Adaptative generalisation over a value hierarchy for the k-anonymisation of Origin-Destination matrices

Benoit Matet*[1,2], Angelo Furno[2], Marco Fiore[3], Etienne Côme[1], and Latifa Oukhellou[1]

[1]Univ. Gustave Eiffel, GRETTIA, Marne-la-Vallée, France
[2]Univ. Gustave Eiffel, Univ. Lyon, ENTPE, LICIT-ECO7, Lyon, France
[3]IMDEA Networks Institute, Madrid, Spain

July 2022

Abstract

The study of transportation relies on mobility data, containing information on the whereabouts and movements of individuals in a study area. These data are often represented in the simple form of origin-destination (OD)-matrices, which are a valuable indicator for the management of transportation networks but also present risks to the privacy of the individuals. In particular, the significant size (i.e., number of distinct flows) and high number of modalities (produced by a high-resolution zoning) of OD-matrices call for an adapted, fast algorithm that can efficiently anonymise them. In this paper, we develop a lightweight approach for the k-anonymisation of OD-matrices that exploits the low dimension of the data to explore a larger solution space than regular generalisation algorithms, while keeping relevant restrictions of the search space in order to be scalable on matrices with high number of flows. We apply it to a variety of real-world large-scale O-D matrices collected by the New York City Taxi and Limousine Commission and derived from the Data for Development (D4D) challenge organised by Orange in Senegal and Côte d'Ivoire. Compared to an extensive benchmark of regular generalisation algorithms and mobility anonymisation state-of-the-art, we show that our method is 27% more precise and 9 times faster than comparable approaches able to scale on the same datasets.

Keywords— Origin Destination matrix, anonymisation, generalisation, suppression

1 Introduction

Mobility data have become more and more accessible thanks to the evolution of processes to collect them from various sources. Initially only available through ground surveys, the wide spread of GPS devices and mobile phones has paved the way for the collection of huge volumes of data, notably Call Detail Records (CDRs) and passive Network Signalization Data (NSD) from mobile phone usage. After processing, these sources offer a detailed report of the trajectories of a significant part of the population over a given study area, offering invaluable information for the organisation of territories and transportation networks. However, it is clear that mobility data contain potentially personal information, and special care should be taken in their handling. We take as a reference the European General Data Protection Regulation (GDPR) [1], which defines as personal information any piece of information pertaining to a particular individual. Apart from their usefulness and their sensitivity, mobility data are also characterized by their uniqueness: individuals tend to have very different trajectories over the course of a day, which makes it difficult to extract trends or group them in order to provide more privacy-preserving mobility reports [2, 3].

The simplest indicator we can extract from a set of trajectories is an Origin-Destination (OD)-matrix, describing the flows between origins o and destinations d over a time window $[t, t + \Delta t)$ (Fig. 1). Typical values for Δt range from 15 to 60 minutes, depending on the specific transport application [4]. A set of OD-matrices is then a regular, relational dataset with three attributes for each flow: an origin o, a destination d, and a time t usually denoting the time step of the beginning of the flow. Although they represent a dramatic simplification compared to trajectories, they are still a crucial indicator of mobility. Like mobility data, OD-matrices keep their property of uniqueness (*i.e.,* flows can be small and isolated from others), and a high

*Corresponding author: benoit.matet@univ-eiffel.fr

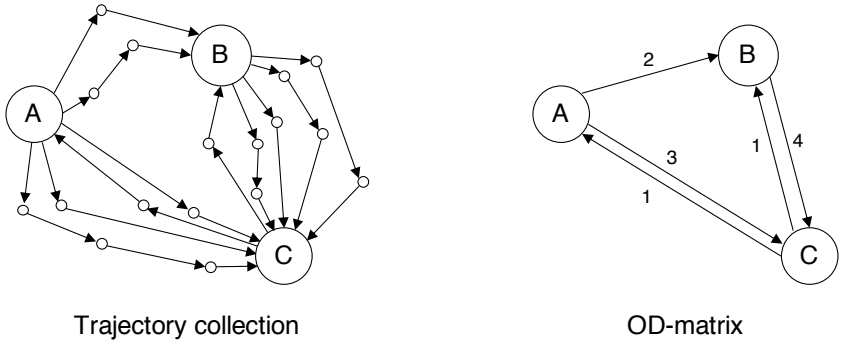


Figure 1: Illustration of a collection of trajectories and the resulting Origin-Destination matrix, set on areas of interest A, B and C.

number of modalities: the origin and destination can be among sets of up to thousands of areas. This makes OD-matrices harder to anonymise than regular relational data. In this work, we propose a methodology to efficiently make OD-matrices meet GDPR's restrictive definition of anonymous data, *i.e.,* the data subject is not or no longer identifiable (Recital 26 GDPR). According to this definition, European regulators consider small flows to constitute personal information, making the distribution of OD-matrices problematic in the European Union even though they are admittedly much safer than the trajectory collections from which they are derived. Formal privacy guarantees are required to ensure small flows are protected from any specific form of re-identification. To this end, we use the widely spread criterion of k**-anonymity** [5]. It is used both in research and by authorities such as French regulator CNIL, which makes it an appropriate criterion as our practical aim is to foster the use of OD-matrices by anonymising them. An OD-matrix is said to be k-anonymous if no flow represents less than k individuals. By k-anonymising an OD-matrix, we can improve its safety against the various categories of attacks [6]:

- **Record linkage**, also called re-identification attack. If a flow in the OD-matrix has only one individual, an attacker can pinpoint a target individual in the data.
- **Attribute linkage**, also called homogeneity attack. If all the individuals from a particular origin go to the same destination at a given time, then an attacker knowing that a target left the origin area can infer where they went, without needing to pinpoint the target first.
- **Probabilistic attack**. If an attacker knows anything about a target, then accessing the OD-matrix can improve their knowledge about the target's whereabouts.

Each one of these three attacks is a relaxation of the previous one, the goal being less ambitious for the attacker but the success more likely. Note that probabilistic attacks may be successful even if the target user is not actually in the data, as long as we consider the OD-matrix to be representative of the population flows in the study area.

Technically, k-anonymity with k as low as 2 is secure against record linkage attacks, although this would understandably not be a satisfying anonymity condition. In particular, it would offer very limited protection against attribute linkage and probabilistic attacks. State-of-the-art usually aims at k between 3 and 5 in the case of hard-to-anonymise datasets, and up to 200 for simple datasets [7]. A complementary property is l-diversity [8], which holds when each generalised value covers at least l distinct modalities. Together, k-anonymity combined with l-diversity ensures protection against attribute linkage attacks, and significantly reduces the potential of probabilistic attacks [9]. For even further protection, it is possible to implement t-closeness [10], which holds when the distribution of attributes in each group of k is no further than a threshold t from the total distributions. However for OD-matrices, it can be argued that k-anonymity alone, given a sufficiently high k, is enough to make the individuals no longer identifiable: The areas of origin and destination can be large enough to avoid attribute linkage, and the low number of attributes renders probabilistic attacks rather vacuous.

The main approach to k-anonymisation is **generalisation and suppression** [5], *i.e.,* reducing the precision of the data until the flows are big enough and suppressing the ones that are not. A toy example of 10-anonymisation generalisation and suppression is illustrated in Fig. 2: flows $A \to A$, $A \to B$ and $A \to C$ are aggregated together in order to reach a volume above 10, and similar aggregations are performed on the flows originating from C and D, and directed towards destinations B, C and D. Flow $B \to A$ is suppressed, which can be preferable when the necessary aggregation to hide it is too coarse. Finding such a solution that minimizes the loss in precision is known to be a NP-hard problem [11]. Historically, the first k-anonymisation algorithm was Datafly [5], which relies on a generalisation hierarchy describing how modalities should be

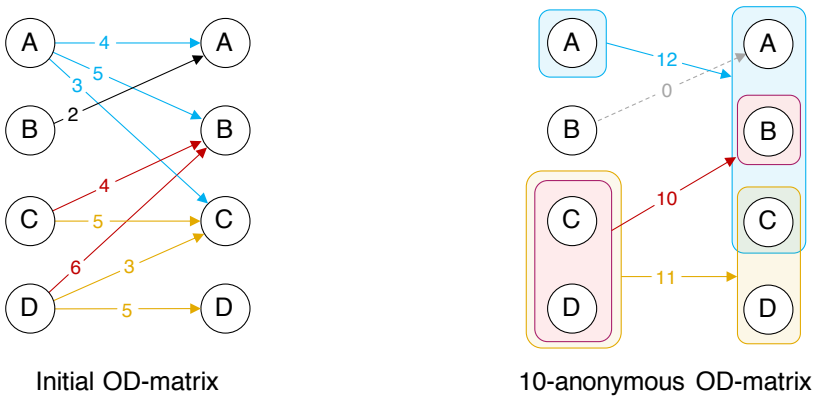


Figure 2: Example of generalisation and suppression of a simple OD-matrix. Note that the flows here have been clustered without any kind of constraint. Some approaches seek particular solutions, notably ones where the generalisations form a partitioning of the domain.

merged together. An example of such a generalisation hierarchy is illustrated in Fig. 3: for a dataset giving the position of individuals in a study zone, the initial modalities are represented by the leaves of the tree. If we choose to generalise the whole dataset one level higher, then the possible modalities are the parents of the leaves. Datafly finds the best horizontal cut in the hierarchy, meaning everyone in the data is generalised to the same level. This **uniform generalisation** approach has the advantage of being scalable to huge volumes of data, as well as data with numerous attributes. In a bid to find a finer-grained result, some approaches look for a generalisation at the individual level, which gives a solution akin to a clustering [7]. In the specific field of mobility data, this approach is better represented by Glove [2], which generalises points in a dataset of trajectories. However, these approaches lack scalability for datasets characterized by a huge number of flows, as is the case for data derived from mobile phone CDRs and NSD: for example, Liang and Samavi [7] report computing times in the order of hours for low values of k, and their approach can be expected to take days for $k \geq 10$ given its exponential time dependency. In order to achieve fast, accurate anonymisation of OD-matrices, dedicated processing is needed. In this paper, we develop an approach for k-anonymisation of OD-matrices with a focus on scalability. Our contribution is three-fold:

- We formulate generalisation and suppression over a generalisation hierarchy as an optimisation problem, taking advantage of the low dimensionality of OD-matrices in order to broaden the search domain of solutions. As representativity is paramount for the value of mobility data, we also control the volume of suppressed records by keeping it under a fixed constraint. We solve it by using state-of-the-art algorithms for dependency-constraint knapsack problems, finding an **adaptative generalisation** finer than uniform ones. Based on this formulation, we propose Adaptative Tree Generalisation (ATG), a lightweight algorithm to efficiently reach k-anonymity. The algorithm is proposed in two versions, **ATG-Dual** and **ATG-Soft**, corresponding to two variations of the problem we solve.
- We evaluate our approaches against an extensive benchmark of anonymisation methods from the state of the art: uniform generalisation from the general field of anonymisation, clustering from mobility data anonymisation, and differential privacy. The approaches are compared on a variety of datasets: New-York city taxis available in open-data, and the Senegal and Cote d'Ivoire datasets available in the scope of the Data for Development (D4D) challenge [12, 13]. Several variations of the data are used to analyze how the various approaches adapt to different conditions, for a total of six distinct datasets.
- As the approaches vary vastly in the format of their output and their interpretations, it is not trivial to compare them fairly. We define and discuss the relevance of various indicators used to measure the information loss due to anonymisation.

The remainder of this paper is organised as follows: we review related work on data anonymisation in Section 2. We then detail our methodology in Section 3. In Section 4, we describe the datasets and the methods we compared in our experiments, then we discuss the appropriate indicators to compare various anonymisation methods before presenting our results. Section 5 concludes the paper. Implementation and methodology details are available in the supplementary material.

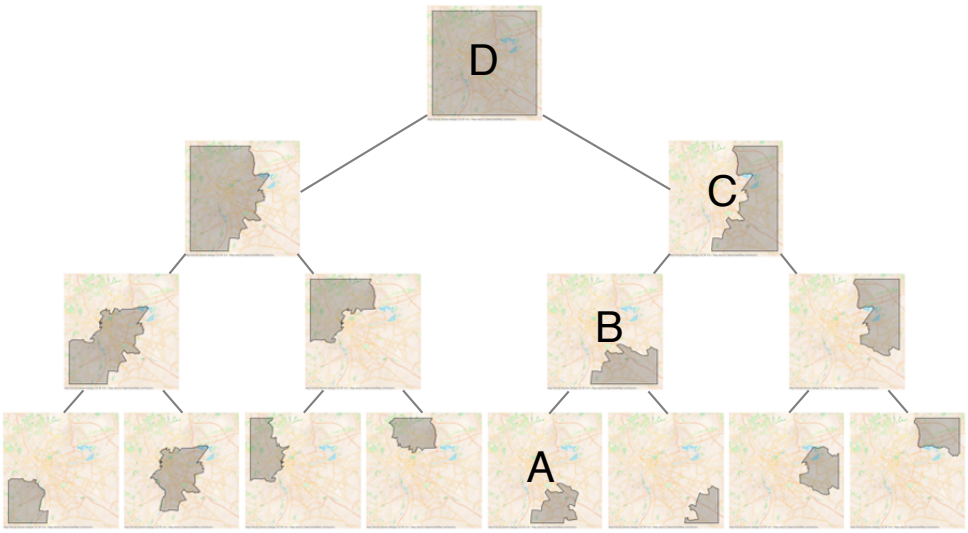


Figure 3: Example of a spatial generalisation hierarchy. The root represents the whole study map, and the children of a node form a partitioning of the parents. An individual present in area A in the data can be generalised to be shown as present in area B, C or D depending on what is necessary in order to hide them in a group of k individuals.

2 Related work

In this section, we first review the k-anonymisation methods developed specifically for mobility data, then in the general case of structured data, and finally differential privacy. We conclude the section with our positioning compared to the k-anonymisation techniques from the state of the art.

2.1 Trajectory data anonymisation

Previous work on the anonymisation of mobility data has focused on the anonymisation of collections of trajectories. the generalisation and suppression of complete trajectories is computationally expensive, as it relies on hierarchical clustering with custom merging procedures to accommodate the complex nature of the data. The first notable work is Never Walk Alone (NWA) [14], which considers inherent spatial uncertainty to create groups of k individuals that share parts of their uncertainty areas. Its time-tolerant variation, Wait For Me (WMA) [15], uses a variation of the edit distance adapted for quantitative values [16] in order to handle the uncertainty in time as well as in space. As the edit distance is computationally expensive, its authors also propose a linear spatiotemporal distance intended for large databases. Glove [2] uses another rule of merging trajectories that yields better precision and does not add artificial trajectory points in order to create groups. A subsequent approach by Tu [17] uses another merging logic that also implements l-diversity and t-closeness. These methods initially proposed for trajectories can be used for an OD-matrix if we consider them to be collections of two-point trajectories. The merging procedures then become straightforward as the time coherence constraint states that the time intervals spanned by the generalised points must not overlap. Then the only possible merge in a single, fixed-timestep OD-matrix consists in merging the origins together and the destinations together. Despite this simplification, the underlying clustering is expected to find a large, variable number of clusters among huge volumes of records, a task for which regular approaches scale poorly [18].

2.2 Generalisation and suppression for relational data

One single OD-matrix, corresponding to a single time step, can be seen as a structured table with only two attributes. Historically, the first k-anonymisation algorithm for structured data is **Datafly** [5], relying on a tree-like Value generalisation Hierarchy (VGH) for each attribute (Fig. 3). Datafly iteratively generalises all the values of an attribute to their parent values in the hierarchy, selecting heuristically at each step the attribute that has the most distinct values. This results in a horizontal cut in the hierarchy of each attribute. We call this family of generalisation **uniform**, because all the values are generalised to the same level. The uniform generalisation approach is still the object of active research, with efforts to replace the heuristic by an efficient exploration of the lattice of all possible generalisations. The latest and best adapted approach to

an OD-matrix is the **OIGH** algorithm [19], specifically designed for data where the hierarchy is the same for all attributes. An example of uniform generalisation applied to an OD-matrix is given in Fig. 4: the initial zones A, B, C, D are associated to a generalisation hierarchy in dashed lines, and uniform generalisation gives a cut in the tree that aggregates every modality of an attribute to the same level. It is not possible to detail destinations between A and B while aggregating destinations C and D together.

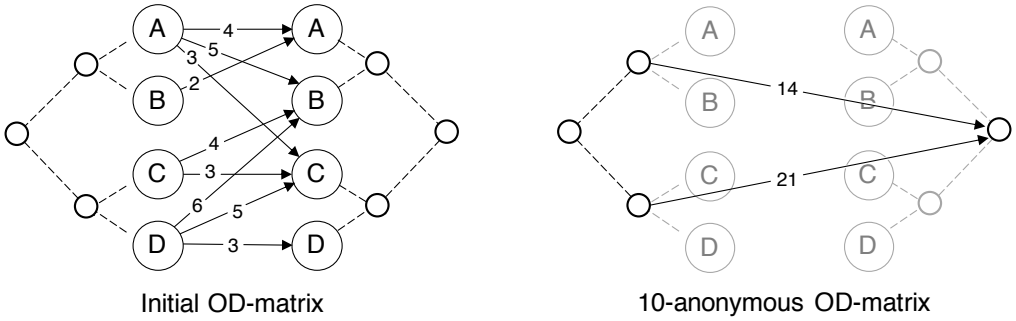


Figure 4: Example of an OD-matrix with hierarchies for origins and destinations, and one possible output of uniform generalisation.

Disregarding the value generalisation hierarchy, we can also consider an OD-matrix to be composed of four attributes, namely the coordinates of origin and destination. The k-anonymisation of quantitative data can be efficiently handled by the **Mondrian** algorithm [20], which finds a partitioning of attributes using an approach inspired by kd-trees [21]. Mondrian considers the flows as 4-dimensional points, and at each step selects a dimension and performs a median cut. Then it repeats for each subset of points, and each branch formed this way stops once it is found that any additional cut would imply clusters of less than k points. As such, Mondrian does not allow suppression at all.

In recent works generalisation and suppression have also been formulated as an optimisation problem. The constraints of the problem ensure k-anonymity while the objective function to minimize is a measure of the coarseness of generalisation. Liang and Samavi [7] formulate the generalisation of each individual value by a mixed-integer linear program. Directly solving for this linear program remains hard, so they propose a practical *Split & Carry* algorithm that splits the problem into smaller, more manageable sub-problems, as well as a greedy search for bigger datasets. The solution found does not necessarily give a partitioning of each attribute as Mondrian does, but rather a partition of the individuals. As such, we can see their approach as a form of clustering. In the strictest form of k-anonymity, we may accept that an individual is indistinguishable from another if the generalisation of their features is included in the generalisation of the other's features, without necessarily being equal. This added degree of freedom leads to **non-homogeneous generalisation** [22], and an application to an OD-matrix is illustrated in Fig. 5. For more clarity, the data of the figure is summarised in Table 1.

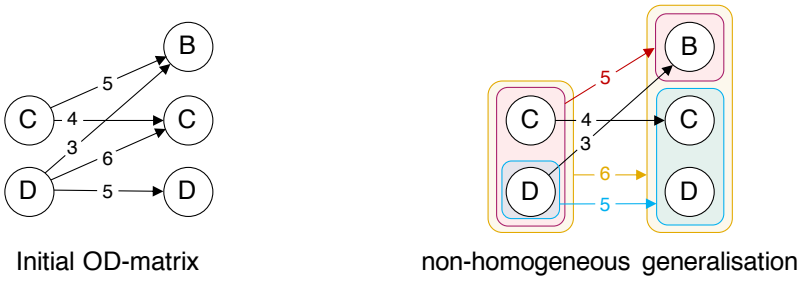


Figure 5: Left: initial OD-matrix to anonymise. Right: same flows with their generalised areas of origin and destination.

Doka et al. [23] formulate non-homogeneous generalisation as a network-flow problem and propose to solve it with an exact algorithm, as well as with a greedy approach that runs in $O(kN^2)$, N being the number of distinct individuals. The Doka approach suffers from this impractical complexity, while proposing an information loss only marginally better than the one obtained by Liang and Samavi. To the author's knowledge, no work

volume	initial origin	initial destination	generalised origin	generalised destination
5	C	B	{C, D}	B
4	C	C	C	C
3	D	B	D	B
6	D	C	{C, D}	{B, C, D}
5	D	D	D	{C, D}

Table 1: Flows of Fig. 5 and their non-homogeneous generalisation. Note how any particular initial flow has its origin and destination included in enough generalised flows so that the cumulated volume is above $k = 10$.

has yet formulated generalisation and suppression restricted to a generalisation hierarchy as an optimisation problem.

2.3 Differential Privacy

Differential privacy is a robust privacy principle introduced by Dwork et al. [24] that does not apply to a dataset in itself but rather to a randomized algorithm taking the dataset as input and returning a series of query results. We say that the algorithm respects differential privacy if for any possible output, the probabilities of returning this output given that any particular individual is in the input dataset or not cannot differ by more than a fixed threshold. This threshold is characterized by a parameter $\epsilon > 0$, which allows to exactly quantify how much we want to enforce privacy. This definition is formalised in Eq. 1, stating that a randomized algorithm $\mathcal{M}$ is ϵ-differentially private if for all subset $\mathcal{S}$ of the possible values returned by $\mathcal{M}$, and for all couples (x, y) of datasets differing by at most one row:

$$\Pr[\mathcal{M}(x) \in \mathcal{S}] \leq \exp(\epsilon)\Pr[\mathcal{M}(y) \in \mathcal{S}]. \tag{1}$$

Where $\Pr[\mathcal{M}(x) \in \mathcal{S}]$ and $\Pr[\mathcal{M}(y) \in \mathcal{S}]$ are the probabilities of $\mathcal{M}$ returning a value belonging to $\mathcal{S}$ given the input datasets x and y, respectively. Note that the factor $\exp(\epsilon)$ is greater than 1 for any $\epsilon > 0$, giving us a two-sided bounding of the ratio of probabilities.

Applied to an OD-matrix, the most straightforward way of satisfying differential privacy is to add to each flow a Laplacian noise with a default parameter of 1 [25]. As it would not make sense to have negative or non-integer flows, we round the flows and keep only the positive ones, without hampering the differential privacy guarantee [26]. This noise must be added to 0-flows in order to respect the differential privacy property, and the probability of it resulting in a non-zero noisy flow (given that the input is a 0-flow) is 30.33%. However, OD-matrices are known to be very sparse, and the density of our datasets range from 0.24% to 4.4%. As a result, a differentially private OD-matrix would be between 6 and and 126 times more costly to store than the original matrix [26], and would mostly consist of blank noise. More critically, investigations into its relevance with respect to GDPR [27] find that it does not guarantee the safety of the dataset in itself, nor does it protect against false re-identifications, which should be prevented as well as true ones. For a comprehensive survey on differential privacy in the general case, see [28, 25].

2.4 Positioning

The presented solutions tend to focus on finding the finest generalisation that respects k-anonymity, while keeping a reasonable computing time for small-scale datasets. Yet mobile phone data offer volumes way above the ones usually considered, to which few solutions from the state of the art can actually scale. For example, for a dataset of 100,000 records, 4 attributes and $k = 5$, Liang and Samavi's fastest model *Split & Carry* [7] takes hours to complete: the computing time being exponential with respect to k, it may amount to days for $k = 10$. These time scales are orders of magnitude above the computing time we aim for in the benchmark. We propose in this paper a simplified context for the optimisation problem, which makes an Adaptative Tree Generalisation specifically adapted for OD-matrices, with their particularity of being exceptionally low-dimensional, high-volume, with a high number of modalities. Our ATG-Dual approach can handle high volumes while being significantly finer than the faster approaches of uniform generalisation. We also propose ATG-Soft, a cheaper version that sacrifices precision in favor of computing time. Fig. 6 summarises the positions of the approaches of the state of the art with respect to the degree of freedom they allow in the solution: uniform tree generalisation restricts the solution space the most, followed by our approach and then Mondrian, which all find partitions of the attributes in order to form clusters. Methods that find a clustering that does not necessarily rely on a partitioning of the attributes have a broader search space. Finally, non-homogeneous generalisation, which does not necessarily produce a clustering, has an even broader search space.

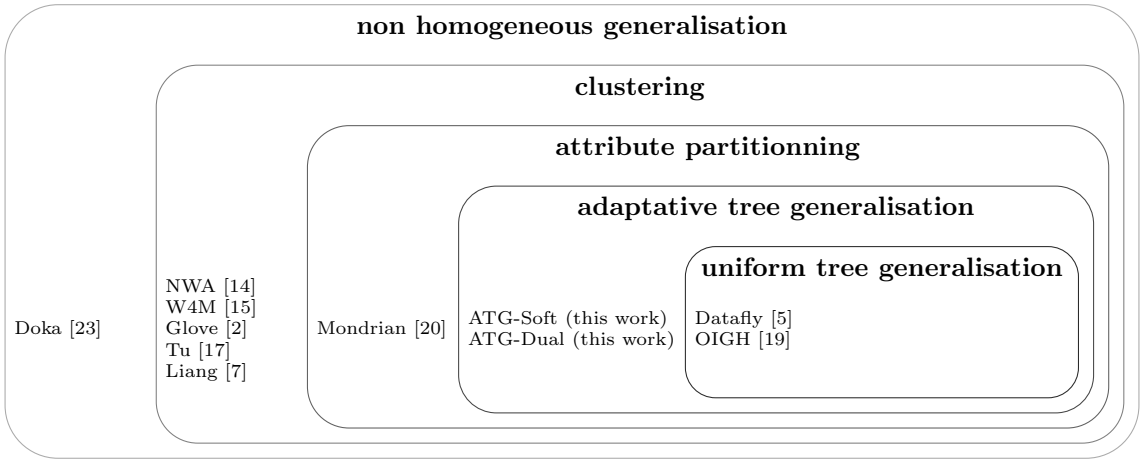


Figure 6: Classification of k-anonymisation approaches with respect to the space of solutions they consider.

3 Methodology

3.1 Overview

We consider an OD-matrix, defined on a set of initial tiles and equipped with a generalisation hierarchy that describes a local order in which we can aggregate the initial tiles to form generalised areas. We call the hierarchy T, and we use the same hierarchy over the origins and the destinations. Each leaf of T represents an initial tile, and each node $n \in T$ represents the area obtained by the union of the children of n. A pruning of T is a tree obtained by pruning out some branches of T (*i.e.,* a node and all the nodes below it). We interpret such a pruning as the set of nodes we wish to *split*, so that their children, if not split, represent a partitioning of the set of initial tiles. This is illustrated in Fig. 7: the pruning defined by the nodes in grey is interpreted as a partitioning defined by the nodes in white in the right-most figure.

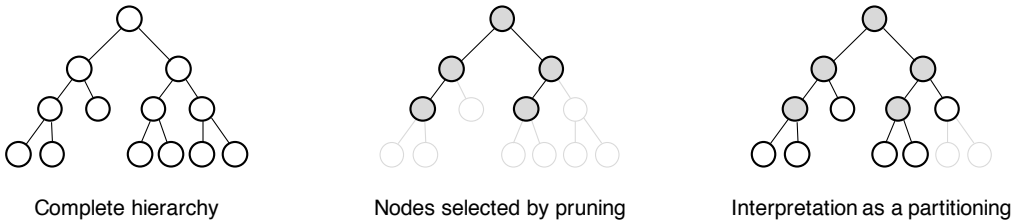


Figure 7: Left: Example of a complete generalisation hierarchy. Middle: Example of a pruning of the hierarchy, satisfying the tree constraint. Right: Interpretation of the pruning as a partitioning of the values.

Formally, we note $(x_n)_{n \in T}$ the binary variable that describes a pruning of T. In this section, we will use the **tree constraint**, that holds if and only if the set of nodes n such that $x_n = 1$ is a pruning of T:

$$\text{tree constraint}(\mathbf{x}, T) = \begin{cases} \forall n \in T, x_n \in \{0,1\} \\ \forall n \in T, x_{p(n)} \geq x_n \\ \forall n \in leaves(T), x_n = 0 \\ x_{p(root(T))} = 1 \text{ by convention} \end{cases}$$

where $p(n)$ denotes the parent of node n in the tree T. As we cannot split leaves of the hierarchy, x_n is necessarily 0 for leaves of T. Let $\mathcal{O}$ be a partitioning of origins, and for each $o \in \mathcal{O}$, $\mathcal{D}_o$ be a partitioning of destinations. For each $o \in \mathcal{O}$, $d \in \mathcal{D}_o$, we note $o \to d$ the flow going from o to d, $v_{o \to d}$ the volume of the flow (measured in number of people, obtained by summing the initial flows), and $|o|$ and $|d|$ the sizes of the origin and destination, measured in number of initial tiles. We consider the total generalisation error G, derived from

the objective functions used in the state of the art [23, 7]:

$$G = \sum_{o \in \mathcal{O}} \sum_{\substack{d \in \mathcal{D}_o \\ v_{o \to d} \geq k}} (|o| + |d|) v_{o \to d}. \tag{G}$$

The generalisation error G penalises the size of the origin and destination for each individual that has not been suppressed. As we have to suppress any individual that has not been k-anonymised, the amount of suppression is given by:

$$S = \sum_{o \in \mathcal{O}} \sum_{\substack{d \in \mathcal{D}_o \\ v_{o \to d} < k}} v_{o \to d}. \tag{S}$$

We now consider the problem of finding partitionings of origins and of destinations that minimize G while keeping S under a suppression constraint C. The formulation of this problem is given for reference:

Problem (Coupled generalisation under maximal suppression constraints)**.** *Let $\mathbf{x} = (x_o)_{o \in T}$ represent an origin's pruning, and for each possible origin $o \in T$, $\mathbf{y_o} = (y_{o,d})_{d \in T}$ represent one destination's pruning. Then the coupled partitioning problem under maximal suppression constraint can be cast as:*

$$\begin{aligned} &\min_{x,y} \sum_{\substack{o \in T, d \in T \\ v_{o \to d} \geq k}} (x_{p(o)} - x_o)(y_{o,p(d)} - y_{o,d})(|o| + |d|) v_{o \to d} \\ &s.t. \begin{cases} \text{tree constraint}(\mathbf{x}, T) \\ \forall o, \text{ tree constraint}(\mathbf{y_o}, T) \\ \sum_{\substack{o \in T, d \in T \\ v_{o \to d} < k}} (x_{p(d)} - x_d)(y_{o,p(d)} - y_{o,d}) v_{o \to d} \leq C \end{cases} \end{aligned} \tag{J}$$

The term $(x_{p(o)} - x_o)$ ensures that we only count the penalties of the leaves of the origin pruning, and the term $(y_{o,p(d)} - y_{o,d})$ ensures that we only count the penalties of the leaves of the destination pruning.

Problem J is already a restriction on the problem of generalisation of flows, yet it remains hard and computationally expensive to solve. In order to simplify it, we uncouple it in two separate problems: we first formulate the **best pruning problem** to find a map of origins $\mathcal{O}$ with the heuristic that origins $o \in \mathcal{O}$ should emit an outgoing volume close to a given target volume v_{target}. As the k-anonymisation is coarser for problems with fewer individuals, this heuristic ensures that the destination maps for different origins will be close in coarseness. It is best justified with the assumption that the flows have similar behaviors regardless of their origins. Then, for each origin $o \in \mathcal{O}$, we can more easily find a partitioning of destinations that minimizes the total generalisation error G under the suppression constraint. These problems are formulated as variations of the best pruning problem.

3.2 Best pruning problem

The best pruning problem considers that generalising values to a node n induces a penalty π_n. Finding the best partitioning then amounts to minimizing the sum of penalties of the leaves of the pruned tree. The resulting problem is given by:

Problem (Best pruning problem)**.** *Let x represent a pruning of T, then the best pruning problem for the penalty function π is defined by:*

$$\begin{aligned} &\min_{x} \sum_{n \in T} (x_{p(n)} - x_n) \pi_n \\ &s.t. \text{ tree constraint}(\mathbf{x}, T) \end{aligned} \tag{P}$$

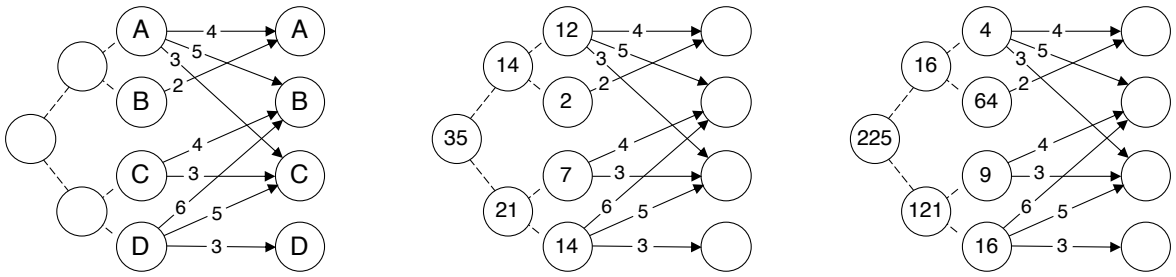
The term $(x_{p(n)} - x_n)$ ensures that we only count the penalties of the leaves of the pruned tree.

If the expression chosen for π_n depends only on the $x_{n'}$ such that n' is a descendant of n, then solving problem P is rather straightforward: by making a simple pass through T, one can recursively choose for each node if it is better to split it or not, with the corresponding best-case penalty. See appendix I for the detailed algorithm. In the case of generalisation of origins, we set π_n so that it measures how far the volume of flows coming out of n is from our target volume v_{target}. We choose the expression:

$$\pi_n = \left(v_{target} - \sum_{d \in leaves(T)} v_{n \to d} \right)^2.$$

An illustration of the corresponding values for π_n is given in Fig. 8. We see that for each node, we can choose either to accept its penalty or accept the sum of the penalties of its children.

Then, we use a variation of Prob. P to find a generalisation of destinations that minimizes the total generalisation error G given a fixed $\mathcal{O}$, under the suppression constraint C.



Initial OD-matrix | emission flows | leaf penalty ($v_{target} = 10$) $\pi_n = (v_{target} - v_n)^2$

Figure 8: Left: example of an OD-matrix with the hierarchy over the origins. Middle: emission flows from each initial and generalised origin. Right: corresponding penalty obtained by setting $\pi_n = \left(v_{target} - \sum_{d \in leaves(T)} v_{n \to d}\right)^2$ with $v_{target} = 10$.

3.3 Pruning problem with suppression constraint

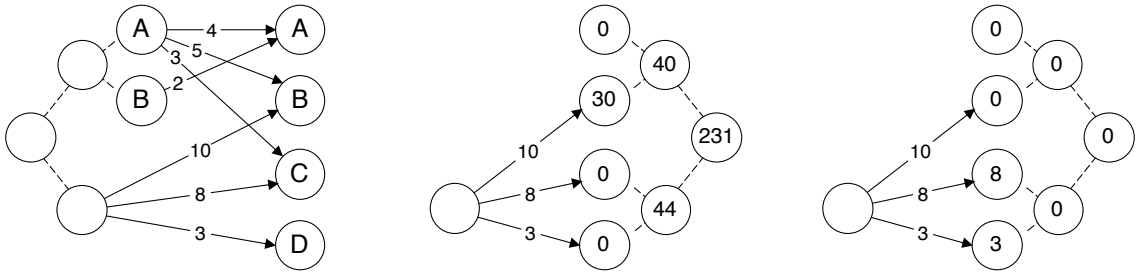
Let $o \in \mathcal{O}$ be an origin. For each $d \in T$, we define the aggregation penalty α_d, equal to the contribution of node d in the expression of G:

$$\alpha_d = \begin{cases} (|o| + |d|)v_{o \to d} & \text{if } v_{o \to d} \geq k \\ 0 & \text{else} \end{cases}, \tag{3.1}$$

and the suppression penalty σ_d, equal to the number of records that would be suppressed if d was a leaf of the pruned tree:

$$\sigma_d = \begin{cases} 0 & \text{if } v_{o \to d} \geq k \\ v_{o \to d} & \text{else} \end{cases}. \tag{3.2}$$

Examples of values for α_d and σ_d are given in Fig. 9: the size of the origin is 2 as it is the result of aggregating the two areas C and D together. We see that the values of α_d increase as we go up in the hierarchy. The lowest total aggregation penalty is achieved by aggregating nothing, but it may imply suppressing more volume than is acceptable.



Initial OD-matrix | aggregation penalties | suppressed volumes (k=10)

Figure 9: Left: example of an OD-matrix with the hierarchy over the destinations. Middle: aggregation penalty for each destination for one given origin. Right: suppressed volumes for each destination, considering $k = 10$.

Minimizing G with fixed origin o amounts to minimizing the sum of α_d while keeping the sum of σ_d under a suppression constraint C. The resulting problem is given by:

Problem (Best pruning problem under hard suppression constraints)**.** *Let x represent a pruning of T and α_d and σ_d be defined according to Equations 3.1 and 3.2 respectively, then finding the best pruning of T for a fixed*

origin o is equivalent to:

$$\begin{aligned} &\min_x \sum_{d\in T}(x_{p(d)} - x_d)\alpha_d \\ &s.t. \begin{cases} \text{tree constraint}(\mathbf{x}, T) \\ \sum_{d\in T}(x_{p(d)} - x_d)\sigma_d \leq C \end{cases} \end{aligned} \tag{H}$$

The term $(x_{p(d)} - x_d)$ ensures that we only count the penalties of the leaves of the pruned tree.

Problem H is better formulated as a variant of a knapsack problem, where the benefit of each node d is the gain in aggregation penalty due to splitting d and the weight of node d is the additional suppression induced by splitting.

Problem (Knapsack formulation of the best pruning problem under hard suppression constraints)**.** *Let $b_d = \alpha_d - \sum_{c\in children(d)} \alpha_c$ be the gain due to splitting d and $w_d = \sum_{c\in children(d)} \sigma_c - \sigma_d$ the weight of destination d. Then the knapsack formulation of H is:*

$$\begin{aligned} &\max_x \sum_{d\in T} x_d b_d \\ &s.t. \begin{cases} \text{tree constraint}(\mathbf{x}, T) \\ \sum_{x\in T} x_d w_d \leq C \end{cases} \end{aligned} \tag{K}$$

With this formulation, the problem has been studied under the name of the Tree knapsack problem [29, 30] or ordered knapsack problem [31]. A variant where the hierarchy is not necessarily a tree is known as the Precedence Constraint Knapsack Problem (PCKP) [32], and has been extensively studied in the field of open-pit mining [33, 34]. The tree knapsack problem can be easily solved by dynamic programming for trees under a few hundred nodes, but for larger trees it is recommended to consider its dual [33]. Equivalently, we directly consider the dual of our original formulation H.

Problem (Dual formulation of best pruning problem under hard suppression constraints)**.** *The dual of problem H obtained by relaxation of the suppression constraint is given by:*

$$\begin{aligned} &\max_\lambda \min_x \sum_{d\in T}(x_{p(d)} - x_d)(\alpha_d + \lambda\sigma_d) - \lambda C \\ &s.t.\ tree\ constraint(x, T) \end{aligned} \tag{D}$$

We can prove that the optimal solution of problem D is feasible for problem H (see appendix F), meaning that the solution respects the hard constraint of no more than C records suppressed. This is a variation on the dual of the PCKP, as we obtain a maximisation problem instead of a minimization problem. We can still solve it with the All Breakpoints Algorithm [33]: This algorithm relies on the fact that the lagrangian function that we aim to maximize:

$$L : \lambda \mapsto \min_x \sum_{d\in T}(x_{p(d)} - x_d)(\alpha_d + \lambda\sigma_d) - \lambda C,$$

is continuous, piecewise linear, and characterized by its **breakpoints** (Fig. 10, left).

This list of breakpoints can be recursively computed by a single pass through T, defining for each node n the lagrangian that would be obtained for a problem set on the branch starting from n. We call this function the **Best Branch Error** of n $B_n(\lambda)$. The Best Branch Error of the root $B_{root(T)}$ is then the lagrangian function L. See appendix D for the definition and properties of the best branch error. However, even though B_n can be determined by a single pass through the branch of n, the computing time also depends on the number of breakpoints, which can be high. The Specific Breakpoints Algorithm (SBA) [34] improves over the All Breakpoints Algorithm. It takes advantage of the piecewise linearity and convexity (in this case, concavity) of the lagrangian function to perform a search akin to dichotomy, which requires evaluating the lagrangian function for only a small number of points (Fig. 10, right). See appendix J for the detailed algorithm.

Evaluating the lagrangian function on a single value λ is equivalent to solving problem P with $\pi_d = \alpha_d + \lambda\sigma_d$, eventually adding $-\lambda C$. We call this variation the soft suppression constraint, which is formulated as:

Problem (Best pruning problem with soft suppression constraint)**.**

$$\begin{aligned} &\min_x \sum_{d\in T}(x_{p(d)} - x_d)(\alpha_d + \lambda\sigma_d) - \lambda C \\ &s.t.\ tree\ constraint(x, T) \end{aligned} \quad . \tag{H'}$$

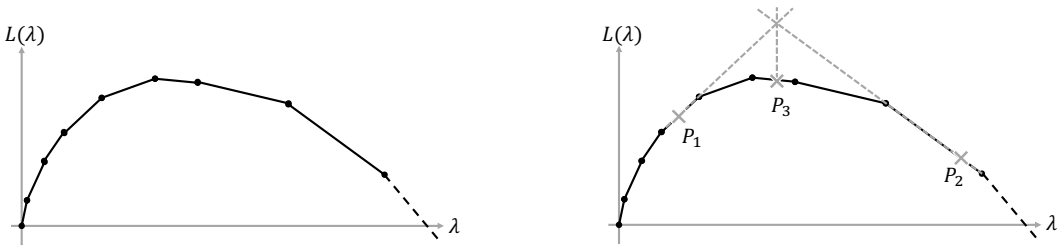


Figure 10: Left: General shape of the lagrangian function of problem D. Right: Geometrical interpretation of one step of the algorithm SBA used to find the maximum.

Note that this is also closer to minimizing a common definition of the generalisation error [7], which contrary to our definition of G, penalizes suppressed records as if they were generalised to the maximum level. Here, suppressed records are penalized as if they were generalised to the level λ, and we can interpret λ as an area. The optimal solution for problem H' has an interesting property: we can prove that it will never imply the generalisation of a flow $o \to d$ such that $|o| + |d| > \lambda$ (see appendix E). As such, we lose the hard constraint on the suppressed volume but gain a hard constraint on the level of generalisation, which can also be of interest for data owners.

3.4 Global suppression constraint

Solving problem H or D separately for each origin $o \in \mathcal{O}$ requires distributing the suppression constraint C so that the solutions together do not suppress more than a volume C. Choosing such a distribution is in itself an optimisation problem. We can instead solve for all the destinations at once, by considering a tree $\tilde{T}$ made of $|\mathcal{O}|$ times the same hierarchy T, connected together with a dummy root as illustrated in Fig. 11. Each subtree represents the destination map of an origin, and thus solving the problem on tree $\tilde{T}$ yields one destination map for each origin in $\mathcal{O}$. Solving the problem for $\tilde{T}$ instead of T is computationally a lot more demanding, hence the importance of problem D solved with SBA, which scales very well for large trees.

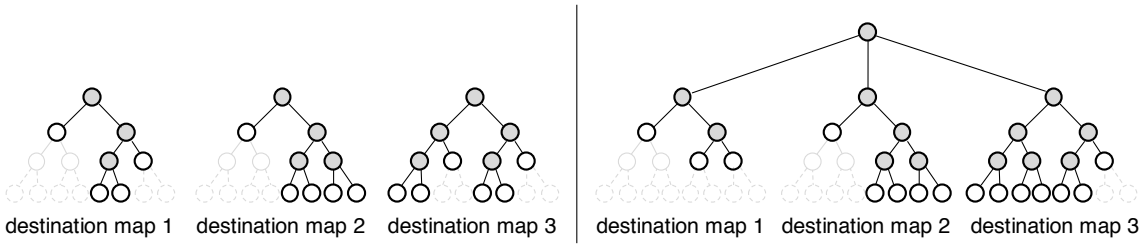


Figure 11: Left: separately generalising the destination map of each origin. Right: solving for the global problem under the unified constraint C

3.5 Parameter choices

Defining a generalisation hierarchy. The hierarchy T is central to our approach. To ensure the best results, it could be valuable to manually define the areas and organise the order in which they ought to be aggregated, in order to give priority to areas that are known to share the same land usage. In this study, we use a satisfying automated approach, using the implicit spatial partitioning of the data when available, *e.g.,* Voronoi tessellation of base stations for mobile phone data. We then run a hierarchical agglomerative clustering on the centroids of the initial areas, and we use the resulting dendrogram as a generalisation hierarchy.

Selection of v_{target}. The decoupling of origins and destinations introduces a hyper-parameter v_{target} that needs to be tuned. A simple criterion can be considered in order to choose a value: setting v_{target} too high will lead to origins more coarsely generalised than destinations, and the contrary for v_{target} too low. As the original goal is to produce an all-purpose anonymisation, we have no reason to favor precision towards origin or destination. Moreover, we observe through experimentation that the least G is obtained for origins and destinations of the same mean size. It is then recommended to set v_{target} such that the mean sizes of origins

and destinations are the closest. The best value depends on the number of areas in the study zone, the total volume of the matrix, and the structure of the flows. In this study, we set a single v_{target} for each dataset, selected via a grid-search on a small sample of matrices.

3.6 Proposed models

Based on the problems we formulated, we propose two approaches to k-anonymise OD-matrices through generalisation and suppression:

- Adaptative Tree Generalisation **(ATG)-Dual**: Uncoupling the problem and solving the dual problem D for destinations with the Specific Breakpoints Algorithm. The hyper-parameter v_{target} is selected via a grid-search on a small sample of matrices. The implementation is detailed in appendix J.
- **ATG-Soft**: As the solving of the soft problem H' also yields results with interesting properties, we include it for information with λ set to 10% of the map size. Note that setting λ to 100% of the map size would be equivalent to using common state-of-the art definitions of the generalisation error, which penalize suppressed records as if they were generalised to the maximum possible level. However, this also leads to prohibitively high suppression costs with solutions that suppress almost nothing. This illustrates that the number of modalities in mobility data is far greater than in other domains, as this problem does not arise in the anonymisation of regular data.

4 Experiments

In this section, we first present the datasets on which we run the experiments, then the benchmark against which we compare our approaches. We discuss the performance indicators we will use to compare the solutions, then we present our results.

4.1 Datasets

We evaluate the various approaches on a selection of datasets obtained from open data and from mobile traces. We denote by `nyc` the dataset from the New York Taxi and Limousine Commission (TLC)[1]. It is set on the definitions of 263 neighborhoods by the NYC Department of City Planning, and the generalisation hierarchy is defined as the dendrogram given by a hierarchical clustering applied on the centroids of the neighborhoods. The other problems considered are derived from the mobile phone data from the Data for Development (D4D) challenge [12, 13].

With the main idea of generating an OD-matrix without consideration of its actual ground truth, we consider any transition between base stations to be an actual trip, without pre-processing. The OD-matrices are then defined as the trips made between base stations over a one-hour period. For the generalisation hierarchy, we use the dendrogram given by a hierarchical clustering applied on the base stations. This gives the datasets `civ` and `senegal`. Several variations on the senegal dataset are also considered: `senegal_crop` with only the 599 eastern-most base stations, which account for most of the activity; `senegal_big` which increases the volumes in the data by summing together timesteps that pertain to different sets of individuals; and `senegal_split` which artificially increases the number of initial tiles by dividing the base stations in four and distributing the flows non-uniformly between them. Table 2 summarises the main characteristics of the datasets: we detail in particular the *density* of the matrix, computed as the number of non-null flows over the total possible number of distinct flows (*i.e.,* the squared number of tiles), averaged over the period of observation, and the *percentage of flows that are 10-anonymous* in the original data. The relatively low amount of these 10-anonymous flows corresponds to a significant share of the total volumes of the data, as they are naturally the biggest flows. The part of the anonymous volume in the original data is given under the column *%anon_vol*. See appendix A for more details.

4.2 Benchmark

We compare our solution to **Glove**, which is the state of the art for k-anonymisation of mobility data. As our approach does not explicitly implement l-diversity and t-closeness, we could not measure the benefits offered by Tu [17] which would only give a coarser generalisation than Glove at a higher computing cost. Glove is drastically simplified when applied to simple ODs, seen as 2-point trajectories without timestamps. In that case, it is almost equivalent to the greedy clustering underlying the approach, *i.e.,* to a regular hierarchical agglomerative clustering (HAC) on the four-dimension points defined by the coordinates of origins and destinations, with complete linkage and L_1 metric. As such, we also propose **Glove-sk**, a simple implementation of Glove using scikit-learn [35] making use of the efficient implementation of HAC to get a full dendrogram,

[1] publicly available at: https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page

name	#matrices	#tiles	average #flows	density	average volume	%anon. flows	%anon. vol
nyc	7242	263	3058	4.4%	15009	15.8%	59.0%
civ	632	1221	3523	0.2%	3117	0.3%	2.6%
senegal	6752	1666	18276	0.7%	41027	5.7%	36.8%
senegal_crop	6528	599	13710	3.8%	29614	5.8%	37.8%
senegal_split	360	6664	305194	0.7%	956742	6.5%	46.0%
senegal_big	360	1666	100322	3.6%	956742	12.3%	80.7%

Table 2: Descriptive statistics of the datasets used for the experiments (#matrices: number of matrices available in the whole dataset, where each matrix represents the flows over a time step; #tiles: number of initial tiles over which the matrices are set; density: average graph density of the matrices; avg. #flows: average number of flows among the matrices in the dataset; avg. vol.: average sum of the flows; %anon. flows: fraction of flows that are above $k = 10$; %anon. vol: fraction of individuals that are in flows above $k = 10$). Note that due to performance concerns, we do not evaluate the benchmark on all available matrices and we rather choose a small sample of matrices for each dataset.

and then choosing the lowest cut that grants k-anonymity to all but C records. Unfortunately, because it can only be agnostic of the volumes of the flows, scikit-learn's HAC performs appallingly badly compared to actual Glove. We still include it in the benchmark as it is by far the most readily available solution for a data owner, running in a reasonable amount of time for data that are not as high-volume as the ones featured in this study. As the implementation offered by scikit-learn benefits from state of the art code optimization, it also acts as a lower bound for the computing time we could expect from glove in the case of the best possible implementation.

Although **Mondrian** has been found to perform very poorly on trajectories [15], we observed that it produces relevant results for OD-matrices. We compare our approaches to our own python implementation of Mondrian, which performs cuts in the coordinates of the flows seen as 4D points. We also compare our own python implementation of uniform generalisation with the **OIGH** algorithm, which gives horizontal cuts in the hierarchies of origins and destinations. The hierarchies we use, derived from dendrograms, do not have all leaves at the same tree depth, which is originally not handled by OIGH. In order to accommodate those hierarchies, we made slight adaptations of OIGH, that we detail in appendix B. For comparison, and although differential privacy does not answer our problem, we compare k-anonymisation to (1,0)-differential privacy reached through **laplace noise**. The results obtained by **suppression** alone are also shown in the result tables.

For reproducibility purposes, we include in appendix B the complete details of our implementations, along with information on the adaptations we made to their original formulations.

4.3 Performance indicators

For the choice of performance indicators, we bore in mind that the particular generalised areas we provide will most likely not be of interest for the final users of the data. It is reasonable to assume that data users will rather be interested in drawing their own areas of interest and querying the matrix for an estimation of the flows between these areas. This is especially true as the various anonymised OD-matrices that represent different timesteps from a single dataset will feature generalisations that are *a priori* inconsistent, and comparing the timesteps requires projecting them on a single static zoning. In the absence of additional information, the best estimation flow between two arbitrary areas is obtained by considering the volumes to be uniformly distributed in their generalised areas, and as a result the flow between the arbitrary areas is proportional to the overlap with the generalised areas. We call this estimation process **reconstruction**. For more details on the use of the reconstruction process for temporal analysis, see appendix H

It is then relevant to actually reconstruct the anonymised OD-matrix and evaluate the difference with the original. It amounts to considering that k-anonymisation is obtained by the addition of a kind of reconstruction noise, which makes it a good way of comparing k-anonymisation with the laplacian noise used for differential privacy. In the results, we report the **reconstruction loss** E computed as the absolute difference across all flows:

$$E = \frac{1}{V} \sum_{o,d \in leaves(T)} |\tilde{v}_{o \to d} - v_{o \to d}|, \tag{2}$$

where we denote $\tilde{v}_{o \to d}$ the reconstructed volume over these initial tiles, and we normalize by the total volume of the flows $V = \sum_{o,d \in leaves(T)} v_{o \to d}$ for readability. Note that the sum is over the leaves of T, corresponding

to the initial tiles over which the OD-matrix is defined. For a discussion on the limits of the reconstruction loss, see appendix G.

We also evaluate the approaches with respect to their total generalisation error G, as defined in Eq. G. Approaches such as Glove and Mondrian that do not rely on a generalisation hierarchy do not have a clearly defined value for G. As they use an axis-aligned bounding box as a reference for their clusters, we count all the tiles intersecting with the box in the size of the generalised area. We give here a general expression for G that is more readily applicable to all generalisation methods, defined over any set $\mathcal{F}^+$ of anonymous flows $o \to d$ such that $v_{o\to d} \geq k$. The value we report in the results is the **mean generalisation error** $\bar{G}$ across matrices, given by:

$$\bar{G} = \frac{1}{V^+} \sum_{o\to d \in \mathcal{F}^+} (|o| + |d|) v_{o\to d}, \tag{3}$$

with $V^+ = \sum_{o\to d\in\mathcal{F}^+} v_{o\to d}$ the total volume of anonymised flows. When the origins and destinations are aggregated to roughly the same level as is normally the case, $\bar{G}$ represents roughly twice the number of tiles in the origin or destination of the average generalised flow. A value of $\bar{G} = 2$ then means that no generalisation was performed.

Finally, as a complementary metric to measure the distortion induced by the anonymisation, we evaluate the **distribution distance** D between each OD matrix and its anonymised versions. This metric is slightly different than the reconstruction loss E as it considers OD-matrices as normalised distributions, not penalising suppression. The distribution distance is given by:

$$D = \sum_{o,d\in leaves(T)} \left| \frac{\tilde{v}_{o\to d}}{V^+} - \frac{v_{o\to d}}{V} \right| \tag{4}$$

with the same notation as the previous definitions.

4.4 Results

We compare the approaches on a restricted number of OD-matrices on the available datasets. For each dataset, the matrices were chosen at random among matrices with a total volume of more than 5000. For each matrix, we set the suppression constraint C to be 10% of the total volume and we set $k = 10$ in accordance with the value accepted by the French regulator CNIL for OD-matrices. Fig 12 gives a dataset by dataset comparison of our ATG-Dual approach versus ATG-Soft, OIGH, Glove, and Glove-sk on metrics $\bar{G}$, E, distribution, and time of computing. For each dataset, we represent the distribution of the performance of the benchmark solver, expressed as the ratio with the performance of ATG-Dual. The line $y = 1$ indicates matrices for which ATG-Dual and the benchmark have the same value, and the line $y = 10$ indicates matrices for which the competitor's metric is 10 times higher than ATG-Dual. As it is best to have a low error and a low computing time, ATG-Dual is better when the distributions are above the line $y = 1$.

Unsurprisingly, we see that the ATG-Soft is faster but coarser than ATG-Dual, as it is essentially a cheaper version that uses a fixed value for λ instead of finding the best one. ATG-Dual gives consistently better results than OIGH in a shorter computing time.

Upon closer inspection, we observe that OIGH is likely to waste time in its search inside the lattice of possibilities, because it yields a lot of ties that then need to be broken. Glove offers a noticeably finer generalisation than ATG-Dual. However, its prohibitive computing time makes it difficult to recommend, and its memory usage is such that we were unable to run it for our bigger datasets `senegal_split` and `senegal_big`. An optimistic take on Glove would be to consider that it has the performance of Glove with the computing time of Glove-sk. Glove-sk in itself offers remarkably bad results, which shows the importance of the volume awareness in the hierarchical clustering proposed by Glove, that could not be handled with HAC. We include it in the comparison as it gives an estimation of the computing time that a truly optimised implementation of Glove could possibly offer. Yet, its computing time is still unsatisfactory compared to our approaches. Mondrian finds generalisations that are coarser than our approach for all datasets except `civ`, but the reconstruction error is much more balanced, with an advantage for Mondrian. This is at least partly explained by the fact that Mondrian does not suppress volume, which has a direct impact on the absolute errors measured in E. As Mondrian requires repeatedly counting volumes on both sides of the cuts we consider, it is much slower than ATG-Dual. Some implementation improvement could be considered in order to optimise the counting steps, but it is unlikely that it could achieve a 100-fold reduction in computing time.

Table 3 summarises the average performance over all matrices in the small datasets `nyc`, `civ`, `senegal` and `senegal_crop`, for which we were able to run Glove. We report the mean value of $\bar{G}$, E, D and S as well as the total computing time to anonymise all the matrices. We see that Glove stands out for its lack of scalability, making it impractical in huge-volume cases. Among the other solutions, a difference of minutes that we measure in the computing time is relevant but not decisive. They must rather be compared based on $\bar{G}$, for which ATG-Dual offers a significant improvement over the state of the art. The better E offered by Mondrian is partly explained by its not suppressing volumes. Each unit of missing volume has a contribution of 1 in the

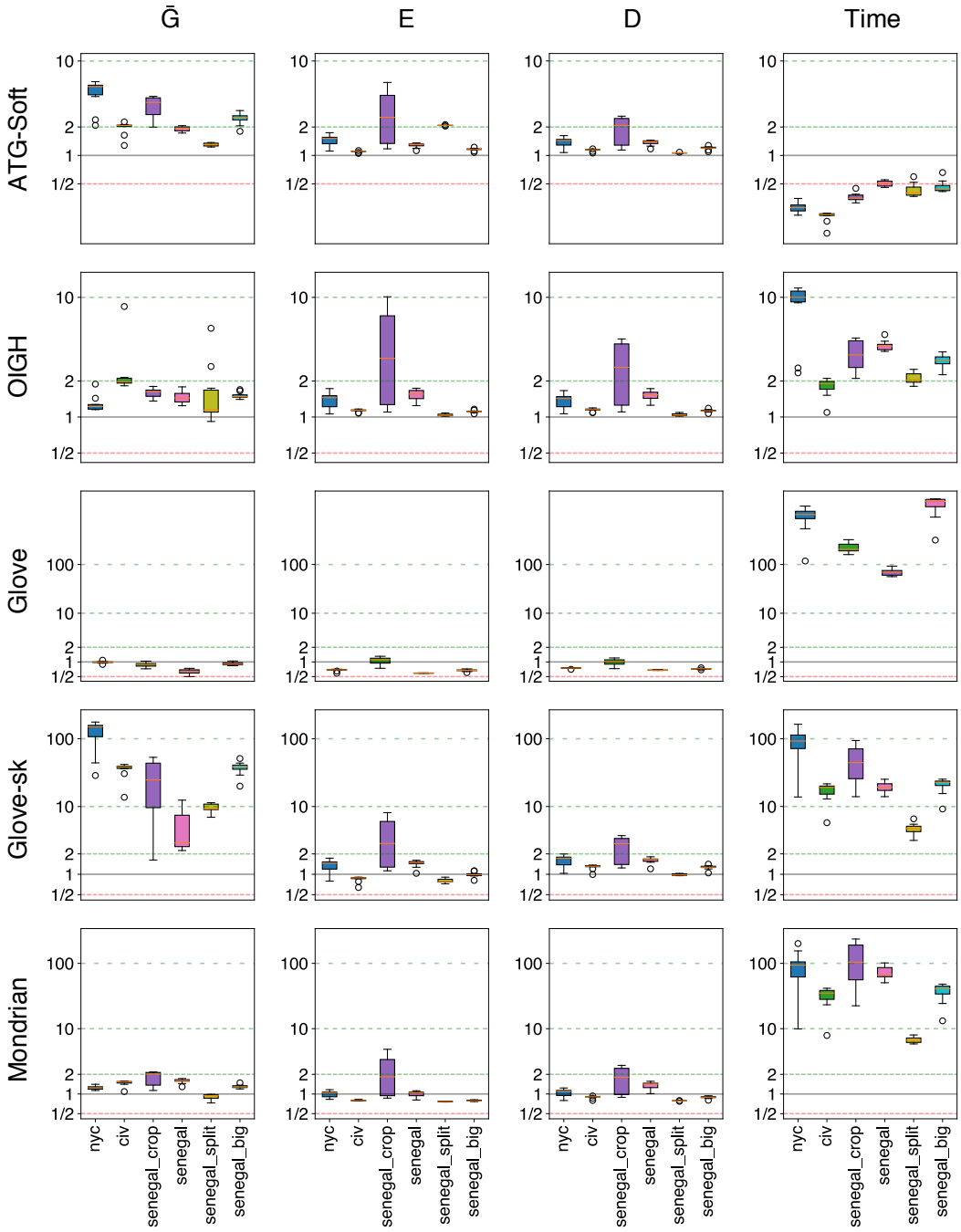


Figure 12: Comparison of the performance of our ATG-Dual approach versus ATG-Soft, Glove-sk, Glove, OIGH, and Mondrian. Each box represents the distribution of the performance over one dataset, expressed as the ratio of the benchmark over ATG-Dual. First column: comparing generalisation error G. Second column: comparing the reconstruction loss E. Third column: comparing the distribution distance D. Fourth column: comparing the computing times. In each case, the line $y = 1$ represents matrices for which the benchmark has the same performance as ATG-Dual, and the lines $y = p$ represents matrices for which the benchmark is p times worse than ATG-Dual. The box plots correspond to the datasets in this order: nyc, civ, senegal_crop, senegal, senegal_split, senegal_big.

expression of E (Eq. 2), which means that an additional 1% of suppression contributes to an increase of 0.01 in E. As Mondrian also offers a better D, we have to consider that Mondrian happens to find generalisations that are more uniform in terms of volumes. For differential privacy, E and S mostly measure the volumes that

have been added to 0-flows, as they represent more than 95% of possible flows in each OD-matrix. Because of this, differential privacy performs worse than any generalisation technique when compared on E. This matches the previous observation [26] that differential privacy is not adapted for sparse data. As the interpretation of $\bar{G}$ is only relevant for generalisation, we do not include it in the table for suppression and laplace_noise.

solver	$\bar{G}$	E	D	time (s)	S
ATG-Dual	24.90	1.18	1.14	23	9.99%
ATG-Soft	40.41	1.76	1.31	9	3.24%
oigh	53.62	1.34	1.30	56	9.87%
glove-sk	444.12	1.10	1.36	358	9.99%
glove	18.70	0.81	0.84	23203	9.98%
mondrian	27.39	0.95	1.02	688	0.00%
suppression	—	0.33	1.34	—	39.46%
laplace_noise	—	2.88	6.78	—	-170.92%

Table 3: Performance on samples of the dataset senegal_crop, nyc, civ, and senegal. $\bar{G}$: mean generalisation error (Eq. 3); E: normalized reconstruction loss (Eq. 2); S: fraction of volumes suppressed (Eq. S); D: distribution distance (Eq. 4). The reported time is the total computing time (in seconds) to run the anonymisation of all matrices of all datasets. Note that laplace_noise adds volumes, as it mostly applies a positive noise on a sparse matrix.

As Glove could not be evaluated for our biggest datasets, we compare the approaches on the same average criterias in a separate Table 4 for `senegal_split` and `senegal_big`. We see that the best computing time we could hope for Glove, given by Glove-sk, is still not satisfactory in this situation. Our approach is of particular interest here, as it offers a finer generalisation than OIGH and Mondrian for a fraction of the time. The ATG-Soft alternative is even faster, yet the difference in computing time at this scale is not of importance. This approach could become relevant for even bigger matrices, for example set on the ten thousand base stations of a mobile operator in France.

Even if our approach is more appropriate, uniform tree generalisation admittedly performs well for OD-matrix generalisation. Indeed, we could expect the best solution to show a high disparity of aggregation levels between densely and sparsely populated areas, which a uniform generalisation cannot offer. This effect is mitigated by the fact that the initial tiles already partially reflect the disparity in activity density, as they rely on base stations or administrative divisions, which are more densely distributed in populated areas. This illustrates the importance of the generalisation hierarchies, as well as the adaptation we implemented in order for OIGH to run on hierarchies whose initial leaves are not all at the same depth. For more detailed results, see appendix C.

5 Conclusion and perspectives

In this paper, we proposed to k-anonymise OD-matrices that are large-scale both in terms of size of the study area and number of flows. To that end, we developed **ATG-Dual**, a tree-based approach that formulates generalisation and suppression as an optimisation problem and finds the best adaptative generalisation, as opposed to a uniform generalisation usually found by similar tree-based approaches. For even heavier OD-matrices, we also propose **ATG-Soft**, a faster version of ATG-Dual that relies on a fixed parameter λ instead of

solver	$\bar{G}$	E	D	time (s)	S
ATG-Dual	**19.85**	**0.72**	**0.78**	111	9.92%
ATG-Soft	65.39	1.07	1.09	**32**	1.54%
oigh	29.68	1.10	1.15	985	9.79%
glove-sk	1361.51	1.04	1.25	3376	10.00%
mondrian	25.19	0.76	0.85	11541	0.00%
suppression	—	0.49	1.02	—	59.00%
laplace_noise	—	2.87	3.33	—	-86.98%

Table 4: Performance on samples of the datasets senegal_big and senegal_split.

finding the best one, sacrificing precision of generalisation for the benefit of computing speed. The formulation we adopt is indeed a restriction on the solution space we consider. State-of-the-art solutions developed for mobility consider broader search spaces as they allow any form of clustering of flows. This is a natural way of considering k-anonymity, and allowing more degrees of freedom is essential in the context of trajectory anonymisation, as they are especially hard to anonymise. Yet even with these techniques it has been observed that the valuable information cannot be expected to resist k-anonymisation for $k \geq 5$. By considering OD-matrices, which are a central input to mobility analysis while being considerably simpler than trajectories, the data can be anonymised with good preservation of information. The generalisation hierarchy then appears as a relevant restriction that is natural to the data user and considerably reduces the computing cost for huge matrices, where traditional approaches struggle to scale. Our two approaches find the best anonymisation in their respective solution spaces under a hard constraint: ATG-Dual enforces a hard constraint on the suppression, which is especially relevant in the context of OD-matrices with a high number of modalities, as the representativity of the data is essential to their value. ATG-Soft enforces a hard constraint on the size of generalisation, which is also of interest to guarantee the precision of the data. Uniform tree generalisation, while being rather scalable and offering acceptable results given the right adaptations, is still slower and coarser than our approaches for huge OD-matrices. In particular, as it was originally designed to generalise numerous attributes with very small hierarchies of less than a dozen modalities, it is only natural to focus on a finer cut in the hierarchy when we have only two attributes with thousands of modalities. Our approaches come at the cost of decoupling the attributes through an *ad hoc* process requiring a hyper-parameter v_{target}. Although in our current solution it has to be set by hand based on the results over a small subset of the data, it should be possible in future work to infer the best value for v_{target} based on characteristics of the input OD-matrix. The cost function used for the generalisation of origins could also in itself be modified in order to better anticipate the processing of destinations. Our work has the practical aim of helping the owners of large-scale OD-matrices, such as mobile operators, to efficiently and effortlessly anonymise their data. However, it is well known that k-anonymity does not protect against all types of privacy attack. The approach should guarantee l-diversity in order to ensure protection against attribute linkage attacks, meaning the generalised areas should all cover at least l locations. In this context, we may consider that each initial area is a location or that each initial area contains a varying number of points of interest counting as locations. In each case, this condition could be guaranteed in future work by adding a minimum level of aggregation as a constraint. In the broader scope of probabilistic attacks, future work should focus on achieving t-closeness, meaning the distributions of destinations given each origin should not differ too much from the global distribution of destinations, and likewise for the distributions of origins. As the criterion requires to know the whole destination distribution, it is not local: we cannot introduce an additional term in each node of the best pruning problem and try to minimize its sum. The variation of the problem to ensure t-closeness would take the form of an additional constraint akin to the suppression constraint, and likely require relaxation as well in order to efficiently solve it. Still, t-closeness would remain hard to achieve in the context of sparse distributions with high number of modalities such as OD-matrices. It may also be of interest to measure the actual privacy risks of the OD-matrices in terms of probability of success of various attack scenarios. Indeed, OD-matrices are already arguably safer than most types of personal data as the number of attributes is very limited.

The k-anonymity offered by our approach is thus a necessary first step toward a more complete solution. Still, it is also a sufficient guarantee in itself for the French regulator CNIL, in charge of enforcing GDPR in France. Currently, mobility data tend to be under-used by their owners as their huge volumes and their personal aspect make their handling costly and legally risky. Achieving cheap, fast, and foolproof anonymisation of mobility data would allow their widespread public use, ensuring the most insights possible are extracted from them in order to organise ever more efficient transportation networks.

Acknowledgment

This research is supported by the French ANR research projects MOBITIC (grant number ANR-19-CE22-0010) and PROMENADE (grant number ANR-18-CE22-0008).

References

[1] European Commission, "2018 reform of EU data protection rules." `https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex%3A32016R0679`, 2018.

[2] M. Gramaglia and M. Fiore, "Hiding mobile traffic fingerprints with glove," in *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*, CoNEXT '15, (New York, NY, USA), pp. 1–13, Association for Computing Machinery, 2015.

[3] Y.-A. Montjoye, C. Hidalgo, M. Verleysen, and V. D. Blondel, "Unique in the crowd: The privacy bounds of human mobility," *Scientific reports*, vol. 3, pp. 1–5, 03 2013.

[4] F. Asgari, A. Amrani, and M. Khouadjia, "Scaling time-dependent origin-destination matrix using growth factor model," in *2021 International Symposium on Computer Science and Intelligent Controls (ISCSIC)*, pp. 51–57, Nov 2021.

[5] L. Sweeney, "Achieving k-anonymity privacy protection using generalization and suppression," *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, vol. 10, p. 571–588, oct 2002.

[6] M. Fiore, P. Katsikouli, E. Zavou, M. Cunche, F. Fessant, D. Le Hello, U. M. Aivodji, B. Olivier, T. Quertier, and R. Stanica, "Privacy in trajectory micro-data publishing: a survey," *Transactions on Data Privacy*, vol. 13, pp. 91 – 149, 2020.

[7] Y. Liang and R. Samavi, "Optimization-based k-anonymity algorithms," *Computers & Security*, vol. 93, no. 101753, pp. 1–17, 2020.

[8] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, "L-diversity: Privacy beyond k-anonymity," *ACM Trans. Knowl. Discov. Data*, vol. 1, p. 3–15, Mar. 2007.

[9] R. Shokri, "Quantifying and protecting location privacy," *it - Information Technology*, vol. 57, pp. 257–263, 01 2015.

[10] N. Li, T. Li, and S. Venkatasubramanian, "t-closeness: Privacy beyond k-anonymity and l-diversity," in *2007 IEEE 23rd International Conference on Data Engineering*, pp. 106–115, 2007.

[11] C. Bettini, X. S. Wang, and S. Jajodia, "Protecting privacy against location-based personal identification," in *Secure Data Management* (W. Jonker and M. Petković, eds.), (Berlin, Heidelberg), pp. 185–199, Springer Berlin Heidelberg, 2005.

[12] Y.-A. de Montjoye, Z. Smoreda, R. Trinquart, C. Ziemlicki, and V. D. Blondel, "D4d-senegal: The second mobile phone data for development challenge," *ArXiv*, vol. abs/arXiv:1407.4885, 2014.

[13] V. Blondel, M. Esch, C. Chan, F. Clerot, P. Deville, E. Huens, F. Morlot, Z. Smoreda, and C. Ziemlicki, "Data for development: the d4d challenge on mobile phone data," *ArXiv*, vol. abs/1210.0137v2, 2013.

[14] O. Abul, F. Bonchi, and M. Nanni, "Never walk alone: Uncertainty for anonymity in moving objects databases," in *2008 IEEE 24th International Conference on Data Engineering*, pp. 376–385, 2008.

[15] O. Abul, F. Bonchi, and M. Nanni, "Anonymization of moving objects databases by clustering and perturbation," *Information Systems*, vol. 35, no. 8, pp. 884–910, 2010.

[16] L. Chen, M. T. Özsu, and V. Oria, "Robust and fast similarity search for moving object trajectories," in *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, SIGMOD '05, (New York, NY, USA), p. 491–502, Association for Computing Machinery, 2005.

[17] Z. Tu, K. Zhao, F. Xu, Y. Li, L. Su, and D. Jin, "Beyond k-anonymity: Protect your trajectory from semantic attack," in *2017 14th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pp. 1–9, 2017.

[18] N. Monath, K. A. Dubey, G. Guruganesh, M. Zaheer, A. Ahmed, A. McCallum, G. Mergen, M. Najork, M. Terzihan, B. Tjanaka, Y. Wang, and Y. Wu, "Scalable hierarchical agglomerative clustering," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery I& Data Mining*, KDD '21, (New York, NY, USA), p. 1245–1255, Association for Computing Machinery, 2021.

[19] W. Mahanan, W. Chaovalitwongse, and J. Natwichai, "Data privacy preservation algorithm with k-anonymity," *World Wide Web*, vol. 24, p. 1551–1561, 09 2021.

[20] K. LeFevre, D. DeWitt, and R. Ramakrishnan, "Mondrian multidimensional k-anonymity," in *Proceedings of the 22nd International Conference on Data Engineering*, vol. 2006, pp. 25 – 25, 05 2006.

[21] J. Friedman, J. Bentley, and R. Finkel, "An algorithm for finding best matches in logarithmic expected time," *ACM Trans. Math. Softw.*, vol. 3, pp. 209–226, 09 1977.

[22] W. K. Wong, N. Mamoulis, and D. W. L. Cheung, "Non-homogeneous generalization in privacy preserving data publishing," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, (New York, NY, USA), p. 747–758, Association for Computing Machinery, 2010.

[23] K. Doka, M. Xue, D. Tsoumakos, and P. Karras, "K-anonymization by freeform generalization," in *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*, ASIA CCS '15, (New York, NY, USA), p. 519–530, Association for Computing Machinery, 2015.

[24] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of Cryptography* (S. Halevi and T. Rabin, eds.), (Berlin, Heidelberg), pp. 265–284, Springer Berlin Heidelberg, 2006.

[25] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Found. Trends Theor. Comput. Sci.*, vol. 9, p. 211–407, Aug. 2014.

[26] G. Cormode, M. Procopiuc, D. Srivastava, and T. Tran, "Differentially private publication of sparse data," in *ICDT '12: Proceedings of the 15th International Conference on Database Theory*, pp. 299–311, 03 2011.

[27] J. Holzel, "Differential privacy and the gdpr," *European Data Protection Law Review*, vol. 5, pp. 184–196, 01 2019.

[28] C. Dwork, "Differential privacy: A survey of results," in *Theory and Applications of Models of Computation* (M. Agrawal, D. Du, Z. Duan, and A. Li, eds.), (Berlin, Heidelberg), pp. 1–19, Springer Berlin Heidelberg, 2008.

[29] V. D. Merwe and D. Jacobus, *The use of partitioning strategies in local access telecommunication network problems and other applications*. PhD thesis, Potchefstroom Campus of the North-West University, 2007.

[30] D. Shaw and G. Cho, "The critical-item, upper bounds, and a branch-and-bound algorithm for the tree knapsack problem," *Networks*, vol. 31, pp. 205–216, 07 1998.

[31] D. S. Johnson and K. A. Niemi, "On knapsacks, partitions, and a new dynamic programming technique for trees," *Math. Oper. Res.*, vol. 8, p. 1–14, Feb. 1983.

[32] H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack Problems*. Springer, first ed., 01 2004.

[33] J. Byun and R. Dimitrakopoulos, "An efficient algorithm for the lp relaxation of the maximal closure problem with a capacity constraint," Tech. Rep. G–2013–60, Groupe d'études et de recherche en analyse des décisions, GERAD, Montréal QC H3T 2A7, Canada, 2013.

[34] N. Maiti, P. Pathak, and B. Samanta, "An efficient algorithm for the precedence constraint knapsack problem with reference to large-scale open-pit mining pushback design," *Mining Technology*, vol. 130, pp. 1–14, 01 2021.

[35] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[36] J.-C. Picard, "Maximal closure of a graph and applications to combinatorial problems," *Management Science*, vol. 22, no. 11, pp. 1268–1272, 1976.

Appendices

A Detailing of the datasets

In this section, we detail the datasets we used in the experiments for reproductibility purposes. For each dataset, when slicing into time steps, we take the hour of departure of the trips for reference.

New York taxis with urban mapping (nyc) This dataset is comprised of all the records of yellow and green taxis from 2019. The data contains the zone of origin and zone of destination for more than 22M trips made over the year. The geography of the 263 tiles if defined by the TLC based on the definitions of neighborhood by the NYC Department of City Planning. We partition the dataset into 7,242 matrices corresponding to time slices of one hour each. The generalisation hierarchy defined on these zones is obtained as the dendrogram of a hierarchical clustering applied on the centroids of the zones, with $L2$ distance and ward linkage.

Cote d'ivoire (civ) This dataset is made from the mobiles trajectories available from the first D4D challenge [13], covering 5M users from 1^{st} December 2011 to 28^{th} April 2012, defined on 1,221 base stations. With the main idea of generating an OD-matrix without consideration of its actual ground truth, we consider any transition between antennas to be an actual trip. The OD-matrices are then defined as the trips made between antennas over a one-hour period. The initial tiles considered for these datasets are the Voronoi tessellation of the base stations projected in `epsg:2165`. The generalisation hierarchy we use is the dendrogram of a hierarchical clustering applied on the base stations, with $L2$ distance and ward linkage.

Senegal (senegal) This dataset is made from the mobiles trajectories available from the second D4D challenge [12], covering 9 million mobile users from 7^{th} January to 22^{nd} December 2013, defined on 1,666 base stations. Only 300 000 users are visible in the data at a time, and the batches roll over a two-weeks period. In the same approach than for the `civ` dataset, we consider any transition between base stations to be an actual trip, and each OD-matrix describe the trips over a one-hour period. The generalisation hierarchy is the also defined as the dendrogram of a hierarchical clustering with the same parameters as for `civ`.

West Senegal (senegal_crop) In order to understand the impact of the number of initial zones on the time of computing, we consider the senegal OD-matrices defined only on the 599 west-most antennas, the most urbanized area of Senegal.

Senegal with Voronoi cells, inflated (senegal_big) The Senegal trajectories cover 300 000 users on a rolling 2-week basis. In order to generate a heavier dataset, we make the hypothesis that people have roughly the same behavior along the year, which allows us to aggregate all matrices with the same time step modulo two weeks into one matrix comprised of the trips of 9M of users.

Senegal with artificial cells (senegal_split) In order to study the effects of bigger maps, we artificially split each senegal base station into four mock antennas. Each trip between two antennas is randomly attributed to one of the 16 possibilities of trips between the sub-antennas following a non-uniform multinomial law: the origin and the destination are independently sampled from a law of parameter $(0.4, 0.3, 0.2, 0.1)$, the sub-antennas being ordered. We chose not to assign the trips uniformly so that the best solution would not start by trivially aggregating all the sub-antennas to their original antennas, and to maintain a low graph density as would be expected from an OD-matrix defined many areas.

B Detailing of the benchmark

In this section, we expand on the implementation details of the algorithms we used for the benchmark.

Adaptative Tree Generalisation For each dataset, we select v_{target} based on a grid-search performed over a small training set. The values obtained for various values of k are detailed in table 5.

dataset	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$	$k=11$	$k=12$	$k=13$	$k=14$	$k=15$
civ	100	100	100	100	100	200	100	200	200	200	200
nyc	100	100	100	100	100	100	100	100	100	100	200
senegal						300					
senegal_big						400					
senegal_crop	200	300	300	300	300	400	400	400	400	400	400
senegal_split						500					

Table 5: Best v_target for the studied datasets for various values of k.

Glove We implement Glove in python, with the addition of a 100-nearest neighbours from scikit-learn [35]. The use of a nearest-neighbors graph significantly improves the time performance of Glove without sensibly impacting the coarseness of its generalisation. By nature, Glove does not consider a generalisation hierarchy and takes as input coordinates of the points of the trajectories, performing spatial generalisation. For this study, we assign as coordinates the coordinates of the centroids of the tiles of the map.

Glove-sk The Glove algorithm is drastically simplified when applied to simple ODs, seen as 2-point trajectories without timestamps. In that case, Glove is almost equivalent to the greedy clustering underlying the approach, *i.e.,* to a hierarchical agglomerative clustering on the four-dimensions points defined by the coordinates of origins and destinations, with complete linkage and L_1 metric. As such, we propose a simple implementation of Glove using scikit-learn [35] making use of the efficient implementation of hierarchical clustering to get a full dendrogram, and then choosing the lowest cut that grants k-anonymity to all but C records. As in our Glove implementation, we also add a 100-nearest neighbors to speed up calculations, as is recommended by scikit-learn's documentation for hierarchical clustering with high number of points. However, the adaptation does not exactly returns what Glove would, because of two main differences:

- First, the complete linkage considers the distance between two clusters to be the maximum distance between the points of each cluster. Glove considers the maximum distance between the hypercubes defined by the clusters.
- More importantly, the hierarchical clustering is agnostic of the volume and eventually suggests merging flows that are both above the anonymity threshold k. This leads to a huge generalisation overhead.

Because of this, the scikit-learn implementation of Glove performs appallingly worse than actual Glove. We still include it in the benchmark as it is by far the most readily available solution for a data owner, running in a reasonable amount of time for data that are not as high-volume as the ones featured in this study. As the implementation offered by scikit-learn benefits from state of the art code optimization, it also acts as a lower bound for the computing time we could expect from glove in the case of the best possible implementation.

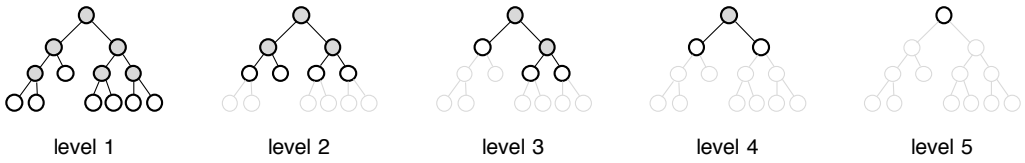


Figure 13: All generalisation levels considered by OIGH for a hierarchy that does not have all leaves at the same depth.

Mondrian We use our own python implementation of Mondrian. As it takes coordinates as input instead of qualitative areas, we assign to each area the coordinates of the centroid, as in Glove. At each step, we chose to cut the dimension that has the highest range, and we cut it to the median. If this cut implies separating a cluster into clusters that have less than a volume of k, we do not cut and we stop the algorithm in the current branch. As Mondrian forms a tree, the other branches make their own independant cuts.

OIGH The OIGH algorithm has been adapted to match the particular structure of the generalisation hierarchies of this studies: Uniform generalisations usually consider all the initial values to be at the same tree depth which is not necessarily the case for us. Some steps of OIGH could then yield generalisations that are not partitioning of the study area. In order to keep a coherent generalisation at all steps, going up one level in our implementation only generalise nodes that are of minimum-cardinal among the available parents. This problematic is illustrated in Fig. 13.

laplace-noise We implement a variation of the noise process described by Dwork [25], that results in integer, non-negative noisy flows: the recommended Laplacian noise is completed by a deterministic rounding and a thresholding to 0 for negative flows. As differential privacy is immune to post-processing [25], these operations preserve the ϵ-differential privacy property (Eq. 1) and save us the burden of storing a full matrix of noise.

C Detailing of the results

We detail the results of each solver over each dataset in table 6.

D Definition of the best pruning error $B_n(\lambda)$

In this section, we define the best pruning error of a node n in a generalisation hierarchy T, designed to solve the dual problem D. We recall that this problem involves finding the λ that maximizes the lagrangian function:

$$L : \lambda \mapsto \min_x \sum_{d \in T} (x_{p(d)} - x_d)(\alpha_d + \lambda \sigma_d) - \lambda C.$$

The best pruning error $B_n(\lambda)$ of a node n is designed to be equal to the contribution of node n and its subtree to the total $L(\lambda)$:

$$B_n(\lambda) = \begin{cases} \min \left\{ \sum_{c \in \text{children } (n)} B_c(\lambda), \alpha_n + \lambda \sigma_n \right\} & \text{if } n \text{ has children and } v_n \geq k \\ \alpha_n + \lambda \sigma_n & \text{else} \end{cases} \tag{5}$$

As $\alpha_n + \lambda \sigma_n$ is the generalisation error of node n, the expression of $B_n(\lambda)$ indicates that it takes the best choice between generalising to node n and splitting node n, when possible. It follows by induction that $B_{root(T)}(\lambda)$ is the minimal value for the generalisation penalty $\alpha_n + \lambda \sigma_n$, and that:

$$L(\lambda) = B_{root(T)}(\lambda) - \lambda C.$$

piecewise linearity As the $(x_d)_{d \in T}$ are in $\{0, 1\}$, the solution x for $L(\lambda)$ is constant in a neighborhood of λ, and the value $L(\lambda)$ is linear as long as x does not change. It is apparent then that L, and each B_n, are piecewise linear, with breakpoints corresponding to thresholds in λ that imply a change in the best solution x. Each piece of B_n correspond to a solution $(x_{n'})_{n' \in branch(n)}$, and each breakpoint correspond to two solutions.

dataset	solver	time (s)	$\bar{G}$	E	E (normalized)	S
civ	ATG-Dual	00:00:05	57.2	1.72	17.05	10.0%
civ	ATG-Soft	00:00:02	74.4	3.58	18.10	5.3%
civ	oigh	00:00:11	106.8	1.79	17.81	9.9%
civ	glove	00:06:34	35.7	1.01	11.57	10.0%
civ	glove-sk	00:00:09	542.9	1.39	16.81	9.9%
civ	mondrian	00:00:37	52.0	1.32	13.40	0.0%
nyc	ATG-Dual	00:00:01	4.7	0.49	4.75	10.0%
nyc	ATG-Soft	00:00:00	9.1	0.62	6.38	3.3%
nyc	oigh	00:00:05	7.0	0.75	7.05	9.7%
nyc	glove	00:04:42	4.2	0.52	4.61	10.0%
nyc	glove-sk	00:00:09	24.4	0.70	7.48	10.0%
nyc	mondrian	00:01:32	7.5	0.49	6.26	0.0%
senegal	ATG-Dual	00:00:09	17.4	1.21	12.00	10.0%
senegal	ATG-Soft	00:00:04	43.5	1.41	14.28	3.2%
senegal	oigh	00:00:26	26.2	1.36	13.48	9.9%
senegal	glove	04:17:38	16.0	0.82	8.66	10.0%
senegal	glove-sk	00:01:18	667.6	1.21	15.10	10.0%
senegal	mondrian	00:05:33	22.7	0.96	10.6	0.0%
senegal_big	ATG-Dual	00:00:15	3.9	0.17	5.92	9.8%
senegal_big	ATG-Soft	00:00:05	13.9	0.53	8.84	1.1%
senegal_big	oigh	00:00:54	6.2	0.79	10.32	9.5%
senegal_big	glove-sk	00:05:40	80.9	0.68	10.7	10.0%
senegal_big	mondrian	00:32:59	7.5	0.41	7.40	0.0%
senegal_crop	ATG-Dual	00:00:06	13.6	1.19	11.95	10.0%
senegal_crop	ATG-Soft	00:00:01	27.5	1.31	13.53	3.0%
senegal_crop	oigh	00:00:12	32.7	1.35	13.67	9.8%
senegal_crop	glove	01:57:47	13.5	0.82	8.87	10.0%
senegal_crop	glove-sk	00:00:52	508.0	1.04	15.10	10.0%
senegal_crop	mondrian	00:03:43	20.4	0.95	10.63	0.0%
senegal_split	ATG-Dual	00:01:35	16.0	0.75	9.76	10.0%
senegal_split	ATG-Soft	00:00:26	83.5	1.14	13.5	1.8%
senegal_split	oigh	00:15:30	19.7	1.09	12.68	9.9%
senegal_split	glove-sk	00:50:23	2304.5	1.09	14.41	10.0%
senegal_split	mondrian	02:39:21	20.2	0.78	9.66	0.0%

Table 6: Detailed results

non-decreasing property w.r.t. $\boldsymbol{\lambda}$ As increasing λ means adding more penalization, the best solution x can only have a higher value. It follows that B_n is non-decreasing with respect to λ. It can be constant, and for each n such that $v_n \geq k$ it actually assumes a constant value after a threshold λ_n^*. Indeed, intuitively, if the penalisation for suppression λ is high enough, it is preferable not to split node n, and accept the constant aggregation penalty α_n. If node n has a volume $v_n < k$, then it can only yield the suppression penalty $\lambda\sigma_n$ and its best pruning error is a linear function.

Concavity w.r.t. $\boldsymbol{\lambda}$ We prove by induction that for each $n \in T$, the best pruning error B_n is concave with respect to λ:

- If n is a leaf of T, then from the expression of B_n it is either constant or linear, so it is concave.
- For a given $n \in T$, B_n is the sum of the best pruning error of the children, which are supposed concave. Then it is maxed by a threshold, which conserves concavity.

It follows that $L : \lambda \mapsto B_{root(T)}(\lambda) - \lambda C$ is also concave. Moreover, as we saw in the previous paragraph, $B_{root(T)}$ is constant value after a threshold $\lambda^*_{root(T)}$, assuming the trivial property $v_{root(T)} \geq k$. It follows that $L(\lambda)$ is necessarily decreasing after this $\lambda^*_{root(T)}$. As it is defined for $\lambda \geq 0$, we conclude that the maximum of L is attained.

E λ is a hard constraint on aggregation level

In this section, we prove that the aggregation x^* found by solving H' respects an aggregation constraint:

$$\forall d \in T \sim leaves(T) \text{ s.t. } v_{o\to d} \geq k, |o| + |d| > \lambda \implies x^*_d = 1.$$

This property means that any volume would rather be split than aggregated to a level above λ, even if it means getting entirely suppressed. This does not concern flows whose destination cannot be split (leaves of T), or flows that are already suppressed ($v_{o\to d} < k$).

Proof. Let $\lambda \geq 0$, $d \in T$, $o \to d$ a flow such that $|o| + |d| > \lambda$. We note $\mathcal{C}$ the set of the children of d. We suppose d is not a leaf of T, so $\mathcal{C} \neq \emptyset$. Recall that in the context of OD aggregation, the aggregation penalty of node d is:

$$\alpha_d = \begin{cases} (|o| + |d|)v_{o\to d} & \text{if } v_{o\to d} \geq k \\ 0 & \text{else} \end{cases},$$

and the suppression error of node d is:

$$\sigma_d = \begin{cases} 0 & \text{if } v_{o\to d} \geq k \\ v_{o\to d} & \text{else} \end{cases},$$

so the best pruning error of node d, defined by:

$$B_d(\lambda) = \begin{cases} \min\left\{ \sum_{c\in\mathcal{C}} B_c(\lambda), \alpha_d + \lambda\sigma_d \right\} & \text{if } d \text{ has children and } v_{o\to d} \geq k \\ \alpha_d + \lambda\sigma_d & \text{else} \end{cases}$$

becomes:

$$B_d(\lambda) = \min\left\{ \sum_{c\in\mathcal{C}} B_c(\lambda), (|o| + |d|)v_{o\to d} \right\}$$

where $B_d(\lambda) = \sum_{c\in\mathcal{C}} B_c(\lambda)$ means that the optimal solution implies splitting the d (so $x^* = 1$), and $B_d(\lambda) = (|o| + |d|)v_{o\to d}$ means that the optimal solution implies keeping d aggregated (so $x^* = 0$). So we have to prove Eq. 1:

$$\sum_{c\in\mathcal{C}} B_c(\lambda) < (|o| + |d|)v_{o\to d}. \tag{1}$$

We separate $\mathcal{C}$ into the set of children with volume above k and the set of children below k:

$$\mathcal{C}^+ = \left\{ c \in \mathcal{C}, v_{o\to c} \geq k \right\}$$
$$\mathcal{C}^- = \left\{ c \in \mathcal{C}, v_{o\to c} < k \right\}$$

Then, by definition of $B_c(\lambda)$:

$$\sum_{c\in\mathcal{C}} B_c(\lambda) \leq \sum_{c\in\mathcal{C}^+} (|o| + |c|)v_{o\to c} + \lambda \sum_{c\in\mathcal{C}^-} v_{o\to c},$$

and $|d|$ and $v_{o\to d}$ can be expressed as sums over the children:

$$\begin{aligned}(|o| + |d|)v_{o\to d} &= \left(|o| + \sum_{c\in\mathcal{C}^+} |c| + \sum_{c\in\mathcal{C}^-} |c|\right)\left(\sum_{c\in\mathcal{C}^+} v_{o\to c} + \sum_{c\in\mathcal{C}^-} v_{o\to c}\right) \\ &= |o| \sum_{c\in\mathcal{C}^+} v_{o\to c} + \sum_{c\in\mathcal{C}^+} |c| \sum_{c\in\mathcal{C}^+} v_{o\to c} + \sum_{c\in\mathcal{C}^-} |c| \sum_{c\in\mathcal{C}^+} v_{o\to c} + (|o| + |d|) \sum_{c\in\mathcal{C}^-} v_{o\to c}\end{aligned}$$

where:

$$\sum_{c\in\mathcal{C}^+} |c| \sum_{c\in\mathcal{C}^+} v_{o\to c} \geq \sum_{c\in\mathcal{C}^+} |c| v_{o\to c} \qquad \text{(strict inequality if } \mathcal{C}^+ \text{ is not empty)}$$
$$\sum_{c\in\mathcal{C}^-} |c| \sum_{c\in\mathcal{C}^+} v_{o\to c} \geq 0$$
$$(|o| + |d|) \sum_{c\in\mathcal{C}^-} v_{o\to c} \geq \lambda \sum_{c\in\mathcal{C}^-} v_{o\to c} \qquad \text{(strict inequality if } \mathcal{C}^- \text{ is not empty)}$$

It follows that Eq. 1 is true:

$$(|o| + |d|)v_{o\to d} > \sum_{c\in\mathcal{C}^+} (|o| + |c|)v_{o\to c} + \lambda \sum_{c\in\mathcal{C}^-} v_{o\to c} \geq \sum_{c\in\mathcal{C}} B_c(\lambda)$$

We proved that for any d with children such that $v_{o\to d} \geq k$, the best pruning error of d is obtained by splitting d. In order to ensure that d is actually split in the optimal solution, we also have to prove that the precedence constraint is met. If we note a a node on the path from the $root(T)$ to d, then we know that $|a| > |d|$. So $|o| + |a| > |o| + |d| > \lambda$, so the optimal solution also implies splitting a. As it is true for all nodes from $root(T)$ to a, by induction d is split in the optimal solution.

□

F Feasibility of the solution found by dual tree

In this section, we prove that the aggregation x^* found by solving problem D respects the suppression constraint:

$$\sum_{x\in T}(x_{p(d)} - x_d)\sigma_d \leq C.$$

As illustrated in appendix D, the function $L : \lambda \mapsto \min_x \sum_{d\in T}(x_{p(d)} - x_d)(\alpha_d + \lambda\sigma_d) - \lambda C$ is piecewise linear, characterized by its breakpoints. Each piece of L is defined on bounds $[\lambda_l^0; \lambda_r^0]$ and represents at least one solution

$$x^0 = \underset{x\in[\lambda_l^0;\lambda_r^0]}{\operatorname{argmin}} \sum_{d\in T}(x_{p(d)} - x_d)(\alpha_d + \lambda\sigma_d) - \lambda C.$$

The value associated to x^0 depends linearly on λ, with derivative:

$$\forall\lambda \in [\lambda_l^0; \lambda_r^0], L'(\lambda) = \sum_{d\in T}(x^0_{p(d)} - x^0_d)\sigma_d - C.$$

Two consecutive pieces of L share a single point $(\tilde{\lambda}, L(\tilde{\lambda}))$, meaning the solution x^0 corresponding to the left piece and the solution x^1 corresponding to the right piece have the same value $L(\tilde{\lambda})$. As proven in appendix D, the maximum of F is reached. It is necessarily reached for a least one breakpoint $(\lambda^*, L(\lambda^*))$, which necessarily verifies: $L'_-(\lambda^*) \geq 0$ and $L'_+(\lambda^*) \leq 0$, where L'_- and L'_+ denote the left-hand and the right-hand derivatives, respectively. Then the derivative of the right piece is $L'_+(\lambda^*)$, so the solution x^* corresponding to the right-hand piece of L verifies:

$$\sum_{d\in T}(x^*_{p(d)} - x^*_d)\sigma_d - C \leq 0.$$

We proved that there always exist an optimal solution x^* to D that respect the suppression constraint.

G Relevance of the reconstruction loss as an indicator of the granularity of a spatial generalisation of an OD-matrix

In this section, we discuss some remarks about the reconstruction loss and its relevance as an indicator for the information loss of an anonymisation algorithm: OD-matrices are by nature sparse and they can also have sparse emission and reception maps, for example if the matrix is set over a fine grid. The k-anonymised version of such an OD-matrix features generalised origins with a small number of emission hot spots separated by a high number of empty tiles, and similarly for destinations. For the sake of simplicity, we consider a generalised flow $o \to d$ containing two hot spots each contributing $v_{o\to d}/2$, and a number z_o and z_d of empty tiles in o and d, respectively. The reconstructed flows uniformly distribute $v_{o\to d}$ into all $z_o \times z_d$ possible flows. This means a difference of $\frac{v_{o\to d}}{z_o z_d}$ for the empty flows, and $\frac{v_{o\to d}}{2} - \frac{v_{o\to d}}{2z_o z_d}$ for the two hotspots. We obtain the absolute difference:

$$\begin{aligned} E_{o\to d} &= (z_o z_d - 2)\frac{v_{o\to d}}{z_o z_d} + v_{o\to d} - \frac{v_{o\to d}}{z_o z_d} \\ &= 2v_{o\to d} - 3\frac{v_{o\to d}}{z_o z_d} \end{aligned}$$

which quickly converge to $2v_{o\to d}$ when $z_o z_d$ grows. As long as $z_o \approx z_d \geq 15$, the noise then mostly represents the volume of the flow, regardless of its other properties, which is not a desirable feature.

H Temporal analysis is still possible even though separate OD-matrices are generalised differently for each timestep

In this section, we illustrate how separating the flows into distinct OD-matrices corresponding to time steps does not stop us from doing temporal analysis. Indeed, the reconstruction process described in Section 4.3

allows us to retrieve an estimation of the flows between any arbitrary couple of zones, which we can choose to be consistent across timesteps. By reconstructing anonymised OD flows on the initial zoning, we can compare them with the initial flows. An exemple of this is given in Fig. 14, which represents the mean volume of some example flows throughout the week, as obtained from the original data and from the data anonymised with ATG-dual.

I Solving the best pruning problem

In this section, we detail Algo. 1 to solve the Best Pruning Problem P: For each node n, α_n can be evaluated in time $\mathcal{O}(1)$. If we note M the number of leaves of T, which correspond in our case to the size of the study area, then a tree has $\mathcal{O}(M)$ nodes and solving problem P can be done in time $\mathcal{O}(M)$. Problem P can also be formulated as a maximisation of the benefit compared to splitting nothing, in which case the objective function sums over all selected nodes instead of only the leaves. This other form of the problem is known as the maximal closure problem, and can also be solved using an efficient max-flow/min-cut algorithm [36].

Algorithm 1: get_pruning

Input : node n, error_fun
Output: best pruning error of node n, and the corresponding pruning

```
error_agg = error_fun(n)
if tree has children then
    error_split ← 0
    pruning_split ← ∅
    for child ∈ tree.children do
        bpe_c, pruning_c ← get_pruning(child, error_fun)
        error_split ← error_split + bpe_c
        pruning_split ← pruning_split ∪ pruning_c
    end for
    if error_split < error_agg then
        return error_split, pruning_split
    else
        return error_agg, {n}
    end if
else
    return error_agg, {n}
end if
```

J Tree aggregation algorithm

In this section, we detail Algo. 2, to decouple the generalisation problem. It relies on the solving of Problem P for the generalisation of origins, which is detailed in Algo. 1.

Algorithm 2: Decoupling of the generalisation problem

Input : $v_{o\to d}$, hierarchy_tree, v_target, k, S
Output: aggregated_od_matrix

```
od_matrix_agg ← ∅
error_fun ← (f : o ↦ (∑_d v_{o→d} − v_target)^2)
_, generalised_origins ← get_pruning(hierarchy_tree, error_fun)
agg_flows ← generalise_destinations(generalised_origins, hierarchy_tree, k, S)
for (o, d, vol) ∈ agg_flows do
    od_matrix_agg.append((o, d, vol))
end for
return od_matrix_agg
```

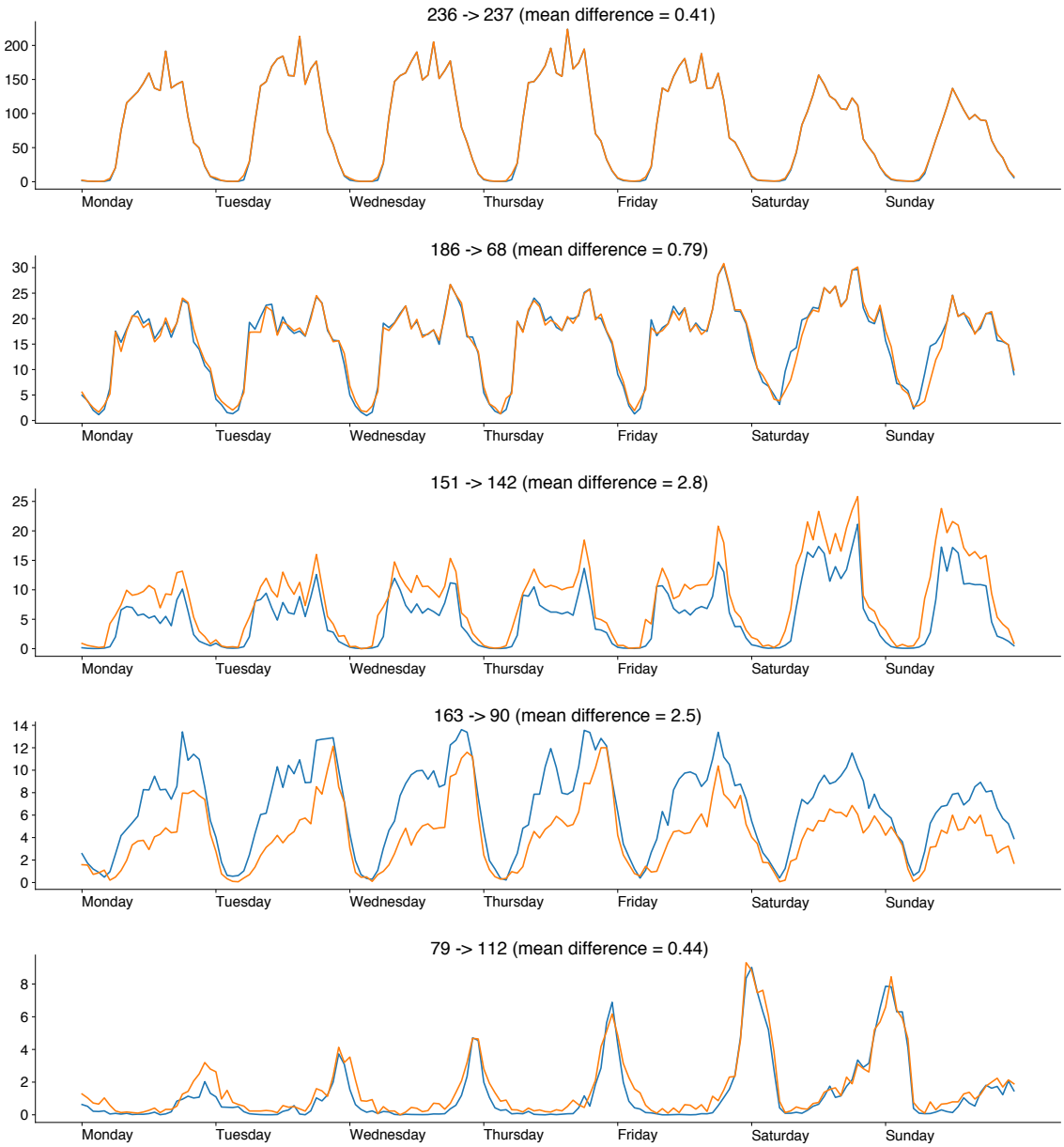


Figure 14: Weekly profiles of $o \to d$ flows of various importance in dataset nyc. Blue curve: profile obtained from the OD-matrices anonymised by ATG-dual then reconstructed. Orange curve: profile obtained from the initial OD-matrices.

For a given origin map $\mathcal{M}_o$, we look for one destination map for each origin such that the total error of all the destinations map is minimal and the total suppressed volume is below the threshold C. The corresponding problem is problem H applied on a tree composed of $|\mathcal{M}_o|$ times the generalisation hierarchy, whose roots are all the children of one dummy node that would be always selected in the solution. Each one of the subtrees represents the destinations for one particular origin, so each node corresponds to a particular flow $v_{o\to d}$. In the following Algo. 3, constructing this tree is handled in the `build_destinations_tree` function, that assigns the aggregations errors $\alpha_{o\to d}$ and suppression costs $\sigma_{o\to d}$. Actually solving for problem H is computationally very expensive, so Algo. 3 rather solve for the dual problem D, which can be fast to solve using the Specific Breakpoints Algorithm [34]. Algorithm 3 is thus an adaptation of the specific breakpoints algorithm for the leaf-error minimization. The Specific Breakpoints Algorithm requires to evaluate the value and the derivative of the best pruning error, which is done in a similar way than the solving of the pruning in Algo. 1. As it does not return the same output, we include it in a separate algorithm 4.

Algorithm 3: generalise_destinations

Input : generalised_origins, hierarchy_tree, k, C
Output: aggregated_od_matrix

dest_tree ← build_destinations_tree(generalised_origins, hierarchy_tree)
$\lambda_l \leftarrow 0$
bpe_l, $\text{bpe'}_l \leftarrow$ dest_tree.root.evaluate(λ_l)
$e_l \leftarrow \text{bpe}_l - \lambda_l C$ `// value of objective function`
$s_l \leftarrow \text{bpe'}_l - C$ `// derivative of objective function`
$\lambda_r \leftarrow$ M
bpe_r, $\text{bpe'}_r \leftarrow$ evaluate(dest_tree.root, λ_r)
$e_r \leftarrow \text{bpe}_r - \lambda_r C$
$s_r \leftarrow \text{bpe'}_r - C$
while True **do**
  $\lambda_m \leftarrow (s_l\lambda_l - s_r\lambda_r + e_r - e_l)/(s_l - s_r)$ `// find the intersection of the tangents`
  bpe_m, $\text{bpe'}_m \leftarrow$ evaluate(dest_tree.root, λ_m)
  $e_m \leftarrow \text{bpe}_m - \lambda_m S$
  $s_m \leftarrow \text{bpe'}_m - C$
  if $\lambda_m = \lambda_l$ **or** $\lambda_m = \lambda_r$ **then**
    error_fun $\leftarrow (f : n \mapsto \alpha_n + \lambda_m \sigma_n)$
    return get_pruning(dest_tree, error_fun)
  else
    if $s_m > 0$ **then**
      $(\lambda_l, e_l, s_l) \leftarrow (\lambda_m, e_m, s_m)$
    else
      $(\lambda_r, e_r, s_r) \leftarrow (\lambda_m, e_m, s_m)$
    end if
  end if
end while

Algorithm 4: evaluate

Input : node n, λ
Output: best pruning error of node n, and its derivative w.r.t λ

```
error_agg = α_n + λσ_n
deriv_agg = σ_n
if tree has children then
    error_split ← 0
    deriv_split ← 0
    for child ∈ tree.children do
        bpe_c, bpe'_c ← evaluate(child, λ)
        error_split ← best_error_split + bpe_c
        deriv_split ← best_deriv_split + bpe'_c
    end for
    if error_split < error_agg then
        return error_split, deriv_split
    else
        return error_agg, deriv_agg
    end if
else
    return error_agg, deriv_agg
end if
```