Privacy and Analysis of Trajectories and Co-Trajectories

Joel Dahne

November 29, 2021

Contents

Abstract

A trajectory of an individual or object corresponds to its movement in space as a function of time. A co-trajectory is a collection of trajectories belonging to different individuals or objects. The thesis is divided into two parts about two different topics related to co-trajectories. The first part is about privacy-preservation in co-trajectories and introduces an algorithm called *SwapMob* for improving the privacy when handling co-trajectories. In the second part we consider different stochastic models for trajectories and co-trajectories that is an important tool when analyzing data from co-trajectories and using it to make predictions.

1 Introduction

With today's use of mobile phones and other location aware devices, such as GPS's, huge amounts of location based data is produced and used. This enables many new applications that can help both the individual and the crowd. For the individual this enables the use of many different location based services, for example navigation services (such as Google Maps or public transport applications), social games and networks (such as Pokemon GO) and services for finding interesting nearby places (such as Yelp). By aggregating data from multiple people it can also be used for large scale decision making, for example in city planning it can be used to find commuting patterns which is helpful for improving public transport and general infrastructure [1, 2].

Both of these use cases come with privacy concerns. Mobility data contains a lot of potentially sensitive information, among other things it can contain data about visits to hospitals and religious places along with time spent at home or at work. For location based services the individual wants the utility from the service but also privacy. If the service provider knows who you are, often the case with for example Googles services, then giving them access to your mobility data allow them to combine this with other data to learn more about you. Other times you are communicating with the service provider under a pseudonym, many public transport application works like this, in which case you are not directly linked to the data. However, due to the high degree of uniqueness of mobility data re-identification of the individual is still often possible [3]. Because of this a balance between privacy and utility when using location based data has to be found.

The focus of this thesis is on the usage of aggregated location data. We begin by giving a mathematical definition of trajectories and co-trajectories. After that the thesis is divided into two parts, one about privacy and one about stochastic models for co-trajectories. The privacy part begins with a brief introduction about previous work on the topic. Then we take a look at one method, SwapMob, useful for improving privacy when working with co-trajectories, which was introduced in the literature last year. We give a more formal definition of the method than previously done and analyse its properties. For the part about stochastic models we first present a very general model for co-trajectories. Then we look closer at a more specific Markov chain model.

2 Trajectories and co-trajectories

In this section we introduce and define trajectories and co-trajectories. We begin by, informally, explaining the meaning we associate with a trajectory and a co-trajectory and then continue by giving a formal definition.

A trajectory is given by the movement of an object, for example, a person or a bus, and can be seen as a function of time giving the location of the object at the time. For example we can consider the trajectory of a person on their way to work, or that of an airplane flying between two airports. Both of these examples are naturally continuous, the location changes continuously with time. We can consider trajectories which are discontinuous as well, for example the location of a persons home, which stays constant most of the time and then jumps to another place whenever the person moves.

When considering a number of trajectories together, each associated with a different object, we are talking about a collection of trajectories, a co-trajectory. A co-trajectory could for example be the trajectories of all, or some, buses in a city for a given day, or the trajectories of all airplanes leaving an airport. We can also have less homogeneous co-trajectories, such as a co-trajectory consisting of trajectories for the commuters in a city together with the trajectories of the buses.

When considering how to define a trajectory we have to take into account that the true movement of an object is in general not known. Most trajectories are naturally continuous, but we cannot capture or measure the continuous movement of the trajectory in practice. Typically, the location of an object at some time is given by a discrete measurement of its current location up to some precision. We therefore adopt a definition of a trajectory which is more related to how we get information about the movement of an object in practice, discrete measurements of its location in space and time.

Performing a measurement gives a datapoint, a datapoint is given by a time and a location for the object. How the location is given depends on what is being measured; for a car it could be given by GPS coordinates or the location on a map, for an airplane you could have GPS coordinates together with an altitude. We can also consider cases when the location is discrete and without an explicit representation in space, such as an address or a node in a graph that represents the road network for instance. In general we use the notation $\mathbb{L}$ for the set of locations, depending on the application different choices for $\mathbb{L}$ can then be used.

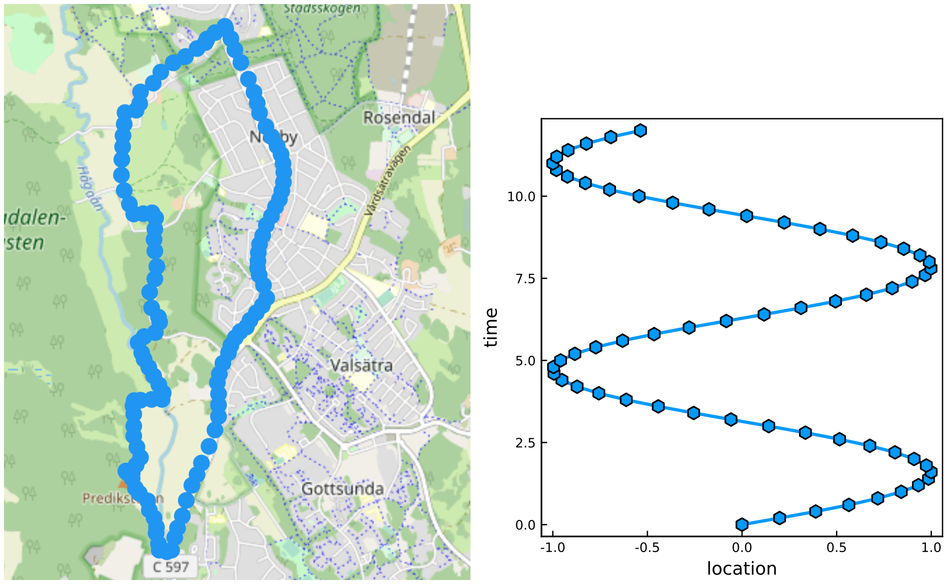
Definition 2.1 (**Datapoint**)**.** A datapoint, d, is given by $d = (x, t) \in \mathbb{L} \times \mathbb{R}_+$ giving the the location, x, and the time, t, for the datapoint. The set of all datapoints, $\mathbb{L} \times \mathbb{R}_+$, will be denoted by $\mathcal{D}$.

For a datapoint d we refer to x as the location of the datapoint and t as the time or the timestamp of it.

A trajectory is now given by a set of datapoints with no two datapoints sharing the same time. We do allow for an infinite number of datapoints but will require that in every finite time interval there is only a finite number of them. In practical applications the number of datapoints will of course be finite, but this is relevant when discussing stochastic models for trajectories in Section 5.

Definition 2.2 (**Trajectory**)**.** A trajectory, r, is given by a set of datapoints, $r \subset \mathcal{D}$, satisfying that no two datapoints have the same time and that for every finite interval of time there is only a finite number of datapoints with the time occurring in this interval. We denote the set of all trajectories by $\mathcal{T}$.

Not allowing datapoints occurring at the same time is reasonable since an object can in general not be in two places at the same time. However since in practice the time for the datapoint is necessarily discretized it could happen



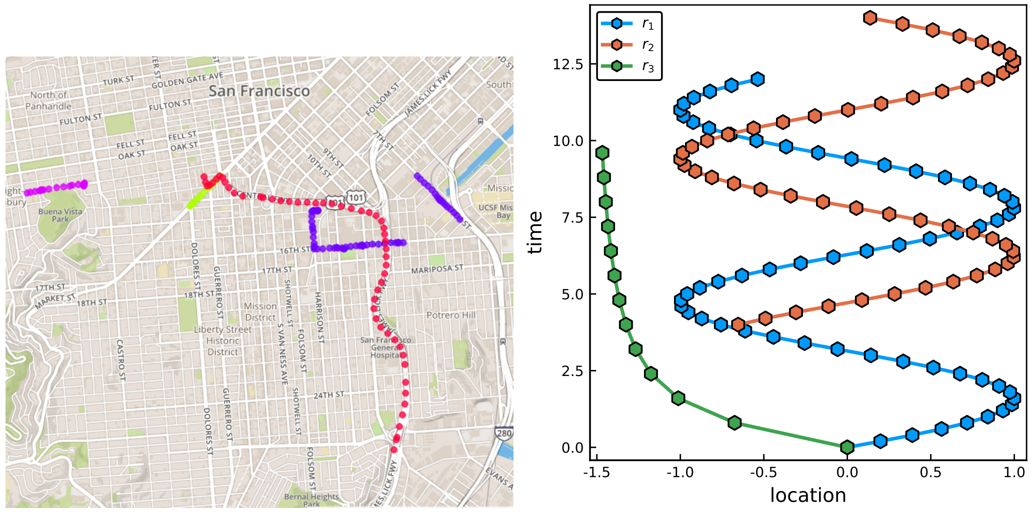
(a) Trajectory from a person cycling nearby Uppsala. $\mathbb{L}$ is here given by positions on Earth, in this case recorded as longitude and latitude.

(b) Trajectory with $\mathbb{L} = \mathbb{R}$ with 60 datapoints in the time interval $[0, 12]$ where the location for the datapoint at time t is given by $\sin(t)$.

Figure 1: Two different kinds of trajectories, one based on real world data and one on generated data.

that we have two datapoints that are captured at slightly different times but get the same discretized timestamp. The main problem with allowing datapoints with the same timestamp is that there is no natural ordering of the datapoints, we cannot order them just by their timestamp. If there is a particular application where multiple datapoints with the same timestamp could occur then you would have to choose one of them to use, how this is done will then depend on the application. See Figure 1 for two examples of trajectories.

A co-trajectory is a collection of trajectories. We allow multiple occurrences of the same trajectory (this would correspond to different individuals with the same trajectory) and it is therefore given by a multiset of trajectories. In our case we are only interested in co-trajectories with a finite number of trajectories but the definition could easily be generalized to infinite ones.



(a) A co-trajectory where the trajectories are given by taxi rides in San Francisco.

(b) A co-trajectory with $\mathbb{L} = \mathbb{R}$ consisting of three trajectories, $\mathcal{R} = \{r_1, r_2, r_3\}$.

Figure 2: Two different kinds of co-trajectories, one based on real world data and one on generated data.

Definition 2.3 (Co-trajectory). A co-trajectory, $\mathcal{R}$, consisting of N trajectories is given by a multiset $\{r_i\}_{i=1}^{N}$ of trajectories $r_i \in \mathcal{T}$.

Two examples of co-trajectories are given in Figure 2.

3 Privacy for co-trajectories

Co-trajectories can, in particular when the individual trajectories corresponds to people, contain a lot of sensitive data. It can reveal information about home and work place as well as information about visits to for example hospitals or religious places [4]. Even if the data is anonymized information can be leaked by for example connecting trajectories with people living at a certain address. For this reason it is important to take into account privacy when analysing and publishing data from co-trajectories [5].

A survey of the current state regarding privacy concerns related to location data is given by Primault et al. [6]. They discuss several different privacy threats coming from location data, these include *Mobility Prediction*, *Social*

Relationships, *Points of Interests* and *Re-identification*. Our focus will be on Re-identification, how an individual can be associated with a trajectory in a co-trajectory.

3.1 Re-identification for co-trajectories

Re-identification is in general possible due to the high degree of uniqueness for trajectories of individuals. For example, de Montjoye et al. [3] studied the uniqueness of mobility data for humans. They had data for one and a half million individuals given by location of mobile phones as determined by the carrier's antennas. In the study they conclude that knowledge of four datapoints for an individual were enough to uniquely identify 95% of the population. Very little information is thus needed to identify an individual even among a very large number of people.

In general re-identification of an individuals trajectory in a co-trajectory is done using partial data of the trajectory, in the above study this partial data were four datapoints from the trajectory. However, partial data about an individuals trajectory could come in many forms. It could be that we know some of the datapoints for the trajectory, or we know that the individual visited some specific area at a certain time. But it could also be less concrete things, such as knowing that the individual always takes a specific way when traveling between two places or that the individual visits the hospital at least once a week.

Since partial data about trajectories could come in many different forms we need a way to represent and model it to analyze it. To represent all of these different kinds of partial data we use a predicate on trajectories. A predicate, p, corresponding to some partial data is a function that given a trajectory returns true if it satisfies the partial data and false otherwise.

$$p(r) = \begin{cases} true & \text{if } r \text{ satisfies the partial data,} \\ false & \text{otherwise.} \end{cases}$$

For example, if we know one datapoint, d, for the trajectory we get the predicate p_d that would return true only for trajectories having this datapoint.

$$p_d(r) = \begin{cases} true & \text{if } d \in r \\ false & \text{otherwise.} \end{cases}$$

If we know that the individual visits the hospital at least once per week the predicate would return true only for trajectories doing that.

3.2 Measuring privacy

To discuss privacy related to co-trajectories we need a way to measure it. The literature contains many different ways for measuring privacy. One of the simplest, but still well used, methods is k-anonymity introduced by Sweeney [7]. A dataset satisfies k-anonymity if the information for each individual cannot be distinguished from at least $k-1$ other individuals.

The original definition of k-anonymity by Sweeney is suitable for tabular data. Attempts have been made to make it more suitable for trajectories and one version is (k, δ)-anonymity introduced by Abul et al. [8] which considers the imprecision when measuring location as one part of the anonymity. We take a slightly different approach and adapt it to our definition of predicates on co-trajectories. A co-trajectory satisfies k-anonymity with respect to a predicate p if at least k trajectories satisfy the predicate. If we for a predicate p denote by $p(\mathcal{R})$ the set of trajectories in $\mathcal{R}$ satisfying the predicate, then it satisfies k-anonymity if $|p(\mathcal{R})| \geq k$. An individual is uniquely identifiable with respect to a predicate p if $|p(\mathcal{R})| = 1$.

This can also be generalized to families of predicates. Given a family $\{p_i\}_{i \in I}$ of predicates a co-trajectory satisfies k-anonymity with respect to this family if $\min_{i \in I} p_i(\mathcal{R}) \geq k$.

One limitation of k-anonymity is that even though the individual is only one in a group of at least k it could be that all these individuals share a specific, possibly sensitive, trait. To handle this problem Machanavajjhala et al. introduced l-diversity [9]. Simply put it extends k-anonymity by also requiring that all groups are sufficiently diverse. They give a formal definition of when a dataset satisfies l-diversity, however their definition is suited to tabular data that they are working with and less suited for co-trajectories. We do not aim to give a formal definition of l-diversity for co-trajectories but will instead keep the discussions regarding diversity on an informal level. A co-trajectory has high diversity with respect to a predicate p if the trajectories in $p(\mathcal{R})$ are diverse, i.e. are not to similar to each other.

Both k-anonymity and l-diversity will come up when analysing the privacy of SwapMob in Section 4.3.

4 SwapMob

To help protecting the privacy of individuals when working with location based data many methods have been proposed in the literature. The goal is to improve the privacy while still being able to use the data for analysis. The methods vary both in the situation they are adapted to and the technique used for protecting the privacy. Some of the methods can be applied to protect privacy for a trajectory from a single individual, these include data perturbation techniques (e.g. [10, 11, 12]) and fake data generation techniques (e.g. [13, 14, 15]). Others are suited for protecting the privacy of trajectories as one of many in a co-trajectory, e.g. mix-zones [16, 17].

One recently proposed method for protecting privacy for co-trajectories is *SwapMob* [18]. The idea of SwapMob is to modify the co-trajectory by swapping segments of trajectories that cross. The goal is to make re-identification of individuals harder, but still keep enough information to allow the co-trajectory to be used for analysis. The aim of this chapter is to give a slightly different, more mathematical, definition of the method as well as perform a more thorough analysis of the method than done in the original presentation.

4.1 Definition

We begin by defining what we mean by swapping of trajectories, then continue by defining swapping on co-trajectories and give an algorithmic description of SwapMob.

Let r_1 and r_2 be two trajectories. Swapping r_1 and r_2 refers to moving datapoints between the two trajectories. More precisely, swapping r_1 and r_2 at time u gives us two new trajectories $\overline{r}_1$ and $\overline{r}_2$ given by

$$\overline{r}_1 = \{(x, t) \in r_1 : t < u\} \cup \{(x, t) \in r_2 : t \geq u\}$$

and

$$\overline{r}_2 = \{(x, t) \in r_2 : t < u\} \cup \{(x, t) \in r_1 : t \geq u\}.$$

So $\overline{r}_1$ consists of the datapoints of r_1 before time u and the ones of r_2 after time u, and reversed for $\overline{r}_2$. Figure 3 and 19 show examples of two trajectories before and after swapping.

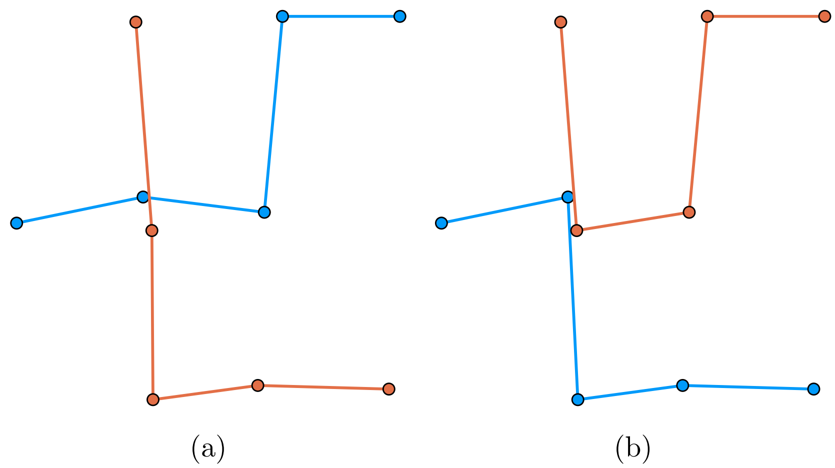


Figure 3: Two trajectories with $\mathbb{L} = \mathbb{R}^2$ are shown before swapping, to the left, and after swapping, to the right. The time of the datapoints is not explicit in this figure.

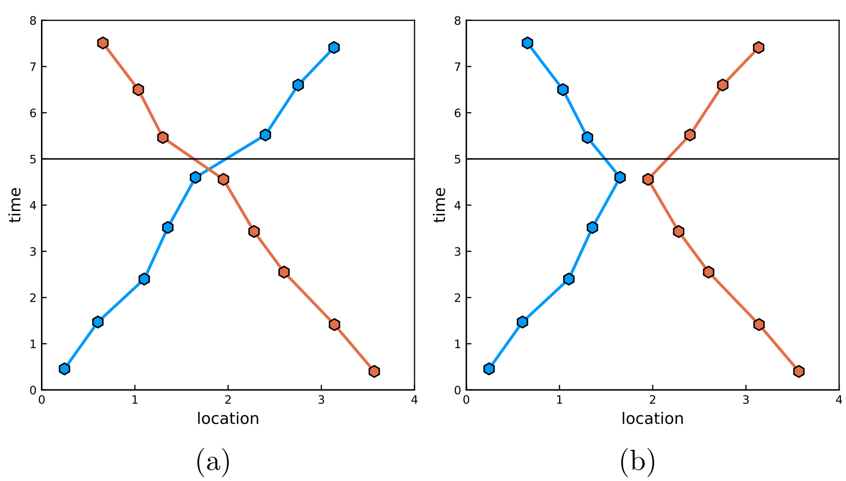


Figure 4: A trajectories with $\mathbb{L} = \mathbb{R}$ is shown before swapping in (a) and after swapping in (b). In this figure the time is explicit and follows the y-axis. The swap occurs at $u = 5$ which is shown by the black line.

While we can swap two trajectories at any time this will in general give us trajectories with big jumps in. Consider for example the case of two trajectories corresponding to two individuals moving in different parts of a city, swapping their trajectories at some time would given two trajectories that at that time suddenly jump from one part of the city the another. To avoid the problem of sudden jumps we restrict swapping to times when the two trajectories are close to each other. This is formalized by defining a *similarity* relation for datapoints. We want two datapoints, d_1 and d_2, to be similar, denoted by $d_1 \approx d_2$, if they are close in both space and time. If $\mathbb{L}$ is a metric space one definition is to let $d_1 \approx d_2$ if the distance between them in both space and time is below some given threshold. The big drawback of this approach is that it is not an equivalence relation, i.e. $d_1 \approx d_2$ and $d_2 \approx d_3$ doesn't imply $d_1 \approx d_3$. This will lead to complications when defining swaps on co-trajectories and also make an efficient implementation in code harder, we therefore choose to use another definition for similarity that does give us an equivalence relation.

We choose a partitioning of $\mathcal{D}$ and let the equivalence relation given by the partitioning be our similarity relation. We choose one partitioning for $\mathbb{R}_+$ and one for $\mathbb{L}$ and let the partitioning for $\mathcal{D} = \mathbb{L} \times \mathbb{R}_+$ be given by their product. For time we always choose the partitioning to be given by equally sized half-open intervals, given $d > 0$ we let $\mathbf{t}_i = [i \cdot d, (i+1) \cdot d)$ for $i = 0, 1, \ldots$. This gives the partitioning

$$\mathbb{R}_+ = \bigcup_{i=0}^{\infty} \mathbf{t}_i.$$

For space the natural choice of partitioning depend on the particular setting. If $\mathbb{L}$ is equal to $\mathbb{R}^2$, or a subset of it, one simple choice is to have the partitioning be given by a square grid. But if $\mathbb{L}$ represents a map a more natural choice might be to have the partitioning be given by geographical properties of the map, such as roads, rivers, mountain chains or similar. In the case of $\mathbb{L}$ being a graph the partitioning could be given by just the nodes themselves.

Given a partitioning of $\mathbb{R}_+$ and $\mathbb{L}$ we get a partitioning of $\mathcal{D}$ and equivalence relation $\approx$ for datapoints where $d_1 \approx d_2$ if they both lie in the same partition. Let r_1 and r_2 be two trajectories, we say that a swap between them is a valid swap if they have datapoints $d_1 \in r_1$ and $d_2 \in r_2$ such that $d_1 \approx d_2$ and the swap occurs right after the time for the datapoints equivalence class. More precisely, if d_1 and d_2 both belong to the equivalence class $\mathbf{x} \times \mathbf{t}$ the swap

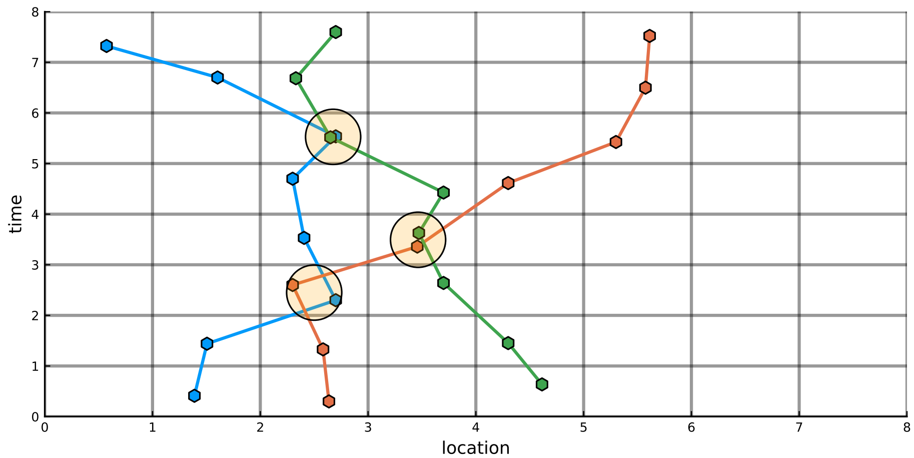


Figure 5: Three trajectories with $\mathbb{L} = \mathbb{R}$ are shown. The grid represents the partitioning of $\mathcal{D}$, where both time and space is partitioned into intervals of length 1. Datapoints which are in the same part of the grid are considered similar. For this co-trajectory there are three pairs for datapoints which are similar, denoted by the circles.

is valid if it occurs at time $u = \sup(\mathbf{t})$. In Figure 5 we see an example of a co-trajectory where we have marked datapoints which are similar.

While we do not restrict which partitioning is used for $\mathbb{L}$, the choice of it, together with the choice of d for partitioning $\mathbb{R}_+$, is very important. If two datapoints are equivalent with respect to the partitioning then swapping trajectories based on that should does not give rise to too big jumps. What constitutes a too big jump heavily depends on the setting and the type of datapoints. If we for example only have datapoints hourly then fairly big jumps are acceptable since an individual can move a lot in an hour, but if we have datapoints for every minute much smaller jumps are reasonable. In Figure 6 we can see an example of how the number of datapoints influences which jumps are reasonable.

4.1.1 Swapping for co-trajectories

We now go through the process of defining swapping for co-trajectories where there are more than two trajectories involved. First we define swapping for multiple trajectories and which swaps are valid for co-trajectories. Then we discuss how to perform several swaps on a co-trajectory and how the order

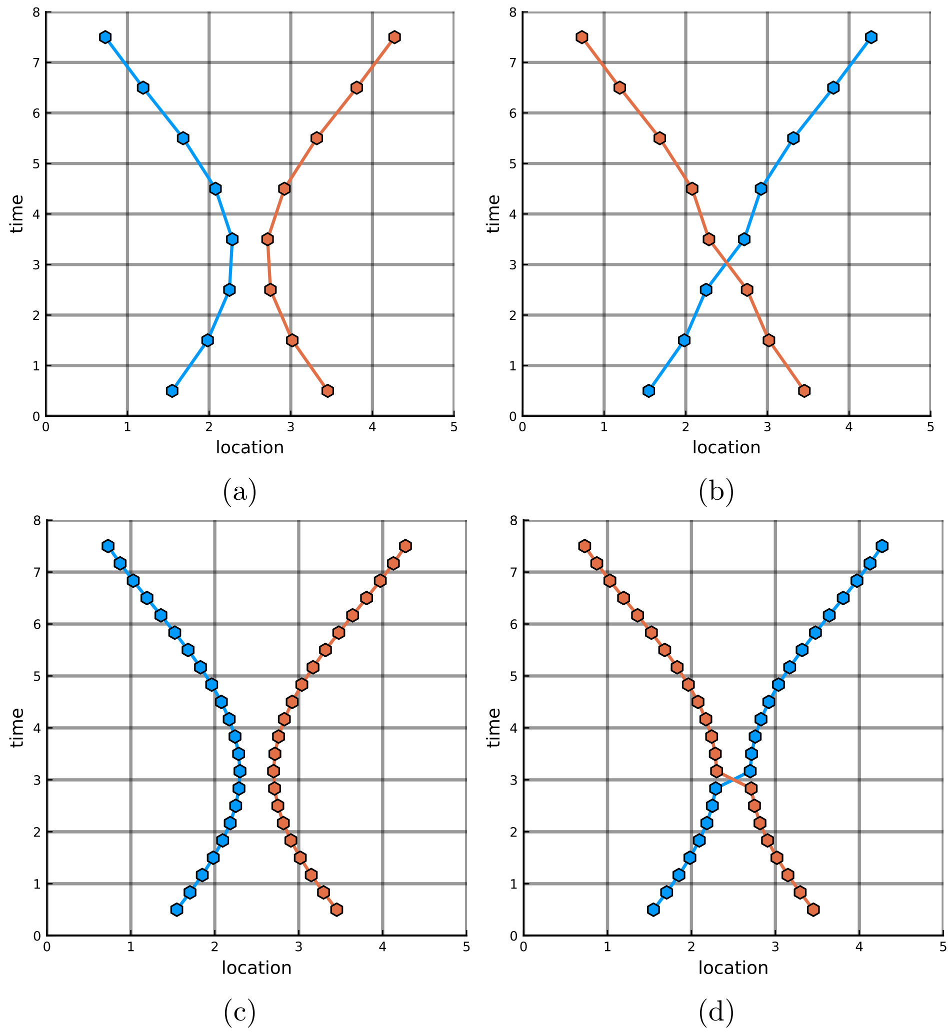


Figure 6: The number of datapoints influences which jumps are reasonable. We see the same trajectories but with different number of datapoints. The same valid swap is performed in both cases at the time $u = 3$. In (a) and (b) the swap looks reasonable and is not to obvious whereas in (c) and (d) it is very clear where the swap is performed and we have a big jump for the trajectories. This indicates that the partitioning for (c) and (d) should be made finer to avoid swaps with to big jumps.

the swaps are performed in affects the outcome.

Swaps for co-trajectories is defined similar to swaps between pairs of trajectories. Given a co-trajectory $\mathcal{R} = \{r_i\}_{i=1}^N$, a swap at time u is given by a permutation $\pi \in S_N$ and gives the co-trajectory $\overline{\mathcal{R}}$ given by $\{\overline{r}_i\}_{i=1}^N$, where

$$\overline{r}_i = \{(x,t) \in r_i : t < u\} \cup \{(x,t) \in r_{\pi(i)} : t \geq u\},\ i = 1, 2, \ldots, n. \quad (1)$$

So $\overline{r}_i$ has the datapoints from r_i before time u and from $r_{\pi(i)}$ starting from time u. A swap is uniquely defined by its time u and permutation π and we denote a swap by $s = (u, \pi)$.

For a swap $s = (u, \pi)$ of a co-trajectory to be valid we require that all the trajectories in the support of π, i.e. all r_i such that $\pi(i) \neq i$, have datapoints which are similar and that the swap occurs right after the time for the datapoints equivalence class.

Definition 4.1 (**Co-trajectory Valid Swap**)**.** A swap $s = (u, \pi)$ on a co-trajectory $\mathcal{R} = \{r_i\}_{i=1}^N$ is called a valid swap if $u = \sup(\mathbf{t})$ and there exists an equivalence class $\mathbf{x} \times \mathbf{t}$ such that for all i in the support of π there exists $d_i \in r_i$ with $d_i \in \mathbf{x} \times \mathbf{t}$ with .

Next we discuss how to perform multiple swaps on a co-trajectory. Let $\mathcal{R}_0$ be a co-trajectory and $s_1, s_2, \ldots, s_n$ be swaps on it. One way to perform all of these swaps is to do them one after another. Performing s_1 on $\mathcal{R}_0$ gives us a new co-trajectory $\mathcal{R}_1$, on this new co-trajectory we can then perform s_2 giving us $\mathcal{R}_2$. In general we get n new co-trajectories $\mathcal{R}_i$ for $1 \leq i \leq n$, where $\mathcal{R}_i$ is given by performing the swap s_i on $\mathcal{R}_{i-1}$. However, for valid swaps chaining them in this way gives rise to problems, even if $s_1, s_2, \ldots, s_n$ are all valid swaps for $\mathcal{R}_0$ we might have that s_i is not a valid swap for $\mathcal{R}_{i-1}$ on which it is performed. See Figure 7 for an example of when this is the case. When chaining the swaps we therefore have to take this into account.

To understand how to chain the swaps we have to consider at what a valid swap does. For a valid swap $s = (u, \pi$ with support I we have datapoints $d_i \in r_i$ for $i \in I$ which are all similar, i.e. in the same equivalence class. The swap s permutes datapoints between these trajectories. Datapoints from the trajectory with d_i, in this case r_i, which occur after time u goes to the trajectory having $d_{\pi^{-1}(i)}$, in this case $r_{\pi^{-1}(i)}$. We want to preserve this behaviour.

If we perform a swap $s' = (u', \pi')$ before s then d_i and $d_{\pi^{-1}(i)}$ might have moved, and if so we need to update s to take this into account. If $u \leq u'$ then neither d_i nor $d_{\pi^{-1}(i)}$ will be moved by s' (since they both occur before time u), s will still be a valid swap and we don't have to update it. If $u > u'$ then both of them will be moved, d_i will belong to the trajectory $r_{\pi'^{-1}(i)}$ and $d_{\pi^{-1}(i)}$ to the trajectory $r_{\pi'^{-1}(\pi^{-1}(i))}$. To preserve the above statement we want s to move datapoints from the trajectory containing d_i, which now is $r_{\pi'^{-1}(i)}$, to the trajectory containing $d_{\pi^{-1}(i)}$, now $r_{\pi'^{-1}(\pi^{-1}(i))}$. For this we need a permutation π'' satisfying

$$\pi''(\pi'^{-1}(\pi^{-1}(i))) = \pi'^{-1}(i).$$

For this to hold for all i we need

$$\pi'' \circ \pi'^{-1} \circ \pi^{-1} = \pi'^{-1} \iff \pi'' = \pi'^{-1} \circ \pi \circ \pi'. \tag{2}$$

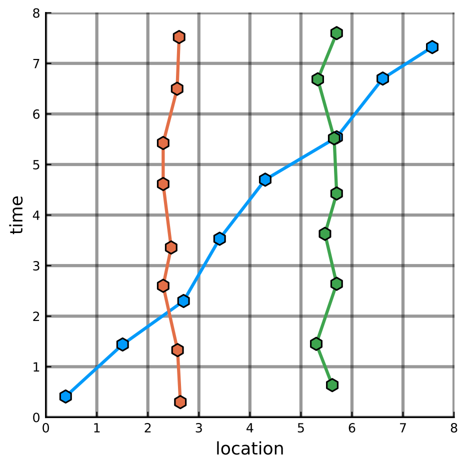
We therefore replace the permutation π for s with the permutation $\pi'' = \pi'^{-1} \circ \pi \circ \pi'$.

To summarize, if we are to perform the valid swaps $s_1, \ldots, s_n$ on the co-trajectory $\mathcal{R}_0$ we start by performing s_1 on it, giving us $\mathcal{R}_1$, then, for all swaps $s_2, \ldots, s_n$ which occur after s_1 in time we update the permutation of them to $\pi_1^{-1} \circ \pi_i \circ \pi_1$. This ensures that all the swaps are valid swaps on $\mathcal{R}_1$. We then continue by performing the updated s_2 and updating the swaps $s_3, \ldots, s_n$, we continue like this until all swaps have been performed. This ensures that all the performed swaps are valid on swaps on the co-trajectory that they are applied to. See Figure 7 for an example of this procedure.

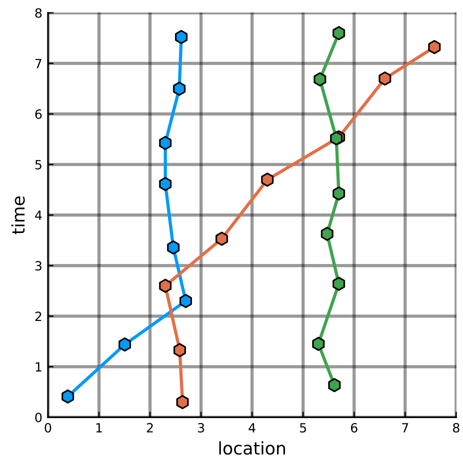
We now discuss how the order swaps are performed in affect the outcome. We show that the order of swaps occurring at different times doesn't affect the outcome but that the order of swaps occurring at the same time can affect the outcome.

Lemma 1. *Let $\mathcal{R}$ be a co-trajectory and consider two valid swaps, $s = (u, \pi)$ and $s' = (u', \pi'$, If $u \neq u'$ then swapping first with s follower by s' gives the same result as swapping first with s' followed by s.*

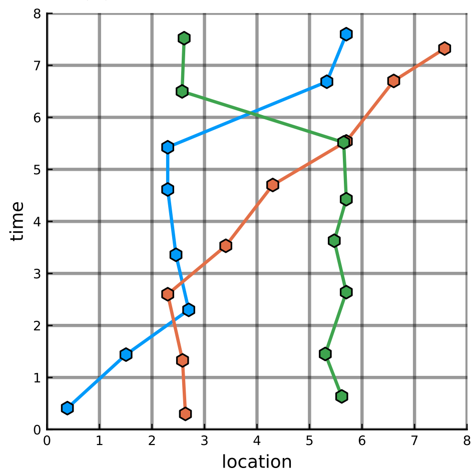
Proof. Without loss of generality assume that $u < u'$. Let $\mathcal{R}^s = \{r_i^s\}_{i=1}^N$ be the co-trajectory given by performing the swap s on $\mathcal{R}$ and $\mathcal{R}^{s'} = \{r_i^{s'}\}_{i=1}^N$ the one given by performing s'. Similarly let $\mathcal{R}^{ss'} = \{r_i^{ss'}\}_{i=1}^N$ be the co-trajectory given by performing the s' on the co-trajectory $\mathcal{R}^s$, in this case



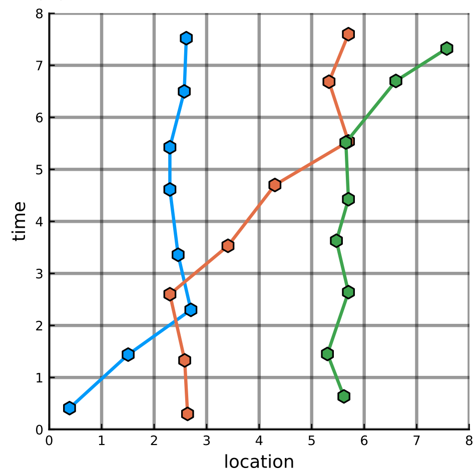
(a) Original co-trajectory



(b) Co-trajectory after first swap



(c) Co-trajectory after second swap, without updating the swap



(d) Co-trajectory after second swap, updating the swap first

Figure 7: The co-trajectory in (a) has two valid swaps, s_1 which swaps r_1 and r_2 at time $u = 3$ and s_2 which swaps r_1 and r_3 at time $u = 6$. In (b) s_1 has been performed and for this co-trajectory s_2 is no longer valid, instead s_2' which swaps r_2 and r_3 at time $u = 6$ is valid. Performing s_2 gives the result in (c) whereas performing s_2' the result in (d). If π_1, π_2 and π_2' are the permutations for s_1, s_2 and s_2' respectively we have that $\pi_2' = \pi_1^{-1} \circ \pi_2 \circ \pi_1$, as given by 2.

we also first update s' according to Equation 2, and let $\mathcal{R}^{s's} = \{r_i^{s's}\}_{i=1}^N$ be the co-trajectory given by performing s on $\mathcal{R}^{s'}$, in this case we don't need to update s since it occurs before s' in time. We have to show that $\mathcal{R}^{ss'} = \mathcal{R}^{s's}$, which we do by showing that $r_i^{ss'} = r_i^{s's}$ for all $1 \leq i \leq N$.

Let $1 \leq i \leq N$, from Equation 2 we get that the permutation for s', when performed after s, is given by $\pi^{-1} \circ \pi' \circ \pi$ which gives us that $r_i^{ss'}$ is given by

$$r_i^{ss'} = \{(x,t) \in r_i^s : t < u'\} \cup \{(x,t) \in r^s_{\pi^{-1}\circ\pi'\circ\pi(i)} : t \geq u'\}. \tag{3}$$

Here r_i^s is given by

$$r_i^s = \{(x,t) \in r_i : t < u\} \cup \{(x,t) \in r_{\pi(i)} : t \geq u\},$$

and $r^s_{\pi^{-1}\circ\pi'\circ\pi(i)}$ by

$$r^s_{\pi^{-1}\circ\pi'\circ\pi(i)} = \{(x,t) \in r_{\pi^{-1}\circ\pi'\circ\pi(i)} : t < u\} \cup \{(x,t) \in r_{\pi(\pi^{-1}\circ\pi'\circ\pi(i))} : t \geq u\}.$$

Putting this into Equation 3 gives

$$\begin{aligned} r_i^{ss'} &= \{(x,t) \in r_i^s : t < u'\} \cup \{(x,t) \in r^s_{\pi^{-1}\circ\pi'\circ\pi(i)} : t \geq u'\} \\ &= \{(x,t) \in r_i : t < u\} \cup \{(x,t) \in r_{\pi(i)} : u \leq t < u'\} \\ &\quad \cup \{(x,t) \in r_{\pi(\pi^{-1}\circ\pi'\circ\pi(i))} : t \geq u'\}. \end{aligned} \tag{4}$$

For $r_i^{s's}$ we get

$$r_i^{s's} = \{(x,t) \in r_i^{s'} : t < u\} \cup \{(x,t) \in r^{s'}_{\pi(i)} : t \geq u\} \tag{5}$$

Here $r_i^{s'}$ is given by

$$r_i^{s'} = \{(x,t) \in r_i : t < u'\} \cup \{(x,t) \in r_{\pi'(i)} : t \geq u'\},$$

and $r^{s'}_{\pi(i)}$ by

$$r^{s'}_{\pi(i)} = \{(x,t) \in r_{\pi(i)} : t < u'\} \cup \{(x,t) \in r_{\pi'(\pi(i))} : t \geq u'\}.$$

Putting this into Equation 5 gives us

$$\begin{aligned} r_i^{s's} &= \{(x,t) \in r_i^{s'} : t < u\} \cup \{(x,t) \in r^{s'}_{\pi(i)} : t \geq u\} \\ &= \{(x,t) \in r_i : t < u\} \cup \{(x,t) \in r_{\pi(i)} : u \leq t < u'\} \\ &\quad \cup \{(x,t) \in r_{\pi'(\pi(i))} : t \geq u'\}. \end{aligned} \tag{6}$$

Since $\pi(\pi^{-1} \circ \pi' \circ \pi(i)) = \pi'(\pi(i))$ we get from Equation 4 and 6 that $r_i^{ss'} = r_i^{s's}$. Since we made no assumption on i this should hold for all $1 \leq i \leq N$ and thus $\mathcal{R}^{ss'} = \mathcal{R}^{s's}$ and we conclude that the order the swaps are performed in doesn't matter. $\square$

Proposition 1. *Let $\mathcal{R}$ be a co-trajectory and consider n valid swaps, s_i $1 \leq i \leq n$, where $s_i = (u_i, \pi_i)$. If $u_i \neq u_j$ for all $i \neq j$, then the order the swaps are performed in doesn't change the result.*

Proof. Given any ordering of the swaps we can apply Lemma 1 to change the order of two swaps that are next to each other in the order without changing the final outcome. For example the ordering $s_1, s_2, \ldots, s_i, s_{i+1}, \ldots, s_n$ gives the same results as the ordering $s_1, s_2, \ldots, s_{i+1}, s_i, \ldots, s_n$ according to Lemma 1.

In turns of permutations this means that adjacent transpositions of the swaps, i.e. changing the order of two swaps next to each other, does not affect the result. Notice that we are here talking about permutations on the order of the swaps, not about the permutations the swaps perform. Since the set of all permutations of the swaps, S_n, is generated by the set of adjacent transpositions we can get to any permutations of the swaps by iteratively applying adjacent transpositions. As non of the adjacent transpositions affect the result we get that the result of the final permutation is the same as the original one. We conclude that for all orderings of the swaps the results are the same. $\square$

The above proposition shows that the order of swaps that occur at different times doesn't matter. For swaps occurring at the same time the order does matter, an example of this is seen in Figure 8. In general we get the following result.

Proposition 2. *Let $\mathcal{R}$ be a co-trajectory and consider two valid swaps, $s = (u, \pi)$ and $s' = (u', \pi')$. If $u = u'$ then performing the swap s followed by s' gives the same result as performing a single swap, $s \circ s' = (u, \pi \circ \pi'$.*

Proof. Let $\mathcal{R}^s = \{r_i^s\}_{i=1}^N$ be the co-trajectory given by performing the swap s on $\mathcal{R}$ and $\mathcal{R}^{ss'} = \{r_i^{ss'}\}_{i=1}^N$ be the co-trajectory given by performing swap the s' on the co-trajectory $\mathcal{R}^s$. Notice that we do not need to update the permutation for s' since it occurs at the same time as s. Also let $\mathcal{R}^{(s \circ s')} =$

$\{r_i^{(s \circ s')}\}_{i=1}^N$ be the co-trajectory given by performing the swap $s \circ s'$ on $\mathcal{R}$. We show that $r_i^{ss'} = r_i^{(s \circ s')}$ for all $1 \leq i \leq N$.

For all $1 \leq i \leq N$ we have that

$$r_i^s = \{(x,t) \in r_i : t < u\} \cup \{(x,t) \in r_{\pi(i)} : t \geq u\}.$$

and for $r_i^{ss'}$ this gives us

$$\begin{aligned} r_i^{ss'} &= \{(x,t) \in r_i^s : t < u\} \cup \{(x,t) \in r_{\pi'(i)}^s : t \geq u\} \\ &= \{(x,t) \in r_i : t < u\} \cup \{(x,t) \in r_{\pi'(\pi(i))} : t \geq u\}. \end{aligned}$$

For $r_i^{(s \circ s')}$ we get

$$r_i^{(s \circ s')} = \{(x,t) \in r_i : t < u\} \cup \{(x,t) \in r_{(\pi \circ \pi')(i)} : t \geq u\}.$$

Since $(\pi \circ \pi')(i) = \pi(\pi'(i))$ we get that $r^{ss'} = r^{(s \circ s')}$. It follows that $\mathcal{R}^{ss'} = \mathcal{R}^{(s \circ s')}$. $\square$

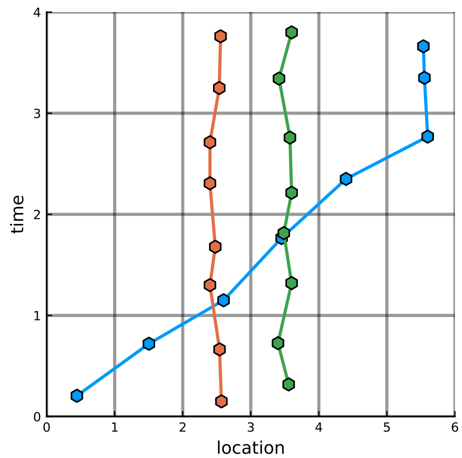
A consequence of this is that the order of the swaps doesn't matter as long as the permutations for them commute with each other. One particular case of this is when the supports for all the permutations are disjoint, since then the permutations commute. In Figure 8 the permutations for the swaps does not commute and the order they are applied in does therefore affect the result.

How the order of swaps affect the outcome will have implications when choosing which swaps to perform when using SwapMob, for details of this see the Section 4.2 where the algorithm for SwapMob is discussed.

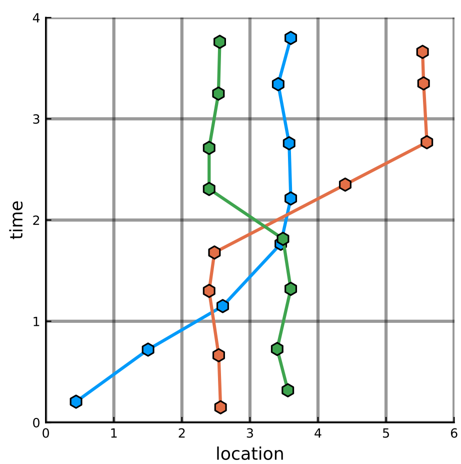
4.1.2 Comparison to mix-zones

Of the methods mentioned in the beginning of Section 4 for protecting privacy when working with trajectories and co-trajectories , the one that is most similar to SwapMob is mix-zones [16, 17].

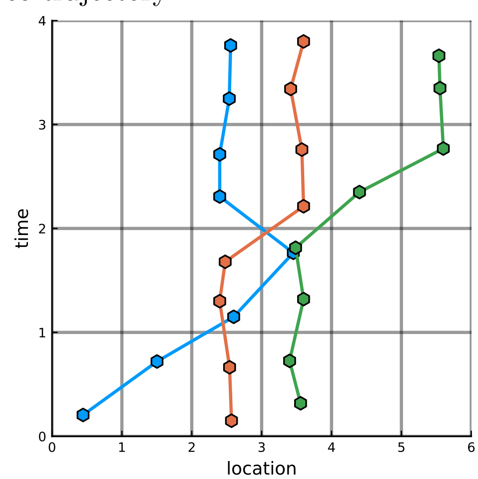
In the context of mix-zones every individual is associated with a pseudonym, the mix-zones are certain areas for which the individual change pseudonym when they enter. With our definitions of trajectories and co-trajectories this corresponds to the individual being associated with two different trajectories, one before entering the zone and one after. This makes it harder to track the



(a) Original co-trajectory



(b) Swapping first with s and then with s'



(c) Swapping first with s' and then with s

Figure 8: The co-trajectory in (a) has two valid swaps, s which swaps r_1 and r_2 at time $u = 2$ and s' which swaps r_1 and r_3 at the same time $u = 2$. In (b) we see the result of first performing s and then s' whereas in (c) we see the result of performing them in the opposite direction.

long term movement of individuals since the data gets spread out between several trajectories whenever they enter the mix-zones and it is hard to know which trajectories belong to the same individual. For this to work properly no data should be collected while the individual is inside the mix-zone, otherwise it is easy to find which trajectories belong together by looking at there first and final datapoint. In addition there should be several individuals in the mix-zone at the same time and it should be hard to correlate where and when you go into the mix-zone with where and when you go out.

SwapMob and mix-zones are similar in that they both split the datapoints of an individual among several trajectories to make it harder to link them all together. For mix-zones this is done when the individual enters a mix-zone and for SwapMob when two individuals come close enough to perform a swap. This makes SwapMob more dynamic than mix-zones, since it allows swapping anywhere whereas mix-zones are more static.

For mix-zones no data should be collected while the individual is inside the mix-zone, this improves the privacy since less data is collected but also limits the usability of the data since no analysis about what happens inside the mix-zone can be done.

For the individual the privacy gained from mix-zones could be more clear. They could know how often they enter a mix-zone, and thus how many times there trajectory is split up. Whereas for SwapMob it can be hard for the individual to know how many swaps they participate it. To know if you enter a mix-zone only depends on your own trajectory whereas to know if you participate in a swap you have know both you own trajectory and the trajectories of others. However for mix-zones you might not know how many other people are in the mix-zone at the same time, and thus how many other people you are mixed with.

To summarize, mix-zones and SwapMob are similar but with some key differences. Mix-zones prohibits data collection in some areas whereas SwapMob does not by itself prohibit data collection anywhere. Mix-zones work with static areas for mixing trajectories whereas SwapMob dynamically finds appropriate places for swapping trajectories.

4.2 Algorithm

In this section we give the algorithm for SwapMob, the main focus is on the mathematical properties but some implementation details will also be discussed. The algorithm is divided into two parts, the first is about determining which swaps to perform and the second about how to perform these swaps. Of these two parts the former is the more interesting one and the one that will play a role in Section 4.3 when analysing the algorithm. We do however start by giving the algorithm for performing the swaps, since some of the remarks there have implications for which swaps we choose to include when determining the swaps to use.

4.2.1 Performing swaps

The algorithm for performing swaps will take as input a co-trajectory and a list of swaps to perform and return a new co-trajectory with all the swaps performed.

We begin by giving an algorithm for performing a single swap on a co-trajectory. The final algorithm will then be given by iterating this for each swap to perform. To perform a swap we simply take the original co-trajectory and for each trajectory in the support of the swap we replace it with the resulting trajectory according to Equation 1.

Algorithm 1 Algorithm for performing a single swap on a co-trajectory

procedure SWAPSINGLE($\mathcal{R}, s = (u,\ \pi)$) ▷ The co-trajectory resulting from performing the swap s on the co-trajectory $\mathcal{R}$
  $\mathcal{R}' \leftarrow \mathcal{R}$
  $I \leftarrow support(\pi)$
  for $i \in I$ **do**
    $part_1 \leftarrow \{(x,t) : (x,t) \in \mathcal{R}[i], t < u\}$
    $part_2 \leftarrow \{(x,t) : (x,t) \in \mathcal{R}[\pi(i)], t \geq u\}$
    $\mathcal{R}'[i] \leftarrow part_1 \cup part_2$
  end for
  return $\mathcal{R}'$
end procedure

When iterating this algorithm for all the swaps in the list we have to take care about which order we perform the swaps in. According to Proposition 1 the result does not depend on the ordering of swaps with different timestamps, the computational cost of computing the result does however depend on the order, when performing a swap we have to update all remaining swaps with a timestamp occurring later than the current swap according to Equation 2. By ordering the swaps so that the timestamps are decreasing we avoid having to update any swaps, because when performing a swap all remaining swaps have timestamps which are occurring at the same time or before the one for the current swap, and thus doesn't have to be updated.

Ordering by timestamp does however not solve the problem of how to order swaps with the same timestamp. One option is to use a random order, however, as discussed above and exemplified in Figure 8, the order does affect the result. This would therefore introduce randomness in the swapping procedure, which we would prefer to avoid. Another option is to use a pre-specified order for them, but there is no natural choice to use for the ordering. To handle this we begin by noticing that if the permutations for all the swaps occurring at a specific timestamp all commute, then the order they are performed in does not affect the outcome as per Proposition 2. In this case we can choose a random order to perform them in without it affecting the result. In particular, if the supports of the swaps occurring at the same time are all disjoint then they commute. One solution is therefore to require that the list of swaps, returned by the algorithm that determines the swaps to perform, satisfies that every trajectory participates in at most one swap for each timestamp, this ensures that swaps occurring at the same time have disjoint support and thus commute. By requiring this we are ensured that the order the swaps are performed in does not matter.

The final algorithm begins by sorting the list of swaps so that the timestamps are decreasing. It then takes the swaps one by one and perform them on the co-trajectory using Algorithm 1. This is seen in Algorithm 2.

4.2.2 Determining swaps

The algorithm for determining which swaps to perform takes as an argument a co-trajectory and returns a list of swaps to be performed. It is the composition of one deterministic and one randomized algorithm. The first algorithm returns a list of groups of trajectories according to their datapoints equivalence classes and the second algorithm then chooses a random permutation to use for each group.

Algorithm 2 Algorithm for performing several swaps on a co-trajectory

procedure SWAP($\mathcal{R}$, $swapList$) ▷ The co-trajectory resulting from performing all the swaps in $swapList$ on the co-trajectory $\mathcal{R}$
  Sort $swapList$ **By decreasing timestamps**
  for $s \in swapList$ **do**
    $\mathcal{R} \leftarrow swapSingle(\mathcal{R}, s)$
  end for
  return $\mathcal{R}$
end procedure

The deterministic algorithm takes as input a co-trajectory, $\mathcal{R}$, it groups the trajectories according to their datapoints equivalent classes and returns these groups together with the supremum time for the respective equivalence class. More precisely it returns a list of elements on the form (u, I), where I is a (multi-)set of trajectories satisfying that there is an equivalence class $\mathbf{x} \times \mathbf{t}$ such that for all $r_i \in I$ there exists $d \in r_i$ with $d \in \mathbf{t} \times \mathbf{x}$ and further $u = \sup(\mathbf{t})$. This ensures that any swap that permutes the trajectories in I which occur at time u is a valid swap.

From Algorithm 2 we have the requirement that no trajectory should participate in several swaps at the same time. This means that for two returned elements (u, I) and (u', I') we should have $u = u' \implies I \cap I' = \emptyset$. One simple way to satisfy this is to only consider the last datapoint in every time interval of a trajectory when finding valid swaps. This can be accomplished by removing all the other datapoints from the trajectories before finding the groups. Notice that since the definition of a trajectory only allow for a finite number of datapoints in every finite time interval we can be certain that there is a well defined last datapoint in every interval.

The algorithm is given in Algorithm 3. It begins by for every trajectory only keeping the last datapoint in every time interval. It then collects a set of all remaining datapoints in the co-trajectory, together with the id of the trajectory they belong to. This set is then grouped according to the equivalence class of the datapoints, giving a set where each element is given by an equivalence class together with a set of ids corresponding to datapoints that occurred in that equivalence class. We discard any equivalence classes having only one id belonging to it, since no swapping can be performed with just one trajectory. We then return the groups of ids together with the time to swap them, i.e. supremum time of their equivalence class.

Algorithm 4 Algorithm for computing groups of trajectories which have datapoints in the same equivalence class.

---

procedure GROUPTRAJECTORIES($\mathcal{R}$) $\triangleright$ Groups of trajectories grouped according to equivalence classes of their datapoints

$\mathcal{R}' \leftarrow \{\text{filter}(r_i) : r_i \in \mathcal{R}\}$

$datapoints \leftarrow \{(i,\ d) : \forall d \in r_i, \forall r_i \in \mathcal{R}'\}$

$groups_1 \leftarrow \{(\mathbf{x} \times \mathbf{t}, \{i : (i, d) \in datapoints, d \in \mathbf{x} \times \mathbf{t}\}) : \forall \mathbf{x} \times \mathbf{t}\}$

$groups_2 \leftarrow \{(\mathbf{x} \times \mathbf{t}, I) \in groups_1 : |I| > 1\}$

return $\{(\sup(\mathbf{t}),\ I) : (\mathbf{t} \times \mathbf{x},\ I) \in groups_2\}$

end procedure

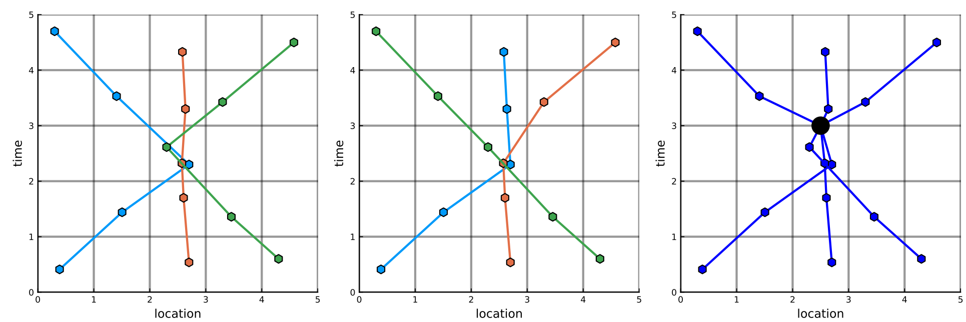
The second, randomized, algorithm associates each element in the list returned by the first algorithm with a permutation, which, together with the time, yields a swap. For the element $(u,\ I)$ the permutation is chosen uniformly at random from all permutations $\pi \in S_{|\mathcal{R}|}$ whose support is a subset of I, this ensures that the resulting swap, $s = (u, \pi)$, is a valid swap.

Composing Algorithm 3 with this randomized algorithm gives us an algorithm which takes a co-trajectory and returns a list of valid swaps. This list of swaps can then be used as an argument for Algorithm 2 to perform the swapping.

4.3 Analysis

Having given the precise algorithm that SwapMob uses we continue with analysing its properties. We analyse both what type of privacy the swapping gives, what an adversary can learn from analysing the swapped data, and what data is preserved under the swapping and how it can be used in analysis.

Applying SwapMob to a co-trajectory, $\mathcal{R}$, gives as result a new co-trajectory, $\mathcal{R}'$. The naive way to analyse $\mathcal{R}'$ would be to see it as any other co-trajectory and use the same tools for analysis. However, this would give us results for $\mathcal{R}'$ and not $\mathcal{R}$, which is the co-trajectory we are interested in. We need a way to take into account that SwapMob has been used to produce $\mathcal{R}'$ and use this to get information about $\mathcal{R}$. The main tool for this will be a representation of the swapped data as a directed graph which encodes the effect of SwapMob.

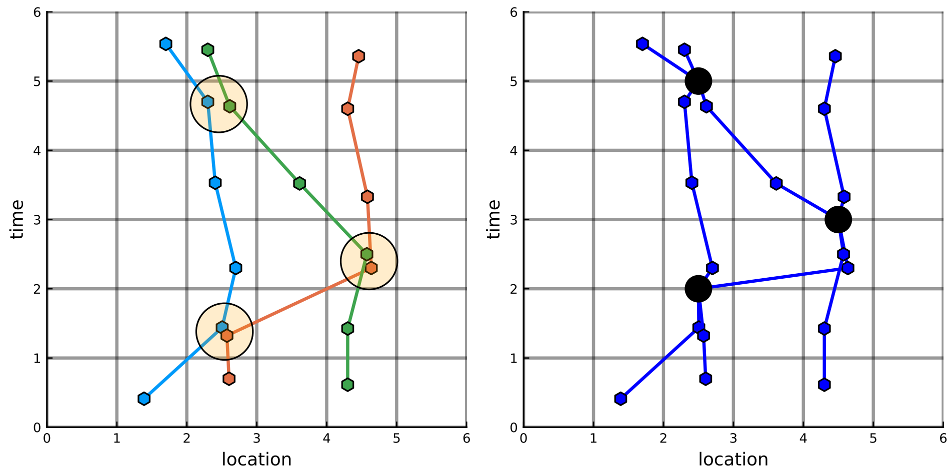


(a) Original co-trajectory (b) Co-trajectory after swapping (c) Co-trajectory with the swap as a vertex

Figure 9: The co-trajectory in (a) has a valid swap at time $u = 3$. In (b) the co-trajectory resulting from performing the swap with the permutation π given by $\pi(1) = 2$, $\pi(2) = 3$, $\pi(3) = 1$ is shown. Finally (c) shows the result of adding a new vertex to represent the swap. Notice that we in (c) no longer use different colors for the different trajectories, this is to highlight that this representation only keeps track of the topology of the co-trajectory and not the original trajectories.

SwapMob as a graph A trajectory can be represented as a directed acyclic graph (DAG) by seeing the datapoints as vertices and putting edges between successive datapoints, in fact this is how we have been visualizing trajectories in most figures so far. Similarly a co-trajectory can be represented by a DAG that is the union of DAG:s of trajectories. Swapping trajectories corresponds to swapping the head of the edges. In Figure 9a we see a co-trajectory and we can note that the representation correspond to a DAG, in 9b a swap has been performed and we can see that this correspond to changing the heads of three edges. Instead of swapping with some specified permutation we can encode the swap by creating a new vertex, a swapping vertex, redirecting the heads to this vertex and creating new edges from this vertex to the previous positions of the heads, the result of this is seen in Figure 9c. Notice that the resulting graph is also a DAG.

If we have a number of swaps on a co-trajectory we can create swapping vertices for all of them in this way. This is well defined as long as no trajectory participate in several swaps at the same timestamp. In fact we don't even need the permutations from the swaps, only the time it should occur and which trajectories should participate. This means that we can use the output



(a) Original co-trajectory (b) DAG representation of swapping

Figure 10: The co-trajectory in (a) has three valid swaps. In (b) all of these are represented by adding vertices to the graph, similarly to Figure 9c the trajectories are no longer shown with different colors.

from Algorithm 3 to generate the graph, Figure 10 shows an example of this.

For a co-trajectory $\mathcal{R}$ we denote by $G(\mathcal{R})$ the graph representation of the co-trajectory, such as the one seen in Figure 10a, and by $G_s(\mathcal{R})$ the graph representation with vertices for the swaps added. When the $\mathcal{R}$) is clear from the context we sometimes drop it and use only G and G_s.

One of the important properties G_s is that it is invariant under SwapMob, i.e. for a co-trajectory $\mathcal{R}$ and a co-trajectory $\mathcal{R}'$ given by applying SwapMob to $\mathcal{R}$ the graphs $G_s(\mathcal{R})$ and $G_s(\mathcal{R}')$ are identical. To see this consider the graph representations of the two co-trajectories, $G(\mathcal{R})$ and $G(\mathcal{R}')$, they are identical except that the heads for the edges where swaps occur are moved (as in Figure 9a and 9b). The graphs $G_s(\mathcal{R})$ and $G_s(\mathcal{R}')$ will thus be the same since the head of the edges that are moved in $G(\mathcal{R}')$ compared to $G(\mathcal{R})$ will be moved to the same new swap vertices anyway, meaning that the difference between $G(\mathcal{R})$ and $G(\mathcal{R}')$ will not affect the G_s.

Every trajectory in $\mathcal{R}$ will correspond to paths in $G_s(\mathcal{R})$. The opposite is not true, every path in $G_s(\mathcal{R})$ will not correspond to a trajectory in $\mathcal{R}$, however it will correspond to a possible trajectory after swapping. A path makes a

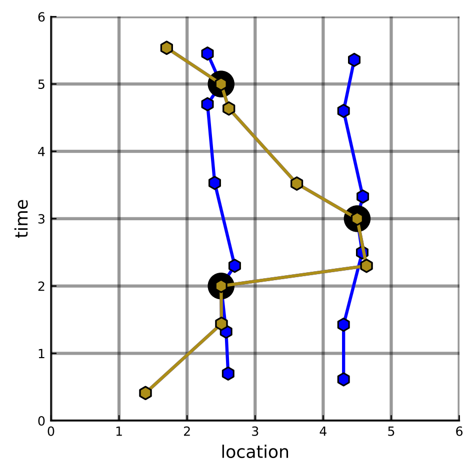


Figure 11: One possible path in Figure 10b is highlighted in yellow. This particular path does not appear in the original co-trajectory from 10a but could exist as a result of applying SwapMob if the right permutations are chosen.

"choice" at every swapping node as to which edge to follow, by choosing a permutation for each swapping node that agrees with this choice we get as a result of the swapping a co-trajectory with this trajectory in it. An example of this is seen in Figure 11. By considering the set of all paths in $G_s(\mathcal{R})$ we can get the set of all possible trajectories which we can get by swapping, in particular all trajectories in the original co-trajectory are in this set.

4.3.1 Privacy

Let $\mathcal{R}$ be a co-trajectory and $\mathcal{R}'$ the resulting co-trajectory from applying SwapMob to it. In this section we are concerned about what data can be extracted about individuals in $\mathcal{R}$ by an adversary having access to $\mathcal{R}'$. The main observation is that since G_s is invariant under SwapMob the adversary will by having access to only $\mathcal{R}'$ still be able to compute $G_s(\mathcal{R})$ since it is the same as $G_s(\mathcal{R}')$. By analysing $G_s(\mathcal{R})$ it can gain information about trajectories in $\mathcal{R}$.

We consider the case of the adversary having access to different kinds of partial data about an individual and what more it can learn about the individual using the data from $\mathcal{R}'$.

As discussed in Section 3 one way to model the partial data the adversary has is by considering a predicate, p, on trajectories, which returns true for trajectories satisfying the partial data and false for those that don't. If the adversary had access to the original co-trajectory, $\mathcal{R}$, it would be enough to use the predicate on all trajectories in it to get the possible trajectories for the individual. However the adversary only have access to $\mathcal{R}'$ and must use this data together with the knowledge that it was generated by applying SwapMob to the original data. The original trajectory for the individual will most likely not be in $\mathcal{R}'$, this is the case as long as it participated in at least one swap. However, the original trajectory will appear as a path in $G_s(\mathcal{R})$, which the adversary has access to. The adversary can thus enumerate all paths in the graph and use the predicate on all of them.

In more formal terms, let P denote the set of trajectories corresponding to the set of all paths in $G_s(\mathcal{R})$, this corresponds to the set of all trajectories we can get by swapping, and as discussed above this set does in particular contain all trajectories of the original co-trajectory $\mathcal{R}$. The adversary can apply the predicate p to the trajectories in P to get the set of all trajectories that could belong to the individual. We denote by $p(P)$ the set of all trajectories in P that satisfy the predicate p. The adversary knows that the original trajectory is in this set.

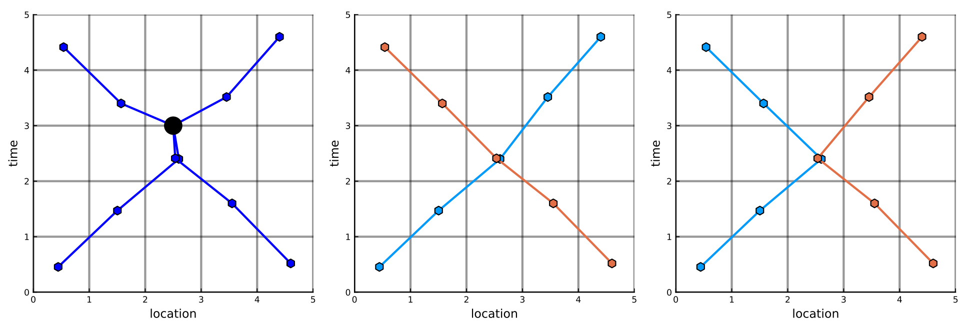
How much privacy SwapMob gives with respect to a specific predicate can then be measured by comparing the two sets $p(\mathcal{R})$, the information gained if SwapMob is not used, and $p(P)$, the information gained if SwapMob is used. The simplest comparison is to look at the number of trajectories in the two sets, this corresponds to the k-anonymity introduced in Section 3. The larger that $|p(P))|$ is compared to $|p(\mathcal{R})|$ the more privacy is gained through SwapMob. How much it increases will depend on the number of swaps for the co-trajectory as well as the predicate.

Only comparing the number of trajectories does however not take into account one of the important differences between $\mathcal{R}$ and P. In $\mathcal{R}$ all the trajectories correspond to different individuals and do in general not share datapoints. For P this is not the case, here a lot of the trajectories will share datapoints since they are constructed by concatenating parts of the trajectories from $\mathcal{R}$. For example we can see that the outlined trajectory in 11 shares datapoints will all the original trajectories seen in Figure 10a. Therefore even if the k-anonymity is increased a lot when comparing $p(\mathcal{R})$

and $p(P)$ the diversity between the two sets might not, this corresponds to the l-diversity not increasing a lot. The diversity of $p(P)$ could be measured in different ways, one simple way would be to look at the intersection of all the trajectories to find datapoints that occur in all of them and thus for sure belong to the individual. As mentioned in Section 3.2 we do not attempt to given a formal definition for measuring diversity and keep the discussions about it on an informal level.

Another aspect that should be taken into account is the probability for each permutation at a swap. For example consider the adversary having access to the graph in Figure 12a. The adversary does not know the original co-trajectory, but knows that it was either the one in Figure 12b or 12c. In Figure 12b the two trajectories both go in straight paths in opposite directions whereas in Figure 12c they go in opposite directions and when they meet they both perform a complete U-turn. In this scenario an adversary might consider the first case more likely, that both trajectories continue on the way they were on. It might seem less likely that both trajectories do a U-turn at the same time. Another example would be when number of datapoints and the partitioning doesn't align well, as in Figure 6, in which case the adversary can also see that one permutation is more likely than the other. In the general case the adversary can use any outside information to obtain a probability distribution for the permutations at a swap node, with no information you get the uniform distribution but with more information you could find permutations which are more likely than others. In the extreme case the adversary is able to determine with a very high probability which permutation was performed at a swap node. This would mean that for a predicate p all paths in $p(P$ are not equally likely, some might be much more likely and other much less. In the analysis below this is not taken into account, indirectly we are assuming that the adversary cannot infer anything about which permutations are more likely.

We now look at some examples. The first example is an example co-trajectory which we use to exemplify the above discussion, we look at a few different kinds of predicates and how they behave with respect to SwapMob. The second example uses a real world dataset, T-drive, from taxis in Beijing [19, 20] and gives an idea of how SwapMob might perform on real data.

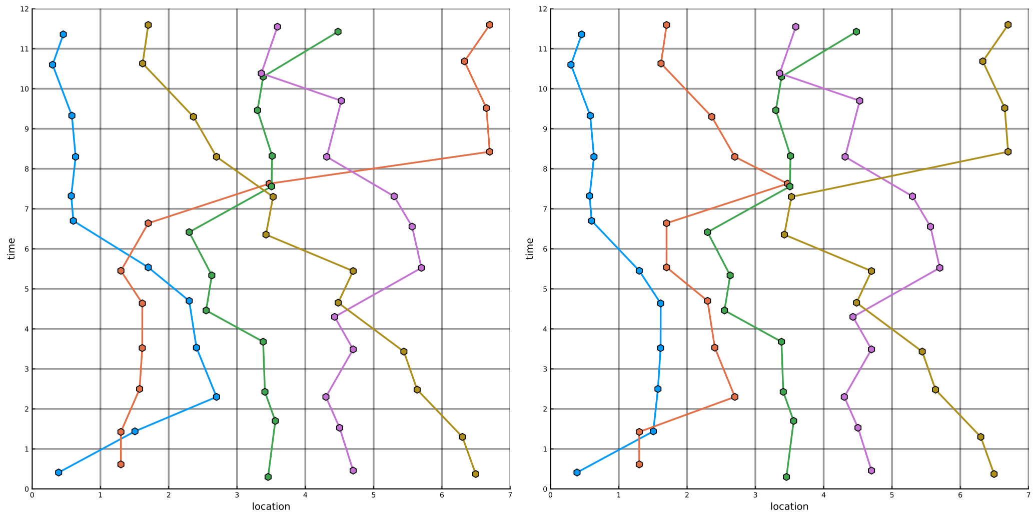


(a) (b) (c)

Figure 12: An adversary having access to the graph in (a) knows that the original co-trajectory was either the one in (b) or the one in (c). The co-trajectory in (b) seems more likely since it corresponds to the two trajectories continuing on their current path, whereas (c) would mean that they both perform a U-turn at the same time.

Example co-trajectory The co-trajectory $\mathcal{R}$ used in this part is the one given in Figure 13a. The co-trajectory consists of five trajectories all consisting of 12 datapoints, it has six places where a swap can occur and in Figure 13b one possible outcome of applying SwapMob to it is shown. In Figure 14a the graph representation $G_s(\mathcal{R}$ is shown, the locations of the six swaps are clear. Let P denote the set of trajectories corresponding to paths in the graph, the number of such trajectories can be checked to be 49.

We begin by considering predicates given by knowing one datapoint of the trajectory exactly. Assuming we know the datapoint d this gives us the predicate p_d which is true for trajectories r satisfying $d \in r$ and false otherwise. Applying p_d to $\mathcal{R}$, directly gives us the trajectory that d belongs to since for this co-trajectory no trajectories have any identical datapoints. If we instead only have access to the swapped data we have to apply it not to $\mathcal{R}$ but to P. In this case $p_d(P)$ will not be unique but give us all trajectories that pass through the given datapoint. For example if d is the datapoint marked in red in Figure 14b, then $p_d(P)$ includes 12 trajectories, marked by yellow in Figure 14b. For this example SwapMob does increase the privacy since $p_d(P)$ contains many more trajectories than $p_d(\mathcal{R})$ and we do not learn exact trajectory of the individual. In terms of k-anonymity we have that the co-trajectory satisfies k-anonymity with respect to the predicate p_d with $k = 12$.



(a) Original co-trajectory

(b) One possible co-trajectory after applying SwapMob.

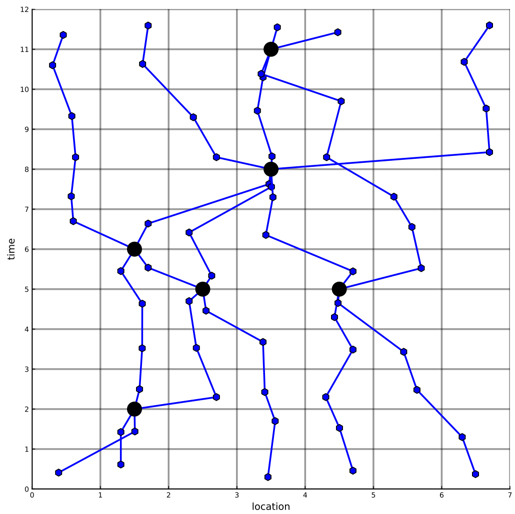
Figure 13

While the privacy in terms of k-anonymity might be good the diversity of the set $p_d(P)$ is not as high. Some datapoints are shared by all trajectories in $p_d(P)$ (marked green in Figure 14b) and we therefore know for sure that the individual has these datapoints in its trajectory. If we only consider what the trajectory looks like after the datapoint d then there are only four different possibilities, if we care only about the part before there are three. All of this shows that while $p_d(P)$ does contain many trajectories the diversity of the trajectories is not as high.

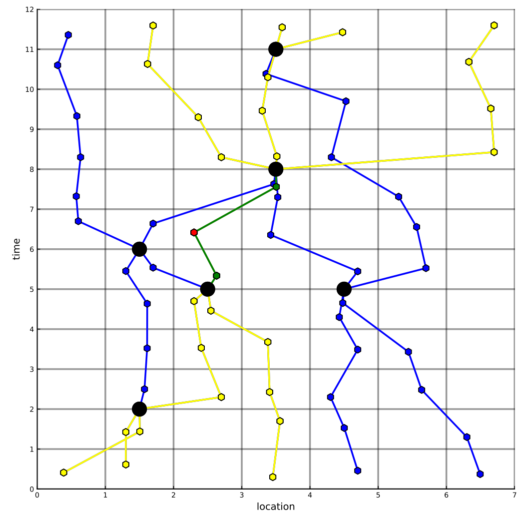
We can also look at the family of predicates given by knowing one datapoint exactly, i.e. the family of predicates $\{p_d\}_{d \in M}$, where M is the set of all datapoints occurring in the co-trajectory and p_d is defined as above. In Figure 15 the distribution of $|p_d(P)|$ for $d \in M$ is shown. We see that this family of predicates satisfy k-anonymity with $k = 4$.

Next we consider predicates given by knowing the first and last datapoint of the individuals trajectory, this gives five predicates, $p_1, p_2, \ldots, p_5$, each corresponding to one of the original trajectories. Again we have that $p_i(\mathcal{R})$ contains the unique trajectory for all $1 \leq i \leq 5$. For $p_i(P)$ we get

$$|p_1(P)| = 2, \ |p_2(P)| = 3, \ |p_3(P)| = 2, \ |p_4(P)| = 2 \text{ and } |p_5(P)| = 1.$$



(a) Representation of the co-trajectory from Figure 13a as a DAG.



(b) One datapoint d is marked in red. The yellow and green lines marks all trajectories having d as a datapoint. The green part corresponds to datapoints which are common for all those trajectories whereas the yellow part corresponds to datapoints which are included in only some of them. In total there are 12 different trajectories passing through d.

Figure 14

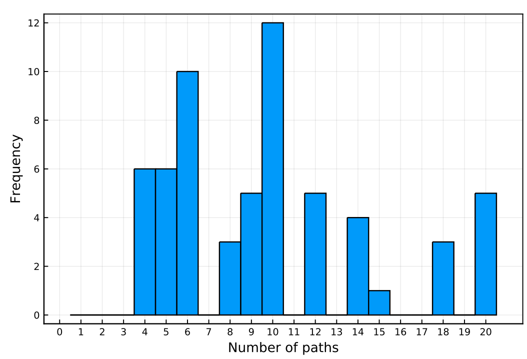


Figure 15: Frequency histogram of $|p_d(P)|$ for all datapoints in the $\mathcal{R}$.

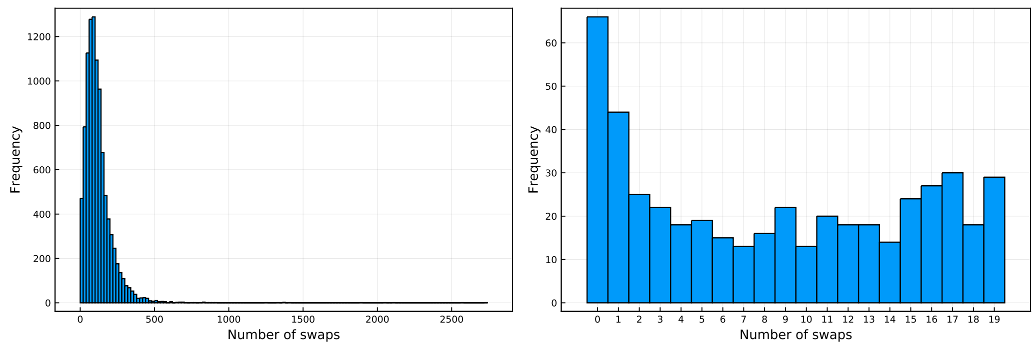
Here the privacy gain in terms of k-anonymity is worse, for this family of predicates it does not even satisfy k-anonymity with $k = 2$. The diversity is also low, for p_1 where the k-anonymity is the highest all three trajectories in $p_1(P)$ are very similar.

T-drive co-trajectory The co-trajectory used in this part is based on data from the T-drive dataset [20, 19]. The dataset contains GPS trajectories of 10,357 taxis during the period February 2 to February 8, 2008 in Beijing and consists of about 15 million datapoints.

The datapoints are given by a time together with a position measured in longitude and latitude. The original data is not clean and does for example contain many datapoints for which the longitude and latitude is equal to zero. Therefore we begin by pre-processing the data to remove these sort of anomalies. To begin with we keep only datapoints for which the longitude and latitude is inside the box $[115, 117] \times [39, 41]$, any point outside this box is definitely outside of Beijing. In addition we discard trajectories with fewer than 10 datapoints. This gives us a co-trajectory consisting of 9926 trajectories and a total of 15,650,074 datapoints.

For choosing the partitioning to use with SwapMob the sampling rate has to be taken into account to avoid problems like those in Figure 6. The average sampling interval for the trajectories is about 180 seconds, based on this we choose to partition the time into intervals of 60 seconds. The average distance between to consecutive points is about 620 meters, a partitioning into a grid of width 100 meters should then be reasonable. The location is however not given in terms of meters but in longitude and latitude. In Beijing 100 meters corresponds to about 0.001 degrees and we can use that for partitioning. It should be noted that since the earth is a sphere a uniform grid in terms of longitude and latitude does not correspond to a uniform grid on the sphere, in our case the precise partitioning is not important and a uniform grid in longitude and latitude works well.

With the above partitioning the number of possible swaps, as returned by Algorithm 3, is 540,812. The average number of swaps for each trajectory is 119 and the distribution of swaps is seen in Figure 16a. Most trajectories participate in less than 500 swaps but a few participate in thousands. Of the 471 trajectories that participate in less than 20 swaps the distribution is seen in Figure 16b, 66 of them do not participate in any swap.



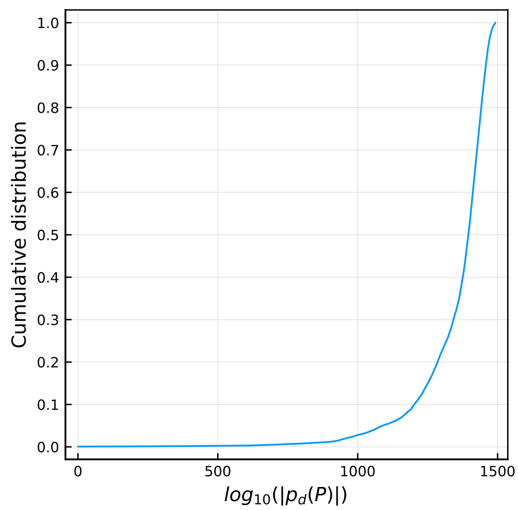
(a) Frequency histogram of number of swaps for all the trajectories in the T-drive co-trajectory. The bars in the figure has a width of 20.

(b) Frequency histogram of number of swaps for all the 471 trajectories participating in less than 20 swaps. Of these 66 are seen to not participate in any swap.

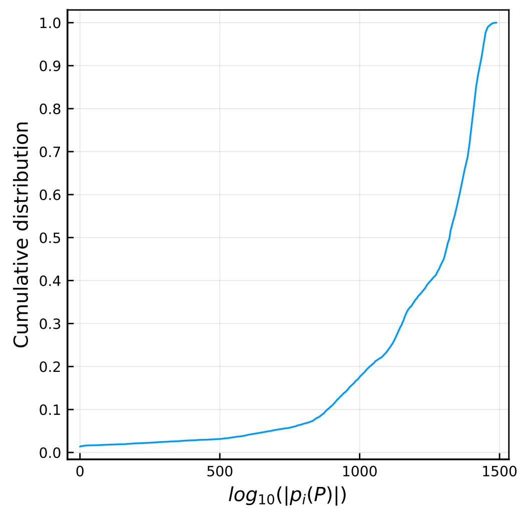
The corresponding graph, *swapgraph*, is very big and computing the total number of possible paths gives a result slightly greater than $6.7 \cdot 10^{1491}$. From this we can conclude that a naïve brute force approach for computing $p(P$ for a predicate p would not be feasible, iterating through all possible paths and checking the predicate is not possible. We therefore have to restrict our attention to predicates for which $p(P)$, or at least $|p(P)|$, can be computed in an efficient way.

Similar to the last example we begin by considering predicates given by knowing one datapoint of a trajectory exactly. Assuming we know the datapoint d we get the predicate p_d. With more than 15 million datapoints in total we are not able to compute $|p_d(P)|$ for every d in the co-trajectory, instead we take a random sample of around 5%, about 780,000, of the datapoints and compute it for these. The result ranges from 1, for trajectories with no swaps, to around 10^{2928}. The cumulative distribution of $\log_{10}(|p_d(P)|)$ is shown in Figure 17a. As seen in the figure the k-anonymity is very big for the vast majority of datapoints, only 1884 datapoints in the sample have a k-anonymity less than 10^{100}.

Next We consider predicates given by knowing the first and last datapoint of the individuals trajectory, this gives 9926 predicates, $p_1, \ldots, p_{9926}$, each corresponding to one of the trajectories. In this case we are able to compute $|p_i(P)|$ for all i's and the cumulative distribution is given in Figure 17b. For 135 trajectories it is equal to 1, which means that they are uniquely identified



(a) Empirical cumulative distribution function for a sample of $|p_d(P)|$ with 5% of the datapoints.



(b) Empirical cumulative distribution function for $|p_i(P)|$ for all trajectories r_i in the co-trajectory.

by the predicate, however it is only for slightly more, 179, trajectories that it is less than 10^{100}.

We can also make a comparison to the results of de Montjoye et al. showing that given data about location of mobile phones, as determined by the carrier's antenna, four datapoints is enough to uniquely identify 95% of people in group of one and a half million [3]. Their case is very different from ours in that the datapoints are not unique, many people are connected to the same antenna at a time and knowing a datapoint does therefore not uniquely identify a person. In our case the datapoints are in practice unique and knowing only one datapoint is enough to uniquely identify an individual in the original co-trajectory. However, after applying SwapMob this is, as we have seen, no longer the case and to compare to de Montjoye et al. we can consider the family of predicates given by knowing four datapoints of a trajectory. How much protection does SwapMob give in this case? Given four datapoints, d_1, d_2, d_3, d_4, we get a predicate $p_{d_1 d_2 d_3 d_4}$ which is satisfied for trajectories which contain all four datapoints. Again we are not able to compute this for all predicates in this family but randomly sample around 20,000 predicates from the family and compute the number of trajectories in $p_{d_1 d_2 d_3 d_4}(P)$ for each one. The sampling is done by choosing first choosing

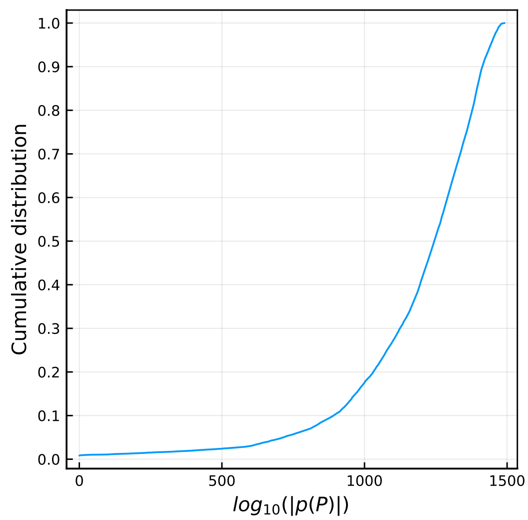


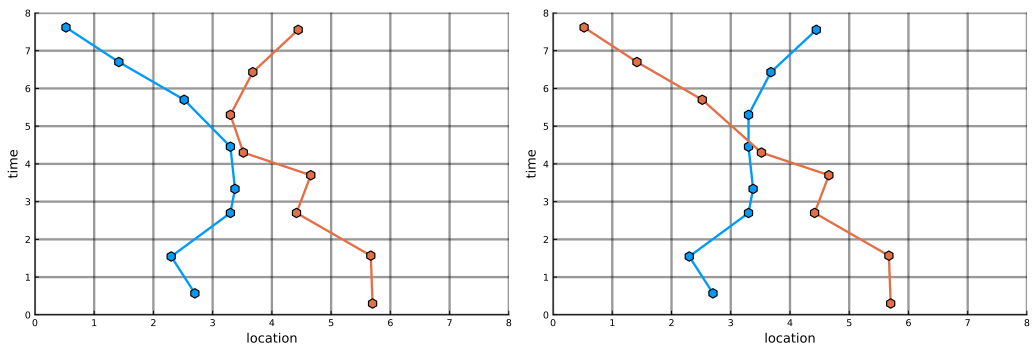
Figure 18: Empirical cumulative distribution function for $|p_{d_1 d_2 d_3 d_4}(P)|$ given by a sample of size about 20,000.

trajectory uniformly at random and then choosing four distinct datapoints for this trajectory. The cumulative distribution is given in Figure **??**. The values are in general lower than those for p_d), which is to be expected since four datapoints conceal more information than one, but similar to those of p_i).

4.4 Utility of data

SwapMob is a tool for improving privacy when working with co-trajectories. However, to be a useful tool it should not only improve the privacy but also leave enough structure so that the result is still useful for analysis. The utility of the data given by SwapMob has to be sufficiently high or SwapMob would never be used. If SwapMob preserves enough structure depends on the type of structure is important for the analysis to be performed. In this section we discuss what type of structure is and is not preserved by SwapMob and also some possible use cases for it. Most use cases listed here are related to mobility data for individuals, in particular in traffic.

On a general level SwapMob does not preserve long time behaviour of individuals since swapping splits up the trajectories into smaller pieces. This is important for privacy but means that it cannot be used whenever long term



(a) Original co-trajectory consisting of two trajectories with one possible swap.

(b) Co-trajectory after swapping the two trajectories.

Figure 19: Most transitions in (a) are preserved after swapping in (b). The only ones that are not are the two transitions between the time intervals $[4, 5]$ and $[5, 6]$. If we only consider the transitions up to the partitioning of the location space these transitions are also preserved, the two transitions changed by swapping still correspond to one transition from $[3, 4]$ to $[2, 3]$ and one from $[3, 4]$ to $[3, 4]$ in location space.

behaviour is analysed. For example it does not preserve origin and destination pairs nor does it work for finding recurring behaviour of trajectories. What it does preserve is local behaviour, everything between two swaps is preserved. It could therefore be used to estimate flow of traffic or number of trajectories caught in red lights.

One important property of SwapMob is that is preserves all individual datapoints. This means it perfectly preserves the density of trajectories since this is only based on the number of datapoints in an area. It can therefore be used to locate popular and less popular areas.

If we consider all the individual transitions of the co-trajectory, that is all transitions between datapoints done by trajectories, it is in general not preserved by SwapMob since at the places where the swaps are performed these transitions are changed. If we only consider the transitions up to the partitioning used with SwapMob, i.e. we only care about between which partitions the transition happen, then it is preserved by SwapMob. This is visualized in Figure 19. This is relevant for the Markov chain model for co-trajectories discussed in Section 5.1.

This is not an exhaustive list of structure that is preserved by SwapMob. Whenever analysis is performed with data on which SwapMob has been used care has to be taken to ensure that the conclusion of the analysis is valid for the original co-trajectory and not only the co-trajectory resulting from SwapMob.

5 Modelling co-trajectories

In this section we discuss different stochastic models for trajectories and co-trajectories. Stochastic models can be an important tool when analysing co-trajectories, extracting information and using it for predictions. For example Markov chain models of trajectories can be used for next-place predictions [4, 21, 22].

We begin by introducing a very general stochastic model for co-trajectories that is closely related to the definitions of trajectories and co-trajectories from Section 2. This model can been seen as a basis on which more specific models can be built, as an example we define a Markov chain model for co-trajectories in Section 5.1.

Similar to how a co-trajectory in Section 2 is defined as a collection of trajectories, a stochastic model of a co-trajectory will be given by a collection of stochastic variables, each corresponding to a trajectory. In general the location space, $\mathbb{L}$, in which the movement of the trajectories take place can be any set, however, for simplicity, we assume throughout this section that the location space is finite. In practice this is not limiting since most trajectories are bounded in space and we can get a finite representation by choosing a sufficiently fine discretization. For example $\mathbb{L}$ could be given by a fine grid on a map or a graph where the nodes represents possible locations.

A trajectory is modelled by a continuous-time stochastic process

$$(X_t)_{t\geq 0} = (X_t : 0 \leq t < \infty)$$

with random variables $X_t : \Omega \to \mathbb{L}$. The location of the trajectory at time t is given by the random variable X_t. This model is a bit to general to work well with Definition 2.2 for trajectories as it allows for trajectories that cannot be represented in that form. For example it allows for trajectories which jump to a location, stay there for zero time, and then jump to another location, something we cannot represent using Definition 2.2.

To handle this problem we shall restrict our attention to stochastic processes that are *right continuous*. In this context, with a finite state space, a stochastic process $(X_t)_{t\geq 0}$ is called *right continuous* if it satisfy that for all $\omega \in \Omega$ and $t \geq 0$ there exists $\epsilon \geq 0$ such that

$$X_s(\omega) = X_t(\omega) \text{ for } t \leq s \leq t + \epsilon.$$

Every path $t \to X_t(\omega)$ of a right continuous process must remain constant for a while in each new state it enters before jumping to the next state. Restricting to right continuous processes turns out to be almost enough to be enough to be able to related the model to Definition 2.2 by considering the *jump times* and the *jump process* of $(X_t)_{t\geq 0}$.

The *jump times*, $J_0, J_1, \ldots$, of $(X_t)_{t\geq 0}$, are defined by

$$J_0 = 0,\ \ J_{n+1} = \inf\{t \geq J_n : X_t \neq X_{J_n}\}$$

for $n = 0, 1, \ldots$, where $\inf \emptyset = \infty$. From right continuity we get that $J_{n+1} > J_n$ for all n. The *jump process* of $(X_t)_{t\geq 0}$ is then the discrete time process $(Y_n)_{n>0}$ given by $Y_n = X_{J_n}$.

From here we get a clear relation to Definition 2.1 of a datapoint, where for $\omega \in \Omega$ the pair $(Y_n(\omega), J_n(\omega))$ gives a datapoint and (Y_n, J_n) can be seen as a random datapoint with Y_n corresponding to the location and J_n to the time of the datapoint. For the set $\{(Y_n(\omega), J_n(\omega))\}_{n>0}$ to correspond to a random trajectory according to Definition 2.2 we need that no two datapoints have the same time, which is true since $J_{n+1} > J_n$, and that we have only a finite number of datapoints in every finite time interval. This last requirement does not necessarily hold, we have three different possibilities for the process:

1. It jumps finitely many times, in which case there is an n such that $J_n = \infty$.

2. It jumps infinitely many times but only a finite number of times in every finite interval $[0, t]$, in which case $J_n < \infty$ for all n but $\lim_{n\to\infty} J_n = \infty$.

3. It jumps infinitely many times in a finite interval, in which case $\lim_{n\to\infty} J_n = \zeta < \infty$.

In the last case the process is said to *explode* and ζ is the *explosion time*. For a trajectory an explosion would correspond to an infinite number of datapoints in a finite time. Since we avoided this in Definition 2.2 we choose to avoid it here as well and therefore further limit the study to non-exploding stochastic processes.

So by limiting us to right continuous, non-exploding stochastic processes $(X_t)_{t\geq 0}$ we can define the discrete stochastic process $\{(Y_n, J_n)\}_{n\geq 0}$, which for $\omega \in \Omega$ gives us a trajectory $\{(Y_n(\omega), J_n(\omega))\}_{n\geq 0}$ according to Definition 2.2. From the definition of Y_n we have that such a trajectory will never have two datapoints in a row with the same location, this is different from a general trajectory where we do allow for multiple consecutive datapoints with the same location. A general trajectory can be seen as a sampling of a stochastic process $(X_t)_{t\geq 0}$ at the times for the datapoints, then it is possible to have several consecutive samples where $(X_t)_{t\geq 0}$ is constant. In this case it is also possible to miss locations that $(X_t)_{t\geq 0}$ visits if the process jumps more than once in between two samples. For any trajectory r we can get the corresponding jump process and jump times by removing any consecutive datapoints with the same location, only keeping the first one.

A co-trajectory can then be modelled by a collection of stochastic processes $\{(X_t^i)_{t\geq 0}\}_{i=1}^N$, where the location of trajectory i at time t is given by the random variable X_t^i. We here make no assumptions about the dependencies between the stochastic processes $(X_t^i)_{t\geq 0}$ and do allow them to be both dependent and independent of each other.

A natural limitation to put on the processes for the trajectories is to not allow them to jump from any state in $\mathbb{L}$ to any other state, instead they are only allowed to jump to "nearby" states. What indicates a nearby state depends highly on $\mathbb{L}$. If $\mathbb{L}$ is given by the vertices in a road network represented by a directed graph then it would be natural to only allow movement along edges of the graph, so if $Y_n = v$ for some vertex v in the graph then Y_{n+1} must be some vertex to which v has an outgoing edge. In general, any limitations on between which states a process can jump can be modelled by a directed graph with the vertices given by $\mathbb{L}$ and the edges representing allowed jumps.

5.1 Markov chain models for co-trajectories

One of the simplest models for a continuous-time stochastic process is a continuous-time Markov chain model. Essentially, a Markov chain model for a trajectory is a model where the future movement of the trajectory only depends on the current location of the trajectory and not on which locations have been visited before that.

We here give a very brief introduction and definition of a continuous-time Markov chain and also some notes about how to infer the Markov structure using data from a co-trajectory. We discuss some of the benefits and some of the drawbacks of using a Markov chain model and also mention some possible generalizations of it.

For a complete introduction to Markov chains see any book in the subject, e.g. [23]. The introduction we give here is based on the structure of the *jump times*, $(J_n)_{n>0}$ and the *jump process*, $(Y_n)_{n>0}$. From the jump times we can define the holding times $(S_n)_{n>1}$ given by

$$S_n = J_n - J_{n-1}.$$

They give the time that the process $(X_t)_{t\geq 0}$ stays in a state before jumping to the next one. For a Markov chain we should have that conditional on $Y_0 = x_0, \ldots, Y_{n-1} = x_{n-1}$, the holding times $S_1, \ldots, S_n$ are independent exponential random variables of parameters $q_{x_0}, \ldots, q_{x_{n-1}}$. So when the process comes to a state x it stays there for and exponential time of parameter q_x, independent of what has happened before, after which it jumps to a new state.

For the jump process $(Y_n)_{n>0}$ the requirement is that it should be a discrete time Markov chain. A discrete time Markov chain is defined by its initial distribution λ that gives the distribution of Y_0 and stochastic matrix Π on $\mathbb{L}$ which gives the probabilities of jumping between two states. A stochastic matrix on $\mathbb{L}$ is given by $\Pi = (\pi_{xy} : x, y \in \mathbb{L})$ satisfying

1. $0 \leq \pi_{xy} \leq 1$ for all $x, y \in \mathbb{L}$;
2. $\sum_{y\in\mathbb{L}} \pi_{xy} = 1$ for all $x \in \mathbb{L}$.

Where π_{xy} is the probability of jumping to state y from state x. For it to work properly with the continuous-time Markov chain we also need that $\pi_{xx} = 0$ if $q_x \neq 0$, if $q_x = 0$ then $\pi_{xy} = 0$ for $x \neq y$ and $\pi_{xx} = 1$. The interpretation of the last part is that if $q_x = 0$ then the process $(X_t)_{t\geq 0}$ will never leave the state x and the jump process $(Y_n)_{n>0}$ will thus have a zero probability to go anywhere else and probability one to stay.

To summarize, a continuous-time Markov chain is defined by the triple $(\{q_x\}_{x\in\mathbb{L}}, \lambda, \Pi)$, where $\{q_x\}_{x\in\mathbb{L}}$ determines how long the process stays in the different states, λ gives the initial distribution and Π gives the probabilities for jumping between the states.

The simplest Markov chain model for co-trajectories is given by a collection of trajectories $\{(X_t^i)_{t\geq 0}\}_{i=1}^N$, where the trajectories are all independent and identically distributed continuous-time Markov chains. Since the trajectories are identically distributed they share the same parameters $\{q_x\}_{x\in\mathbb{L}}$ and matrix Π.

Using collected data for a co-trajectory the parameters $\{q_x\}_{x\in\mathbb{L}}$ and Π) can be estimated. Every trajectory is then seen as one instance of the Markov chain model and by removing consecutive datapoints with the same location, only keeping the first one, we get an instance of the jump process and the jump times for every trajectory. With this, the parameters $\{q_x\}_{x\in\mathbb{L}}$ can be estimated by the average holding times for the different states, i.e. q_x is given by the average time a trajectory stays in the state x. The matrix Π can be estimated by considering the transitions between the states, for a state x all transitions going out from that state are collected and the probabilities π_{xy} are then given by the number of transitions from x to y divided by the total number of transitions from x.

As mentioned in Section 4.4 SwapMob preserves transitions on partitioning used. If the same partitioning is used for the Markov chain SwapMob will thus preserve the estimate of Π). The holding times $\{q_x\}_{x\in\mathbb{L}}$ will also be preserved when swapping which means that SwapMob fully preserves the estimated Markov chain.

The Markov chain model gives a very local picture of a co-trajectory. The transition probabilities gives information about general direction of movement in a location, where trajectories are most likely to go directly after visiting it. The parameters $\{q_x\}_{x\in\mathbb{L}}$ give information about how long time a trajectory usually spend at the locations. If the trajectories represent people this could give information about what type of activities are carried out at the location, we expect someone to stay longer at the cinema than at the super market, but if the trajectories represent vehicles it could be interpreted as speed.

By comparing co-trajectories collected at different times this can give a quantitative measure of differences. Comparing the holding time parameters gives us information about changes in how long time trajectories spend at a location, this could for example indicate changes in traffic flow. From the transition probabilities we can see changes in direction of movement, for example by comparing co-trajectories from the morning with ones from the evening we could see differences based on the commuting going in different directions.

There is however a lot of information in co-trajectories that the Markov chain model does not capture, it does not handle global nor time dependent behaviour well. One example of global behaviour it does not capture is movement between two locations that are far from each other. The Markov chain can only capture the individual transitions between all locations in between but since it only takes into account the last visited location it cannot track the whole path. Most movement of people is time dependent, for example during the morning, the day and the evening the movement is very different. Since the transition probabilities in the Markov chain does not depend on time this cannot be taken into account. What can be done is to consider several different models, using data from the morning to make a model for the morning, data from the day to make a model for the day and data from the evening to make a model for the evening. In this way you can see differences in behaviour by comparing the different models, however this requires you to manually choose the times for each model.

Another thing this model does not handle is dependencies between trajectories, in this simple model all trajectories are assumed to be independent. Often times there will be dependencies between trajectories, for example in traffic flow trajectories that are close to each other will have a similar flow.

There are many possible generalizations that can be made to this simple model. One natural generalization is to let the transition probabilities depend not only on the current state but on the last n states. Another one is to allow different trajectories to be modelled by different Markov chains. This would make sense if there are different classes of trajectories, in which case a separate Markov chain model could be used for every class.

6 Conclusions and future work

We have presented a method, SwapMob, for enhancing privacy when working with co-trajectories. We have given a graph representation of the method for analysing the increase in privacy that it gives. When analysing the privacy for a real dataset we have seen that SwapMob greatly increases the k-anonymity. SwapMob also preserves many properties that are important for analysis, such as datapoint density and local behaviour of trajectories.

The main focus for future work would be to continue to analyse the privacy given by SwapMob. Considering other measures for privacy than k-anonymity, in particular coming up with a way to measure diversity, would be important. Also considering different kinds of predicates, with a focus on those that would be likely to encounter in practice. Taking a look at other kinds of privacy that just re-identification attacks would also be of interest.

A very general stochastic model for co-trajectories was introduced. While it could serve as a base for other more specific models it is by itself most likely to general to be of use. One more specific mode, a Markov chain model, was defined together with how to estimate its parameters using data from a co-trajectory. Some generalizations of the Markov chain model were also discussed.

We have only taken a very brief look at stochastic models for co-trajectories and there is much potential for future research. It would be of interest to apply the Markov model to real questions about real data to see how useful it is. Looking further at possible generalizations of the Markov model as well as studying other types of stochastic models is also a line to go.

References

[1] Jakub Novak, Rein Ahas, Anto Aasa, and Siiri Silm. Application of mobile phone location data in mapping of commuting patterns and functional regionalization: A pilot study of Estonia. *Journal of Maps*, 9:10–15, March 2013.

[2] Fabio Pinelli, Rahul Nair, Francesco Calabrese, Michele Berlingerio, Giusy Di Lorenzo, and Marco Sbodio. Data-Driven Transit Network Design From Mobile Phone Trajectories. *IEEE Transactions on Intelligent Transportation Systems*, 17:1–10, May 2016.

[3] Yves-Alexandre de Montjoye, César A. Hidalgo, Michel Verleysen, and Vincent D. Blondel. Unique in the Crowd: The privacy bounds of human

mobility. *Scientific Reports*, 3(1), December 2013.

[4] Sébastien Gambs, Marc-Olivier Killijian, and Miguel Núñez del Prado Cortez. Show Me How You Move and I Will Tell You Who You Are. In *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Security and Privacy in GIS and LBS*, SPRINGL '10, pages 34–41, New York, NY, USA, 2010. ACM.

[5] Linnet Taylor. No place to hide? The ethics and analytics of tracking mobility using mobile phone data. *Environment and Planning D: Society and Space*, 34(2):319–336, April 2016.

[6] Vincent Primault, Antoine Boutet, Sonia Ben Mokhtar, and Lionel Brunie. The Long Road to Computational Location Privacy: A Survey. *Communications Surveys and Tutorials, IEEE Communications Society*, page 1, 2018.

[7] Latanya Sweeney. k-ANONYMITY: A MODEL FOR PROTECTING PRIVACY. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, October 2002.

[8] Osman Abul, Francesco Bonchi, and Mirco Nanni. Never Walk Alone: Uncertainty for Anonymity in Moving Objects Databases. In *2008 IEEE 24th International Conference on Data Engineering*, pages 376–385, Cancun, Mexico, April 2008. IEEE.

[9] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. L-diversity: privacy beyond k-anonymity. In *22nd International Conference on Data Engineering (ICDE'06)*, pages 24–24, April 2006.

[10] Miguel Andrés, Nicolás Bordenabe, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. Geo-Indistinguishability: Differential Privacy for Location-Based Systems. pages 901–914. ACM, November 2013.

[11] Gabriel Ghinita, Panos Kalnis, Spiros Skiadopoulus, Jaffar Joxan, Gabriel Ghinita, Panos Kalnis, and Spiros Skiadopoulos. PRIVE: Anonymous Location-Based Queries in Distributed Mobile Systems. In *In WWW '07: Proceedings of the 16th International Conference on World Wide Web*, pages 371–380. ACM Press, 2007.

[12] Kaifeng Jiang, Dongxu Shao, Stéphane Bressan, Thomas Kister, and Kian-Lee Tan. Publishing Trajectories with Differential Privacy Guar-

antees. In *Proceedings of the 25th International Conference on Scientific and Statistical Database Management*, SSDBM, pages 12:1–12:12, New York, NY, USA, 2013. ACM. event-place: Baltimore, Maryland, USA.

[13] Daniele Quercia, Ilias Leontiadis, Liam Mcnamara, Cecilia Mascolo, and Jon Crowcroft. SpotME If You Can: Randomized Responses for Location Obfuscation on Mobile Phones. In *In Proceedings of the 31st International Conference on Distributed Computing Systems (ICDCS*, 2011.

[14] Nikos Pelekis, Aris Gkoulalas-Divanis, Marios Vodas, Despina Kopanaki, and Yannis Theodoridis. Privacy-aware querying over sensitive trajectory data. In *Proceedings of the 20th ACM international conference on Information and knowledge management - CIKM '11*, page 895, Glasgow, Scotland, UK, 2011. ACM Press.

[15] H. Kido, Y. Yanagisawa, and T. Satoh. Protection of Location Privacy using Dummies for Location-based Services. In *21st International Conference on Data Engineering Workshops (ICDEW'05)*, pages 1248–1248, April 2005.

[16] A.R. Beresford and F. Stajano. Location privacy in pervasive computing. *IEEE Pervasive Computing*, 2(1):46–55, January 2003.

[17] A.R. Beresford and F. Stajano. Mix zones: user privacy in location-aware services. In *IEEE Annual Conference on Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second*, pages 127–131, Orlando, FL, USA, 2004. IEEE.

[18] Julián Salas, David Megías, and Vicenç Torra. SwapMob: Swapping Trajectories for Mobility Anonymization. In Josep Domingo-Ferrer and Francisco Montes, editors, *Privacy in Statistical Databases*, Lecture Notes in Computer Science, pages 331–346. Springer International Publishing, 2018.

[19] Jing Yuan, Yu Zheng, Chengyang Zhang, Wenlei Xie, Xing Xie, Guangzhong Sun, and Yan Huang. T-Drive: Driving Directions Based on Taxi Trajectories. November 2010.

[20] Jing Yuan, Yu Zheng, Xing Xie, and Guangzhong Sun. Driving with Knowledge from the Physical World. August 2011.

[21] Sébastien Gambs, Marc-Olivier Killijian, and Miguel Núñez del Prado Cortez. Next place prediction using mobility Markov chains. In *Proceedings of the First Workshop on Measurement, Privacy, and Mobility - MPM '12*, pages 1–6, Bern, Switzerland, 2012. ACM Press.

[22] Zhixian Yan, Dipanjan Chakraborty, Christine Parent, Stefano Spaccapietra, and Karl Aberer. SeMiTri: A Framework for Semantic Annotation of Heterogeneous Trajectories. *Proceedings of the 14th International Conference on Extending Database Technology (EDBT 2011)*, pages 259–270, 2011.

[23] J. R. Norris. *Markov Chains*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 1997.