number of encounters increases in relation to the value of Δ_t . If all the

devices are in a synchronized time, the value of this parameter should be reduced.

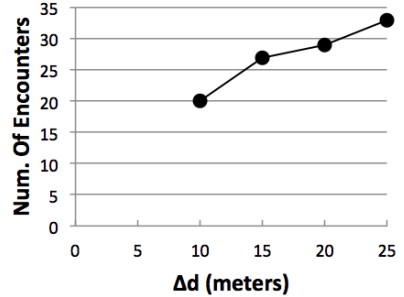
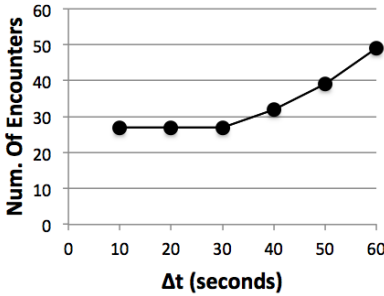


Figure 10 – Number of Detected Encounters for different values of Δ_t

Figure 11 – Number of Detected Encounters for different values of Δ_d

The number of encounters detected for different values of Δ_d , when Δ_t is fixed in 10 seconds, is shown in Figure 11. When the value of Δ_d is 10m, 20 encounters were detected, but when Δ_d is 25 meters this values go to more than 30 encounters.

The value of Δ_d also contributes for the number of detected encounters. This parameter values depend on the precision (in meters) of the collecting data device. The lower the precision of the devices, the higher should be the value of Δ_d .

The number of encounters is an important aspect to data analysis. The more reliable are the data, the lower will be the values for Δ_t and Δ_d . However, even when the dataset has precision limitations, the algorithm could also detect interesting patterns.

We also evaluated the computation time of the *BeingTogether* algorithm. This experiment shows how scalar this algorithm is. The computation time for the *BeingTogether* algorithm for different values of Δ_t , when the value of Δ_d is fixed in 15 meters, is illustrated in Figure 12. When Δ_t is 10s, the computation time is around 5 minutes. For the maximum observed value of Δ_t (60s), the computation time is higher than 40 minutes.

The higher the value of Δ_t is, the higher will be the number of points that will also have the Δ_d verified. So, consequently, the processing time increases according to the time threshold.

The computation time for the *BeingTogether* algorithm over different values of Δ_d , when Δ_t is fixed in 10 seconds, is illustrated in

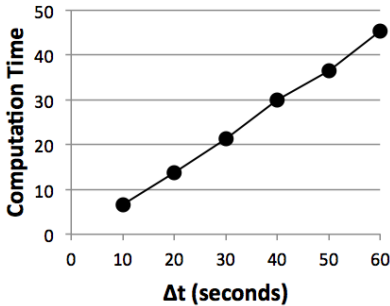


Figure 12 – Computation Time in minutes over different values of Δ_t

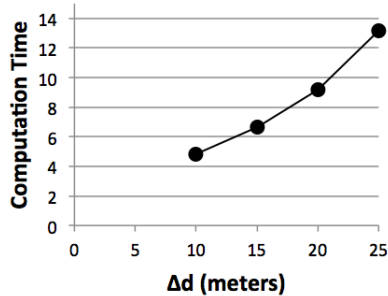


Figure 13 – Computation Time in minutes over different values of Δ_d

Figure 13. When the value of Δ_d is 10m, the computing time is around 5 minutes, and when Δ_d is 25 meters this value goes to around 13 minutes. This experiment shows that the variable that has more influence in the computation time is Δ_t , but Δ_d also contributes to increase this time.

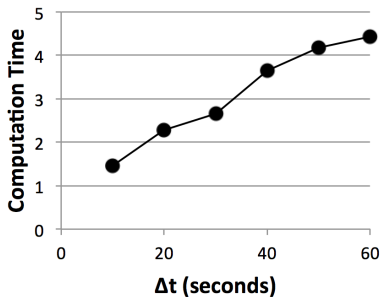


Figure 14 – Computation Time in seconds over different values of Δ_t in the original dataset

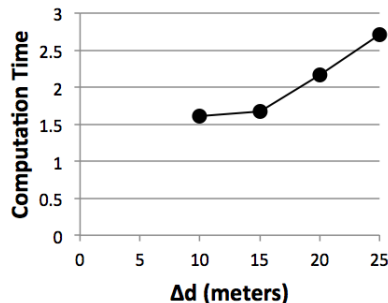


Figure 15 – Computation Time in seconds over different values of Δ_d in the original dataset

The higher the value of Δ_d is, the higher will be the number of times that the *addOrMergeEncounter* function will be used. Since this function compares a point with others, more time is needed to compute the encounters with a high value of Δ_d .

We also verified the computation time for the Amsterdam database with the original trajectories, without interpolation. This database has

62.758 points. In this case, the computation time was calculated in seconds. The results are shown in Figures 14 and 15.

4.4 DISCUSSION

The closest work to our proposal is (MA et al., 2014), which detects only relationships between pairs of objects and the result is quite different from ours, represented in a matrix. Ma et. al. considers the relationship from object A to object B different from the relationship from B to A, due to the user entropy. Our method returns a table with the ranking of the relationship degree for groups of objects, so both methods are not directly comparable.

To the best of our knowledge, there is no other work in the literature that infers the relationship degree between a set of moving objects considering frequency, duration and area of their encounters. However, in some cases, those measures are not the most appropriate. For instance, when some objects collect data during a longer time than others, and, consequently, have more encounters, their frequency, duration, and encounter area measures will always be higher than objects that collected data during a short period.

Another significant issue observed is when a set of objects remain together for a long time and go together to several different places. The *BeingTogether* algorithm only generates a new encounter if another object enters or leaves an existent encounter or if the objects get separated. In this case, only one encounter will be generated, with a unique encounter area. Since only one encounter is generated, the Frequency-Based Relationship Degree is underestimated. The Area-Based Relationship Degree will also be reduced, since only one encounter area is detected.

For instance, in the first experiment (UFSC), in Table 3, the objects {5, 6} visited Places 2, 4 and 9. However, only two encounters between them were generated. The first one between {5, 6} and the other with more objects.

On considering the mentioned limitations, in the following chapter we propose the algorithm MORE++, which overcomes these problems.

5 MORE++

In this chapter we present *MORE++*, a new algorithm to compute the relationship degree that takes into account the number of days that the objects collected their trajectories. This method also solves the encounter area and frequency problems discussed in Chapter 4.

5.1 MAIN DEFINITIONS

In order to compute the relationship degree for a group of objects in a database where the objects collected data in different periods and with different amount of time, we first present the new definition for Frequency-Based Relationship Degree (R'_f). In the *MORE* algorithm, the frequency was obtained by the division of the number of encounters of a group of objects, divided by the maximum number of encounters that any other set of objects had in the database. The new measure considers the number of days that a group had encounters divided by the number of collecting days for the same group.

Definition 10. *Frequency-Based Relationship Degree.* Let $DB = \{e_1, e_2, \dots, e_n\}$ be a set of encounters w.r.t. Δ_d, Δ_t and $minTime$ of all sets of moving objects in a trajectory database. Let $E(o_1, o_2)$ denote the set of all encounters between objects o_1 and o_2 . Let $f(E(o_1, o_2))$ represent the number of different days that objects o_1 and o_2 met and, $C(o_1, o_2)$ represent the number of intersecting days that the objects o_1 and o_2 collected data. The Frequency-Based Relationship Degree between a pair (o_1, o_2) is given by:

$$R'_f(o_1, o_2) = \frac{f(E(o_1, o_2))}{|C(o_1, o_2)|} \quad (5.1)$$

On considering that the object o_1 collected data on days $\{1, 3, 4, 5, 7\}$ and object o_2 collected data on days $\{1, 4, 5, 6\}$. The value of $C(o_1, o_2)$ will be $\{1, 4, 5\}$, then, the value of $|C(o_1, o_2)|$, in this case, is 3.

We claim that the more different days a group of objects meet, the higher should be their relationship. For instance, considering two groups of objects, where all objects collected data for 30 days. If two objects met 15 times at the same day, and another group had meetings in 15 different days, the objects that met in different days have a higher

relationship that those which met several times in only one day.

Figure 16 shows an example with 10 objects. The x axis represents the day in which the objects collected data. For instance, objects Jenny, Marcos, Britney and Chris collected data from day 2 to 30. The objects that had encounters are represented by the colored lines. The first four encounters (e_1 to e_4) had 1 day of duration. The encounter e_1 was among Jenny, Marcos, and Britney. Jenny and Marcos were the only two objects in the encounters e_2 and e_4 . Encounter e_3 involved Jenny, Marcos, Britney and Chris. Richard and Kate collected data during 27 days and had 4 days of encounter. Walter and Monica collected data during 25 days and had 4 days of encounter. Amanda and Liz had encounters during all the days they were collecting data.

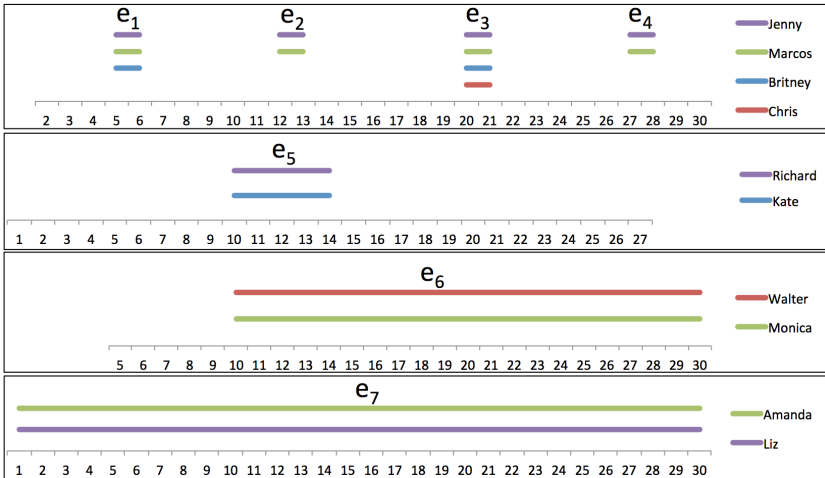


Figure 16 – Temporal Representation of encounters between the same individuals

By considering the new proposed measure, objects Amanda and Liz have the maximum value for Frequency-based Relationship Degree ($R'_f = 1$), since they had encounters every day they collected data. Objects Walter and Monica had encounters in 20 days, and collected data during 25 days, so their Frequency-based Relationship Degree will be 0.8 ($20/25$).

The duration of an encounter tells how much time two objects spent together. The intensity of the relationships is directly proportional to the duration of the encounters. The measure proposed in *MORE* for the Duration-Based Relationship degree was obtained by

the division of the duration sum of all encounters between a set of objects, divided by the maximum value of duration sum in the database. The new measure also considers the duration sum of all encounters of a group, but this value is now divided by the number of days that the objects collected data, converted to the same duration unit of the encounters (minutes, hours,...). The Duration-Based Relationship Degree is given in Definition 11.

Definition 11. *Duration-Based Relationship Degree.* Let $DB = \{e_1, e_2, \dots, e_n\}$ be a set of encounters w.r.t. Δ_d, Δ_t and $minTime$ of all sets of moving objects in a trajectory database. Let $E(o_1, o_2)$ denote the set of all encounters between objects o_1 and o_2 . Let $C(o_1, o_2)$ represent the intersecting days that objects o_1 and o_2 collected data, and t_c be the time conversion to transform the intersecting days for the same unit of the duration for encounters (minutes, hours, ...). The Duration-Based Relationship Degree between a pair (o_1, o_2) is given by:

$$R'_d(o_1, o_2) = \frac{\sum_{z=1}^{z=E(o_1, o_2)} (endTime_z - beginTime_z)}{|C(o_1, o_2)| \cdot t_c} \quad (5.2)$$

where z represents each encounter between o_1 and o_2 .

The value t_c represents a constant conversion used to keep the values in the same unit. For instance, if the encounter duration is measured in minutes, the value of $C(o_i, o_j)$ must be converted, and the value of t_c will be 24 hours multiplied by 60 minutes ($t_c = 1,440$).

Back to the example in Figure 16, observing the encounters between the objects Jenny and Marcos (e_1, e_2, e_3 and e_4), and between Richard and Kate (e_5), it is possible to observe that the values of R'_f and R'_d will be the same for those pairs.

The last feature used to measure the relationship degree is the encounter area. In Figure 16 we have three examples of objects that stayed together all time during consecutive days, so generating only one encounter ($\{Amanda, Liz\}$, $\{Walter, Monica\}$, $\{Richard, Kate\}$). Since we want to measure the number of *different* areas a group of objects stays together, we cannot use the whole encounter area to determine the number of different places. In order to obtain the number of different areas that a group of objects visited, we use the well-known concept of *stops* and *moves* (PALMA et al., 2008), to break large encounters in smaller ones. On breaking an encounter area in stops and moves, we obtain the places where the group spend time and they were moving

together.

To determine the Area-Based Relationship Degree we use the number of different stop-encounters and move-encounters of a group of objects, as given in Definition 12.

Definition 12. *Area-Based Relationship Degree.* Let $DB = \{e_1, e_2, \dots, e_n\}$ be a set of encounters w.r.t. Δ_d, Δ_t and $minTime$, of all sets of moving objects in a trajectory database. Let $E(o_1, o_2)$ denote the set of all encounters between o_1 and o_2 . Let $S(o_1, o_2) = \{s_1, s_2, \dots, s_r\} | s_1 \cap s_2 \cap \dots \cap s_r = \emptyset$ be the set of different stop-encounter areas between o_1 and o_2 . Let $M(o_1, o_2) = \{m_1, m_2, \dots, m_v\} | m_1 \cap m_2 \cap \dots \cap m_v = \emptyset$ be the set of different move-encounter areas between o_1 and o_2 . The area-based relationship degree between o_1 and o_2 is:

$$R'_a(o_1, o_2) = \frac{|S(o_1, o_2)| + |M(o_1, o_2)|}{\max(|S(o_i, o_j)| + |M(o_i, o_j)|)} \quad (5.3)$$

On considering duration, frequency, and encounter area, the final relationship degree between two or more objects is computed by the sum of the degrees multiplied by a respective weight, as shown in Definition 13.

Definition 13. *Relationship Degree.* Let R'_f be the Frequency-Based Relationship Degree. Let R'_d be the Duration-Based Relationship Degree. Let R'_a be the Area-Based Relationship Degree. Let w be the weight of each measure. The final relationship degree between o_1 and o_2 is computed as:

$$R'(o_1, o_2) = R'_f(o_1, o_2).w_{R_f} + R'_d(o_1, o_2).w_{R_d} + R'_a(o_1, o_2).w_{R_a} \quad (5.4)$$

with $w_{R_f} + w_{R_d} + w_{R_a} = 1$

In the following section we present the algorithm to infer the relationship degree between moving objects, according to the previous definitions, called *MORE++* (Moving Objects Relationship inference from Encounters).

5.2 MORE++ ALGORITHM

The input of the algorithm *MORE++*, shown in Listing 5.1, is: the set of encounters detected by the algorithm *BeingTogether E*, the weights w_f, w_d , and, w_a , and, the time conversion unit t_c . The output

is a list with the relationship degree of the moving objects R .

Listing 5.1 – MORE++ Algorithm

```

1  Algorithm MORE++
2  Input:   $E$  //set of encounters
3            $w_f$  //weight of Frequency
4            $w_d$  //weight of Duration
5            $w_a$  //weight of Area
6            $t_c$  //time conversion unit
7  Output:  $R$  //list of objects, with their relationship degree
8
9   $encountersPerObjects = retrieveEncountersPerObjects(E)$ 
10  $max_a = getMaxStopMoveCount(encountersPerObjects)$ 
11 for each set of objects  $O \in encountersPerObjects.values$  do
12    $C_o = C.getIntersectionCollectingDays(o)$ 
13    $f_o = getEncounterDaysOf(encountersPerObjects.get(o))$ 
14    $R'_f = \frac{f_o}{C_o}$ 
15
16    $d_o = sumDurationOf(encountersPerObjects.get(o))$ 
17    $R'_d = \frac{d_o}{C_o * t_c}$ 
18
19    $R'_a = \frac{countDistinctStopsMovesOf(encountersPerObjects.get(o))}{max_a}$ 
20
21    $result.R = R_f * w_f + R_d * w_d + R_a * w_a$ 
22    $R.put(O, result)$ 
23 end for
24 return  $R$ 

```

The first step is to organize the encounter list in the same form it was done by *MORE*, using the function *retrieveEncountersPerObjects()* (line 9). Once we have the list of objects with their respective encounters, it is possible to get the maximum value for the distinct stop and move encounter area (line 10).

The concept of stops and moves was defined for *one trajectory*. So, we had to adapt the CB-SMOT algorithm (PALMA et al., 2008) to detect the stops and moves for *encounters*. Each encounter, which is a set of subtrajectories (Definition 4), is transformed in a single trajectory. In this way, we could determine when a group of objects is moving or stopped.

For each group of objects (line 11), the algorithm computes the Relationship Degree based on Frequency (line 14), Duration (line 17) and, encounter areas (line 19) according to definitions 10, 11 and 12, respectively. Finally, the relationship degree is computed (line 19) and is added to a list R (line 17).

5.3 RUNNING EXAMPLE

In order to better understand the relationship inference of the *MORE++* algorithm, we applied it over the example shown in Figure 16. The output is illustrated in Table 8, which is sorted in descending order by R' . Table 8 shows the Frequency (R'_f), Duration (R'_d), and Area(R'_a) based relationship degree between Jenny, Marcos, Britney, Chris, Richard, Kate, Walter, Monica, Amanda and Liz. Notice that because the area of the encounters is not visually represented in the example of Figure 14, we will not use the area in the running example, giving weight zero for this measure, and 0.5 for both frequency and duration. Similarly, as the duration of the encounters is not represented in Figure 14, we considered 24 hours as the duration of each encounter. Therefore, the columns of frequency and duration have the same values in Table 8.

Table 8 – Relationship Measures for the running example *MORE++*

O	R'_f	R'_d	R'_a	R'
{Amanda, Liz}	1	1		1
{Walter, Monica}	0.8	0.8		0.8
{Richard, Kate}	0.15	0.15		0.15
{Jenny, Marcos}	0.14	0.14		0.14
{Jenny, Marcos, Britney}	0.07	0.07		0.07
{Jenny, Marcos, Britney, Chris}	0.03	0.03		0.03

On observing the example in Figure 16, the group with the highest relationship during the period was Amanda and Liz, since they had encounters every day of collecting data. Therefore, in Table 8 they have a relationship with the maximal degree: 1. Notice that the objects Walter and Monica (the second group of the rank) also stayed a long period together. They collected data for 25 days and had encounters in 20 days. Objects Richard and Kate collected data during 27 days, but having only 4 days of encounter, so their relationship degree is significantly reduced in relation to other objects. Objects Jenny and Marcos collected data during 29 days and also had 4 days of encounter. So, their relationship is very similar of the relationship between Richard and Kate, that also had 4 days of encounter, but collected data during less days. Jenny, Marcos and Britney met for 3 days. The lowest R' value was between objects Jenny, Marcos, Britney and Chris, which had only one encounter during the 29 days of data collecting.

5.4 EXPERIMENTAL EVALUATION

To evaluate our second proposal, i. e., the *MORE++* algorithm, we generated another dataset with more encounters. Table 9 shows this data, our ground truth for encounters. The dataset contains 15 objects that collected data during three different days, generating 13 encounters. For instance, objects {1, 2, 3} had 3 encounters at 3 different places (*Place*₁, *Place*₇, and *Place*₉). Objects 12 and 13 had 5 encounters at 6 different places. Notice that objects 12 and 13 had two collecting days (*Place*₇, *Place*₁, *Place*₈, *Place*₆, *Place*₃ and *Place*₂).

Table 9 – Scheduled Encounters at UFSC during 3 days

oid	day	1st place	⇒	2nd place	⇒	3rd place
		[17:40, 17:50]	⇒	[17:55, 18:05]	⇒	[18:10, 18:20]
1	1	<i>Place</i> ₁	⇒	<i>Place</i> ₇	⇒	<i>Place</i> ₉
2	1	<i>Place</i> ₁	⇒	<i>Place</i> ₇	⇒	<i>Place</i> ₉
3	1	<i>Place</i> ₁	⇒	<i>Place</i> ₇	⇒	<i>Place</i> ₉
4	1	<i>Place</i> ₆	⇒	<i>Place</i> ₇	⇒	<i>Place</i> ₉
5	1	<i>Place</i> ₂	⇒	<i>Place</i> ₄	⇒	<i>Place</i> ₉
6	1	<i>Place</i> ₂	⇒	<i>Place</i> ₄	⇒	<i>Place</i> ₉
7	1	<i>Place</i> ₃	⇒	<i>Place</i> ₈	⇒	<i>Place</i> ₃
8	1	<i>Place</i> ₃	⇒	<i>Place</i> ₆	⇒	<i>Place</i> ₃
9	1	<i>Place</i> ₄	⇒	<i>Place</i> ₉	⇒	<i>Place</i> ₆
10	1	<i>Place</i> ₄	⇒	<i>Place</i> ₉	⇒	<i>Place</i> ₅
11	1	<i>Place</i> ₅	⇒	<i>Place</i> ₃	⇒	<i>Place</i> ₄
12	1	<i>Place</i> ₇	⇒	<i>Place</i> ₁	⇒	<i>Place</i> ₈
13	1	<i>Place</i> ₇	⇒	<i>Place</i> ₁	⇒	<i>Place</i> ₈
12	2	<i>Place</i> ₆	⇒	<i>Place</i> ₃	⇒	<i>Place</i> ₂
13	2	<i>Place</i> ₆	⇒	<i>Place</i> ₃	⇒	<i>Place</i> ₂
14	3	<i>Place</i> ₆	⇒	<i>Place</i> ₃	⇒	<i>Place</i> ₂
15	3	<i>Place</i> ₆	⇒	<i>Place</i> ₃	⇒	<i>Place</i> ₂

Each different encounter is represented in Table 9 in a different color. All the encounters were correctly detected by the *BeingTogether* algorithm, according to our definition of encounter.

We ran *MORE* and *MORE++* over this dataset and the differences can be observed in Table 10. Objects 12 and 13 had the strongest relationship degree on both methods, since they have the highest duration, frequency, and visited more different areas.

Since *MORE* considers that all objects have the same time of collecting data, the only group that collected data during two days

Table 10 – Relationship Degrees between objects at UFSC

O	$MORE$	$MORE++$
{12, 13}	1	0.971
{14, 15}	0.318	0.905
{5,6}	0.362	0.866
{1,2,3}	0.427	0.865
{9,10}	0.253	0.694
{1,2,3,4}	0.291	0.642
{7,8}	0.299	0.562
{1,2,3,4,5,6}	0.172	0.451

{12, 13} had the highest relationship degree ($R = 1$). Other groups, that collected data for only one day, had very low relationship values for the method $MORE$ (all from 0.318 to 0.172). Even groups that stayed together during all the experiment had low grades, as {1, 2, 3} and {5, 6}.

Notice, in Table 9, that objects 12 and 13 visited 6 different places ($Place_7$, $Place_1$, $Place_8$, $Place_6$, $Place_3$ and, $Place_2$). From the colors in Table 9 one may notice that encounters were detected only at 5 different places because for $Place_1$ and $Place_8$ (red color) only one area was computed by the algorithm *BeingTogether*, given that the objects stayed together all the time. For $MORE$, the number of areas for the encounters of objects {12, 13} is equal to 5. To solve this problem, in the $MORE++$ we apply the *stops* and *moves* computation over the subtrajectories involved in the encounter, in order to split different areas that should not be merged. So, $MORE++$ will split the encounter in red in three different places: the stop at $Place_1$, a move between $Place_1$ and $Place_8$, and the stop at $Place_8$. So, the final number of areas for objects 12 and 13 is 7, since we consider the movement between places as a different encounter area. From our point of view, the best number of encounter areas is 7, since we compute all stopping areas and moving areas.

In the algorithm $MORE$, the frequency of encounters for objects {12, 13} is equal to 5, which is also the number of different areas. For $MORE++$, the encounter frequency of this group is 2, since they had encounters during two days, and the R'_f is maximum (1) because they had encounters every day they collected data.

For the group {14, 15}, for instance, notice from Table 9 that they meet at three different places ($Place_6$, $Place_3$, $Place_2$), but only one encounter area (gray color) was detected by the *BeingTogether*

algorithm. Therefore, for the *MORE* method the encounter frequency and area is 1, since only one encounter is detected, when these values are divided by the maximum values of frequency and area the relationship degree got too low. For *MORE ++*, 5 encounter areas were detected, the stop at $Place_6$, a move between $Place_6$ and $Place_3$, a stop at $Place_3$, a move between $Place_3$ and $Place_2$ and a stop at $Place_2$. The encounter frequency is maximum for *MORE ++* since objects 14 and 15 had only one day of collecting data.

It is worth mentioning that the value of the relationship degree computed by the *MORE ++* method (0.904) is more realistic than the degree computed by *MORE* (0.318), since objects 14 and 15 stayed together during all the time of collecting data.

The groups $\{1, 2, 3\}$, $\{5, 6\}$ and $\{14, 15\}$ stayed together during all the experiment and they should have high and similar values of relationship degree, correctly detected by the *MORE ++*, with values from 0.905 to 0.865. The values of their relationship was not the same because of variations on the duration of the encounters.

6 CONCLUSION

In the last two decades, there was a popularization of different GPS-enabled devices, that allow recording the moving objects location. Consequently, there was an increase in the amount of mobility data generated from these devices. To the best of our knowledge, none of the existing works in the literature proposed the inference of relationship degree between multiple objects based on their encounter patterns extracted from moving object trajectories.

In this thesis we proposed new definitions of encounter and encounter area. We also proposed a new method to compute encounters (*BeingTogether*) and two different methods to infer objects relationships (*MORE* and *MORE++*). The *MORE* algorithm considers the encounter duration, the frequency of encounters and the different encounter areas. This method works well for datasets where moving objects have similar number of data collecting days. *MORE* is useful for small datasets, as the Amsterdam, that has only two collecting days and for sports, where, for instance, the method could be useful to analyse the relationship between the players in a match. The *MORE++* considers the number of days of collecting data of the objects and improves the area-based relationship degree, considering stops and moves, in order to split encounter areas when two or more objects stay together during long periods. *MORE++* is useful to analyse historical data, weeks or months, for investigative applications, for instance. Both methods present conceptual advantages over related work such as the possibility to infer the relationship degree between multiple objects.

The main contributions of our proposal include: (i) new definitions of encounter pattern as well as the encounter area; (ii) new measures for achieving the relationship of individuals based on their encounters; (iii) efficient algorithms to compute encounters and infer the relationship degree of individuals from their trajectories.

We evaluated the proposed methods with a running example and performed an experimental study with real trajectory data in a simulated scenario, where the encounters were known. The results of the experiment showed that our method was able to identify relationships between pairs and groups of objects.

The main challenge of this thesis was the experimental evaluation, because there is no benchmark or ground-truth database available. The existent trajectory datasets do not have the relationship information,

and when a dataset has the relationship degree, the trajectories are not accessible. The necessity of interpolation is also an important issue to improve our method.

As a result of this work, a paper named *Inferring Relationships from Trajectory Data* was published in the Brazilian Symposium on Geoinformatics 2015 (SANTOS et al., 2015).

In the proposed approach, we use raw trajectory data without considering semantic information. However, as future work, we will investigate new measures and the use of semantic information to ensure the value of a relationship degree among a group of objects. With more reliable information, provided with semantics, the relationships could be labelled as couple, friends, family, and so on.

Since the lack of data is an open issue, another future work is to collect GPS data associated to the information about the relationship, and publish this data as a ground-truth for other works.

Another future work is to improve the *BeingTogether* algorithm to detect encounters without interpolation. The primary idea is to consider that the object was stopped when there is gap in the data.

REFERENCES

- ALVARES, L. O. et al. An algorithm to identify avoidance behavior in moving object trajectories. **Journal of the Brazilian Computer Society**, Springer, v. 17, n. 3, p. 193–203, 2011.
- BAK, P. et al. Scalable detection of spatiotemporal encounters in historical movement data. In: WILEY ONLINE LIBRARY. **Computer Graphics Forum**. [S.l.], 2012. v. 31, n. 3pt1, p. 915–924.
- BOGORNY, V. et al. Constant—a conceptual data model for semantic trajectories of moving objects. **Transactions in GIS**, Wiley Online Library, v. 18, n. 1, p. 66–88, 2014.
- BRAZ, F. J.; BOGORNY, V. **Introdução a Trajetórias de Objetos Móveis: conceitos, armazenamento e análise de dados**. [S.l.]: Univille, 2012. ISBN 9788582090022.
- BRILHANTE, I. R. et al. Cometogether: discovering communities of places in mobility data. In: IEEE. **Mobile Data Management (MDM), 2012 IEEE 13th International Conference on**. [S.l.], 2012. p. 268–273.
- CHO, E.; MYERS, S. A.; LESKOVEC, J. Friendship and mobility: user movement in location-based social networks. In: ACM. **Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining**. [S.l.], 2011. p. 1082–1090.
- CRANDALL, D. J. et al. Inferring social ties from geographic coincidences. **Proceedings of the National Academy of Sciences**, National Acad Sciences, v. 107, n. 52, p. 22436–22441, 2010.
- DODGE, S.; WEIBEL, R.; LAUTENSCHÜTZ, A.-K. Towards a taxonomy of movement patterns. **Information visualization**, SAGE Publications, v. 7, n. 3-4, p. 240–252, 2008.
- EAGLE, N.; PENTLAND, A.; LAZER, D. Inferring friendship network structure by using mobile phone data. In: PNAS. **Proceedings of the National Academy of Sciences (PNAS)**. [S.l.], 2009. p. 15274–15278.

GUDMUNDSSON, J.; KREVELD, M. van; SPECKMANN, B. Efficient detection of patterns in 2d trajectories of moving points. **Geoinformatica**, Springer, v. 11, n. 2, p. 195–215, 2007.

LAUBE, P.; KREVELD, M. van; IMFELD, S. Finding remo - detecting relative motion patterns in geospatial lifelines. In: WILEY ONLINE LIBRARY. **Computer Graphics Forum**. [S.l.], 2005. v. 31, n. 3pt1, p. 915–924.

LETTICH, F. et al. Detecting avoidance behaviors between moving object trajectories. **Data Knowl. Eng.**, v. 102, p. 22–41, 2016. Disponível em: <<http://dx.doi.org/10.1016/j.datak.2015.12.003>>.

MA, C. et al. Effective social relationship measurement based on user trajectory analysis. **Journal of Ambient Intelligence and Humanized Computing**, Springer, v. 5, n. 1, p. 39–50, 2014.

PALMA, A. T. et al. A clustering-based approach for discovering interesting places in trajectories. In: ACM. **Proceedings of the 2008 ACM symposium on Applied computing**. [S.l.], 2008. p. 863–868.

PHAM, H.; SHAHABI, C.; LIU, Y. Ebm: an entropy-based model to infer social strength from spatiotemporal data. In: ACM. **Proceedings of the 2013 international conference on Management of data**. [S.l.], 2013. p. 265–276.

RAESSENS, J. Playing history: Reflections on mobile and location-based learning. **Didactics of microlearning. Concepts, discourses, and examples**, p. 200–217, 2007.

SANTOS, A. A. dos et al. Inferring relationships from trajectory data. In: **XVI Brazilian Symposium on GeoInformatics, Campos do Jordão, São Paulo, Brazil, November 29 - December 2, 2015**. [s.n.], 2015. p. 68–79. Disponível em: <<http://urlib.net/8JMKD3MGPDW34P/3KP3262>>.

SIQUEIRA, F. de L.; BOGORNY, V. Discovering chasing behavior in moving object trajectories. **Transactions in GIS**, Wiley Online Library, v. 15, n. 5, p. 667–688, 2011.

WANG, H.; LI, Z.; LEE, W.-C. Pgt: Measuring mobility relationship using personal, global and temporal factors. In: IEEE. **Data Mining, 2014. ICDM'14. International Conference on**. [S.l.], 2014. p. 570–579.