Int J Data Sci Anal (2017) 4:285–299
DOI 10.1007/s41060-017-0065-y

REGULAR PAPER

# Scalable and flexible clustering solutions for mobile phone-based population indicators

**Alessandro Lulli**[1] · **Lorenzo Gabrielli**[2] · **Patrizio Dazzi**[2] · **Matteo Dell'Amico**[3] · **Pietro Michiardi**[4] · **Mirco Nanni**[2] · **Laura Ricci**[5]

Received: 11 February 2017 / Accepted: 16 July 2017 / Published online: 28 July 2017


**Abstract** Mobile phones have an unprecedented rate of penetration across the world. Such devices produce a large amount of data that have been used on different domains. In this work, we make use of mobile calls to monitor the presence of individuals region by region. Traditionally, this activity has been conducted by means of censuses and surveys. Nowadays, technologies open new possibilities to analyse the individual calling behaviour to determine the amount of residents, commuters and visitors moving in an area. To this end, in this paper we provide a clustering technique completely unsupervised able to cluster data by exploring an arbitrary similarity metric. We make use of such technique, and we define metric to analyse mobile calls and individual profiles. The approach provides better population estimation with respect to state of the art when results are compared with real census data and greatly improves the execution time of a previous work of some of the authors of this paper. The scalability and flexibility of the proposed framework enables novel scenarios for the characterization of people by means of data derived from mobile users, ranging from the nearly real-time estimation of presences to the definition of complex, uncommon user archetypes.

**Keywords** Clustering · Apache spark · Population indicators · Mobile calls · Distributed algorithm

✉ Alessandro Lulli
lulli@di.unipi.it; alessandro.lulli@dibris.unige.it

Lorenzo Gabrielli
lorenzo.gabrielli@isti.cnr.it

Patrizio Dazzi
patrizio.dazzi@isti.cnr.it

Matteo Dell'Amico
matteo_dellamico@symantec.com

Pietro Michiardi
pietro.michiardi@eurecom.fr

Mirco Nanni
mirco.nanni@isti.cnr.it

Laura Ricci
laura.ricci@unipi.it

1 University of Genoa, Genoa, Italy

2 ISTI, CNR, Pisa, Italy

3 Symantec Research Labs, Sophia Antipolis, France

4 EURECOM, Campus SophiaTech, Biot, France

5 University of Pisa, Pisa, Italy

## 1 Introduction

Between 2013 and 2017, mobile phone penetration has increased from 61.1 to 69.4% of the global population, according to several reports [1]. An assumption, which is often considered valid, is that the position of the mobile devices is the position of their users [2]. As a consequence, it is possible to use such information in many different domains and in ways for which they were not meant. For instance, such data have been successfully used for traffic monitoring [3] or tourist movements. Thanks to this information, available at the level of the telecom infrastructure (e.g. calls, SMS), we define a conceptual and technological framework that characterizes the mobility of the users without requiring any kind of interaction with the software and the specific hardware of the mobile device. Our framework provides instruments able to estimate individuals living in a certain region (residents), the ones that are used to work or study into that region (commuters) and those who visit a certain area (visitors). In these cases, it is important to have tools to group individuals sharing a common behaviour properly. In a previous work, by some of the authors of this paper [4], the

algorithm adopted for data clustering was $K$-means. This is easy to use and adopted in many circumstances, but it is not free from weakness. First of all, it requires identifying the number of clusters (the $K$ parameter), whose choice is not immediate. Additionally, $K$-means is not able to discriminate noise data that characterize most of the real-world datasets. One of the consequences is that noise data may significantly affect the quality of the result and the compactness of clusters. Beyond the "functional" limitation of $K$-means, from the non-functional viewpoint, it is very challenging to design scalable distributed clustering algorithms. In fact, albeit $K$-means is in principle easy to parallelize, it suffers of a large runtime when $K$ is large and requires a large number of similarity computations.

To overcome the aforementioned limitations and address the issues underpinning previous works, we already proposed a base version of Muchness [5], a framework that is able to estimate the number of residents, commuters and visitors in a given region by exploiting mobile phone data. In this paper, we extend our contributions as the following:

- We enhance the clustering algorithm and provide a completely unsupervised clustering algorithm which does not require input parameters from the users and is able to accommodate an arbitrary similarity metric;
- we make a study on indicators about individuals movements such as the flows between home and work locations and from home to visit places;
- based on the outcome of the estimations of the number of residents, commuters and visitors, we build individual profiles and we define a personalized metric able to capture similarities between individuals. This opens new possibilities for the study of individuals behaviour.

The remaining of this paper is organized as follows: Sect. 2 introduces the related works, Sect. 3 provides insight about the typology of data we analyse, Sect. 4 describes the seminal idea that we extend in this work, Sect. 5 describes the details of our framework, while Sects. 6 and 7 present the results we have obtained. Section 8 details the impact of the research and the future works.

## 2 Related work

Several studies use data driven by mobile phone calls due to their large market penetration in the recent years. In fact, many works study all the possible social and economic indicators that can be extracted from such data. In this section, we discuss works related to ours about mobile calls data analysis (Sect. 2.1) and about scalable clustering algorithms (Sect. 2.2).

### 2.1 Mobile calls data analysis

Mobile phone traces have been utilized to monitor the traffic in cities and analyse tourist movements. In particular, two popular works focus on this issue for the cities of Rome [3] and Graz [7]. From these works, many others, for instance Ahas et al. [8], analyse that it is possible to detect the places visited by the individuals by analysing the calls they perform. In addition, a plethora of works, for instance, the winner of the Nokia Mobile Data Challenge [9], build predictors able to determine the next position of an individual given the current context.

qst ha usato conoscenze sulle CHIAMATE (non per noi .

De Jonge et al. [10] study different approaches making use of call records spanning 2 weeks, in Netherlands. They give insights into the indicators obtainable analysing the phone calls. For instance, the number of phone calls can be used as an indicator to estimate the economic activity of a certain region. They make use of the $K$-means clustering algorithm to determine day pattern clusters of the call activity. However, they suggest that a deeper study of the calling behaviour should be performed on a larger dataset covering multiple weeks to correctly estimate population density.

One of the first works using mobile data to estimate the population has been presented by Terada et al. [11]. In this work, they monitor the presence of mobile terminals present in each base station area in different time intervals. Such data are refined with census information and at the end the per-cell populations are aggregated in grid sections or municipalities. This result may be affected by errors. Checking only the presence in a cell cannot detect if an individual is a resident, who should be counted as living in the area, or just a visitor. Due to this, subsequent works try to exploit mobile data in a different manner.

Deville et al. [6] improve the ideas of De Jonge and exploit mobile phone data for estimating population density. They propose a framework called MP. According to such methodology, population density is estimated as a function of the night-time phone calls occurring in a given area. However, a simple rule-based approach to identify user presence may hinder to derive some more useful information about the calling behaviour of the users. For instance, it would be cumbersome to define rules able to characterize individuals that are Commuters or Visitors. To overcome the aforementioned limitations, in a seminal work Furletti et al. [12] defined how to build individual profiles based on mobile phone calls. Such profiles characterize the calling behaviour of a user, in different time slots. By analysing these profiles, it is possible to identify three categories of users: Residents, Commuters or Visitors. Sociometer [4] focuses on this characterization to aggregate users having a similar calling behaviour with the $K$-means clustering algorithm. The centroid of each cluster is compared with pre-defined archetypes representing the categories of interest, and then, each cluster is classified by

anche qua si

**Table 1** Overview of frameworks to estimate population

| Name | Method | Clustering configurable | Residents | Commuters | Visitors |
|---|---|---|---|---|---|
| MP [6] | Rules on each data | N/A | Yes | No | No |
| Sociometer [4] | Clustering $K$-means | Number of clusters | Yes | Yes | Yes |
| Muchness [5] | Clustering $k$-NN based | 3 main parameters | Yes | Yes | Yes |
| Muchness+ | Clustering $k$-NN based | Completely unsupervised | Yes | Yes | Yes |

means of the associated archetype. Hereafter, we use the term *exemplar* to refer to the cluster's centroid.

The seminal idea of this work has been presented by Lulli et al. [5] where a novel scalable algorithm has been used to estimate the population. We give additional details of this work in Sect. 4. This work advances the achievements of Sociometer and Muchness in the following directions: (i) it provides a scalable distributed approach which can process a sensibly larger collection of data requiring no input from the user; (ii) it defines a personalized similarity metric that leads to better clustering results and is able to cluster different kind of data relative to different mobile calls aggregation strategies; (iii) it automatically removes outliers to improve the overall quality and to provide a better estimation of the population; (iv) it does not require to provide in advance the number of clusters as in $K$-means; (v) it studies the indicators that can be extracted from mobile calls such as how individuals move to reach their working location. Table 1 shows the main differences between our work and the related works described in this section.

### 2.2 Scalable clustering algorithms

Mobile data are usually of large size. In this work, we analyse the phone calls performed daily in the Italian region of Tuscany. Due to this, when we need to cluster such amount of data a scalable clustering algorithm is of paramount importance. In this section, we cover several scalable clustering algorithms related to ours.

dà note sul metodo statistico

One of the most popular clustering algorithm is $K$-means which aggregates data around $K$ centroids. It has three main limitations: the $K$ parameter has to be user-provided, it is limited to euclidean spaces, it has a bias on the initial selection of centroids. Moreover, despite parallel and distributed implementations of $K$-means exist, they suffer of longer running time when $K$ is large due to the large number of comparisons.

Another interesting class of clustering algorithms falls in the DBSCAN family, defined by Ester et al. [13]. The underpinning idea is to cluster items that have at least MinPts neighbours at maximum distance $\varepsilon$. The main advantages against $K$-means are the following: (i) it is not required to know the number of clusters in advantage; (ii) the ability to cluster items with complex shapes instead of aggregating items that are simply close (according to the euclidean distance) to a centroid. MR-DBSCAN [14] has been the first proposal targeting a distributed implementation of DBSCAN, realized as a 4-stage MapReduce algorithm. This approach focuses on the definition of an efficient data partitioning in a $d$-dimensional Euclidean space, where each partition is assigned to a worker node. This solution is limited, similarly to $K$-means, to work on euclidean spaces.

Recently, a distributed clustering algorithm based on nearest neighbour graphs [15] able to deal with arbitrary similarity metrics has been proposed, albeit in the original paper experiments have been performed only for the JaroWinkler metric. This is at the basis of the approach of Muchness. In such work, a metric conceived for the data under exam making use of a linear combination of Euclidean distance and Jaccard similarity has been used (additional details can be found in Sect. 4). However, such approach has several drawbacks. Albeit it is not necessary to define the number of clusters in advantage, it requires three parameters to tune the quality and the execution time of the algorithm. Due to this, in this work we extend such algorithm in order to remove the requirement of setting input parameters from the user. This will facilitate its usage and target a good trade-off between clustering quality and execution time.

## 3 Data description

Telco operators collect customer data for billing purposes. Refer to Fig. 1 to have an overview of how data are created, collected and aggregated. From one Telco operator in Italy, we received anonymized data of calls performed in Tuscany (Italy) recorded during the period between February and March 2014. Each call record is a tuple having the anonymous identifier of the user, the call timestamps and the cell id. Nevertheless, our approach for the analysis of the individuals can be applied to every kind of data which provides the above information. We manage approximately 60 mln calls (column Telco data in Fig. 1). Each cell id can be assigned to a municipality. A municipality is an administrative tessellation of the territory. Our data span between municipalities having a density of population in the range 6–261 individuals per square kilometre. Figure 2 describes the amount of calls

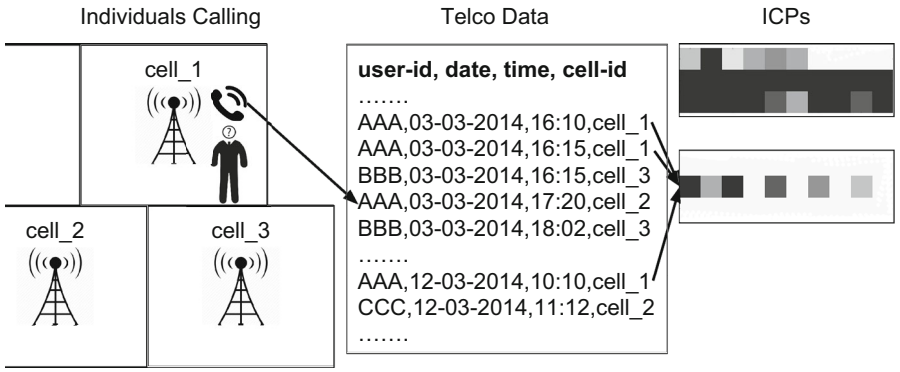


**Fig. 1** Individuals perform calls under a given cell (relative to a municipality). Each call is collected from the Telco operators. We receive such data, and we aggregate the calls of each individual creating a user calling profile (ICP)

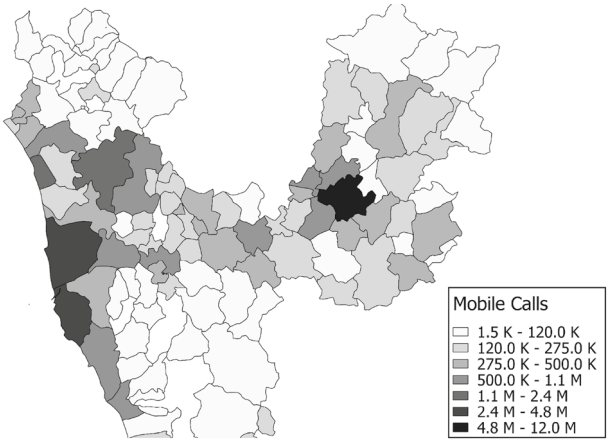


**Fig. 2** Amount of calls in the dataset. *Darker colours* represent a higher amount of calls collected in the dataset

collected in the dataset for each municipality. As expected, the cities are characterized by the largest amount of calls with respect to small municipalities. For each individual, we compute an Individual Call Profile (ICP), following the approach defined in a paper from Furletti et al. [12]. An ICP represents the calling behaviour of an individual in a municipality (column ICPs in Fig. 1). Due to this, each individual may have multiple ICPs if the user performed calls in different municipalities in the time period. These are used to identify if an individual is a resident, commuter or visitor in the municipality. Each ICP is a 30-dimensional array in which each position represents a specific time slot of the day (morning, afternoon, evening) discriminating between weekdays and weekends for a total of the 5 weeks under analysis. A value greater than 0 indicates that the represented user performed at least one call in a specific time slot. At the end of the aggregation process, we obtain around 2.6 mln ICPs representing calls generated by about 800k individuals from 115 different municipalities.

The clustering algorithm uses the ICPs as input to provide clusters of individuals and tag such clusters as Resident, Commuter or Visitor. Such information is eventually processed, to estimate the number of residents, commuters and visitors for each municipality. It is possible to extract many useful information from such data. For instance, check Fig. 3a, b where are described the amount of workers travelling in-coming and out-coming the municipality of Pisa from the other municipalities. In particular, we identified all the individuals being commuters in Pisa. Than, for each individual $i$ of this set we searched in our result in which municipality, if different from Pisa, $i$ has been recognized as resident. Figure 3a represents the amount of residents in the municipalities that are commuters in Pisa. It is nice to observe that the majority of the work travellers are from the surrounding municipalities, principally from Livorno. Finally, we reversed such approach searching for each individual $i$ resident in Pisa where $i$ has been recognized a commuter. We call this the out-coming result presented in Fig. 3b. Despite the majority of the Pisa's resident work in the surrounding of Pisa, some of them travel everyday to Florence, the biggest city in Tuscany.

## 4 Preliminaries: Muchness

In this section, we provide some details about the Muchness technique targeting population estimation. In particular, we highlight some shortcomings that we improve on Muchness+.

Muchness estimates the population in 3 phases:

- *individual characterization*: we start from raw data about mobile calls where for each call we have the timestamps, an individual identifier and the position of the caller. These data are aggregated resulting for each individual in a individual calling profile (ICP) for a given municipality. An ICP provides information about the time of the day when the individual perform calls.
- *clustering*: we cluster the ICPs with a specialized similarity metric;
- *classification*: each cluster is classified as composed of Residents, Commuters or Visitors.

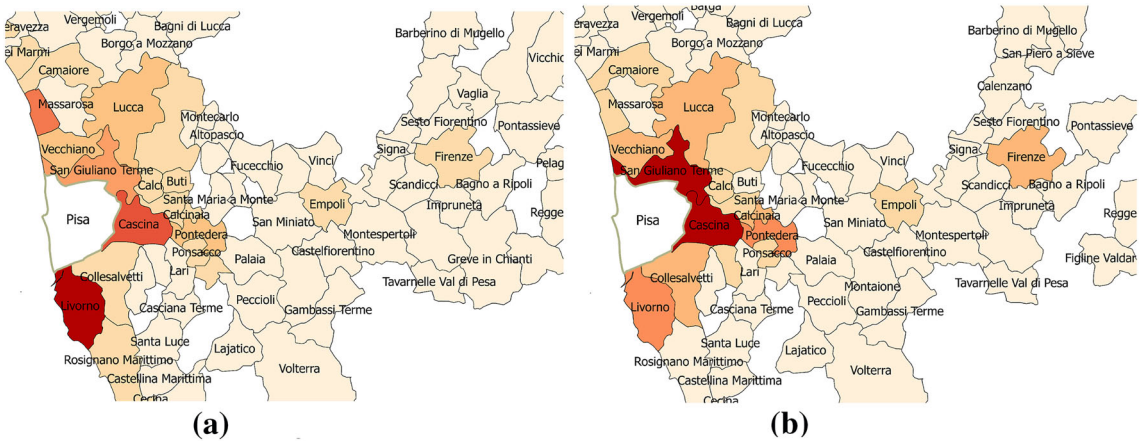


**Fig. 3** In-coming and out-coming workers for the Pisa municipality. **a** In-coming workers. **b** Out-coming workers

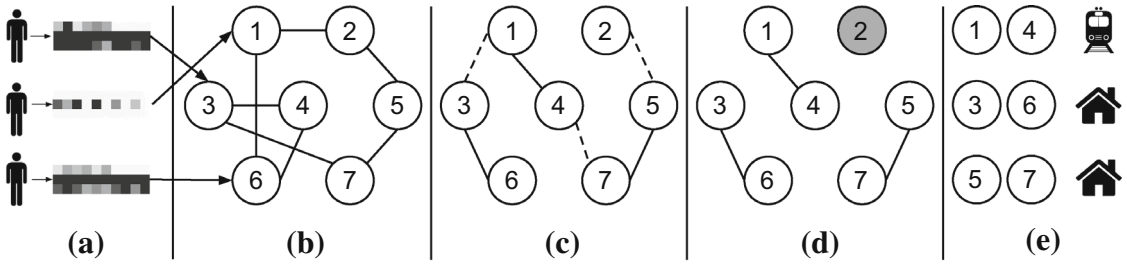


**Fig. 4** Muchness analytical process. **a** For each individual we assign an ICP. **b** Each ICP becomes a node in a graph. **c** We search for similar nodes and at the end we prune low similarity edges (*dashed*). **d** We search for connected components and we identify outliers (node 2). **e** For each cluster, we define an exemplar (icons) classified as Resident, Commuter or Visitor

Figure 4 gives an overview of the whole analytical process of Muchness. For each mobile user, we build an ICP (see column A). Then, we start our clustering algorithm that has the peculiarity of accepting an arbitrary similarity metric. It is a two-phase algorithm. First, we build iteratively a nearest neighbour graph ($k$-NN), according to the given similarity metric. Second, we search for connected components in the $k$-NN graph. We make use of the ICPs to generate the graph. At the bootstrap, we randomly link each node to few other nodes (see column B). Then, the algorithm iterates, starting from the initial graph, adjusting the neighbourhood of each node with the most similar nodes. In the following stage, the edges connecting nodes whose similarity is below a given threshold parameter are pruned (see column C). The resulting clusters are the connected components [16,17] derived from the pruned graph (column D). It is worth to notice how in this phase the nodes without neighbours are identified as outliers (situation represented in Fig. 4 by node #2). Finally, for each cluster an exemplar is generated (column E), used by the automatic classifier to label the clusters as Resident, Commuter or Visitor. The exemplar is an ICP where the value of each element of the array is the average of the relative values of all the ICPs of the cluster. Then, the classification is performed similarly to [4].

This solution requires specifying three parameters: $k$, $numIter$ and $\varepsilon$.

- $k$ represents the number of neighbours for each node in the graph, and it affects both the quality and the execution time of the clustering. In general it is acceptable to set a value $\in [5, 10]$ to have a good trade-off between quality and time as suggested in Lulli et al. [15];
- $numIter$ fixes the number of iterations performed by the algorithm. Larger value provides a better $k$-NN graph at the cost of a longer running time;
- $\varepsilon$ is a threshold parameter that drives the edge pruning process to avoid that very different nodes would fall in the same cluster.

## 5 From Muchness to Muchness+: a framework for census

In this work, we target to improve the analytical process of Muchness described in Sect. 4. In particular, we think it is of paramount importance to provide a completely unsupervised approach (i.e. where no configuration is required from the user) to avoid trial-and-error approach when changing the

**Algorithm 1:** Muchness+ clustering algorithm.

```
1  k-NN = RandomInitialization()
2  meanSimilarity = mean(k-NN)
3  tmp = -1
4  while i < numIter ∧ |tmp − meanSimilarity| > 0.01 do
5  |  H = {ReverseMap(n)∀n ∈k−NN}
6  |  T = {CheckNeighborhood(n)∀n ∈H}
7  |  k-NN = {ReduceNeighbor(n, l)∀(n, l) ∈T}
8  |  tmp = meanSimilarity
9  |  meanSimilarity = mean(k-NN)
10 |  i = i + 1
11 end
12 S={n ∈k-NN |s_u < 1}
13 ε=mean(S)
```

**Algorithm 2:** Muchness procedures.

```
1  procedure ReverseMap(Node n)
2  |  forall the u ∈Neighborhood(n) do
3  |  |  emit(n, u)
4  |  |  emit(u, n)
5  |  end
6  procedure CheckNeighborhood(Node n)
7  |  forall the u ∈ Neighborhood(n).Limit (ρk)∪{n} do
8  |  |  l = ∅
9  |  |  forall the v ∈ Neighborhood(n) ∪ {n} \ {u} do
10 |  |  |  l = l ∪ ((v, distance(u, v)))
11 |  |  end
12 |  |  emit(u, l)
13 |  end
14 procedure ReduceNeighbor(Node n, List⟨(Node, Distance)⟩
   l)
15 |  localMeanSimilarity=mean(l)
16 |  if localMeanSimilarity ∼ 1 then
17 |  |  orderedList = orderDESC(l).Limit(j/2)
18 |  |  emit(n, orderedList)
19 |  else
20 |  |  orderedList = orderDESC(l).Limit(j)
21 |  |  emit(n, orderedList)
22 |  end
```

data to achieve a good result. To this end, we introduce some techniques to avoid the input of all the parameters required by the previous Muchness approach. We call this new version of the algorithm Muchness+. In addition, we define how we can find similarity metrics that adapt to mobility data without choosing them according to the algorithm implementation.

### 5.1 Improving the clustering algorithm

In this section, we provide insights into how we improve the algorithm with respect to the one used in Muchness. In particular, we concentrate on the following aspects:

- avoid bad performance in degenerate cases, bounding the number of messages to $O(\rho k)$ (Sect. 5.1.1);
- provide a completely unsupervised algorithm, which does not require parameters from the users. In particular, as highlighted in Sect. 4, the previous algorithm requires three parameters that are not required any more in Muchness+: $k$ the size of the neighbourhood on each node (Sect. 5.1.2), $numIter$ the number of iterations (Sect. 5.1.3), $\varepsilon$ to prune low similarity edges (Sect. 5.1.4).

Refer to Algorithm 1 for an high level description of the clustering algorithm. Also, refer to Algorithm 2 for the details of the different phases of the algorithm and the optimizations introduced.

#### *5.1.1 Introducing a sampling mechanism*

Before giving the details of the sampling technique introduced in Muchness+, it is interesting to analyse the number of messages required to build the $k$-NN graph. The directed $k$-NN graph in each iteration is initially reversed to construct an undirected graph (see Function ReverseMap Alg. 2 Line 1). Due to this, it may happen that a node $u$, if by construction initially has $k$ directed neighbours like all the other nodes, after the reverse operation in the worst case may have $n-1$ neighbours, where $n$ is the number of node in the graph. If this is the case, $O(kn)$ messages are required, in the following phases of the algorithm, in node $u$ to communicate the 2-hop neighbourhoods to all its $n-1$ neighbours (see the Forall at Alg. 2 Line 7).

To avoid such degenerate scenarios, we introduce a sampling parameter $\rho$. After the reverse operation, each node keeps uniformly at random a maximum of $\rho k$ neighbours (see Line 7). This operation permits to bound the number of messages on each node, instead to $O(kn)$, in $O(\rho k)$.

#### *5.1.2 Automatic neighbourhood selection*

The $k$ parameter in Muchness represents the number of neighbours for each node in the graph. It affects both the quality and the execution time of the clustering. In general, it is acceptable to set a value $\in [5, 10]$ to have a good trade-off between quality and time as suggested in Lulli et al. [15]. However, this may require an analysis and an input from the user. Due to this, we provide an heuristic to avoid such input and we let each node autonomously resize its neighbourhood depending on its state.

On each node $u$, at each iteration $t$, we define the average similarity between $u$ and all its current $k$ neighbours equal to $s_u^t$. Our algorithm is iterative and improve the neighbours of each node in each iteration. Due to this, given $t' > t$ we have $s_u^{t'} \geq s_u^t$. If a node $u$ discovers a neighbour $v$ in iteration $t$, the neighbour $v$ can be substituted in $t' > t$ only by a node $z$ whose similarity with $u$ is greater. Due to this, if $u$ has already collected enough good neighbours, where good means that the similarity metric is to close to 1, we can limit its view,

whereas if $u$ has no good neighbours, we need to keep $k$ large to improve the possibility to discover better neighbours.

From the above hint, we defined a methodology able to automatically set on each node a correct value for the parameter $k$. We set $k = j$ when $s_u^t$ is low (see Algorithm 2 Line 20), and we set $k = j/2$ when $s_u^t$ is high (see Line 17). We let each node perform computation at two speeds, respectively, when it searches neighbours or when it has already found good neighbours. In this case, $j$ can be set to a value large enough to permit to discover many nodes. From previous results [15], [5] is safe to set $j = 10$. This permits to perform more computation on nodes that need to improve the neighbours and just preserve the connectivity on the other nodes.

#### *5.1.3 Early termination*

The $numIter$ parameter defines the number of iterations that the algorithm performs. Previous results suggest that it is not required to perform a large number of iterations to obtain a good result. In addition, often a large part of the running time is spent for marginal improvements. Knowing the improvement of the solution over time enables the algorithm to decide on an early termination that may save longer running time. In addition, it is cumbersome to have a fixed amount of iterations to be performed without knowing the state of the algorithm.

We remark that our algorithm is improving the approximation in each iteration, and such approximation can be monitored to decide for an early termination. As before, we define on each node $u$ in each iteration $t$ the average similarity between $u$ and all its $k$ neighbours equal to $s_u^t$. Also, we define $S^t$ equal to the average of all the $s_u^t$ for each node $u \in G$. In Algorithm 1, we make use of the mean function to compute $S^t$ on each iteration (see Line 9). When the improvement of $S^t$ in two subsequent iterations is less that 0.01 we stop early the computation of the algorithm (see Line 4). This means that the majority of the nodes do not improve the neighbours and we can safely stop the computation.

#### *5.1.4 Automatic ε pruning*

In Muchness, one of the most important parameters is $\varepsilon$. It is a threshold parameter that drives the edge pruning process to avoid that very different nodes would fall in the same cluster. It is affecting the second part of the algorithm (i.e. the same $k$-NN graph can be used with different $\varepsilon$ values to cut a different number of edges). However, due to its nature, it is affecting the result considerably and it requires a trial-and-error approach to be refined.

In Muchness+, we define an heuristic capable of providing a good approximation to the expected value to be assigned to $\varepsilon$. As before, we define on each node $u$, in each iteration $t$, the average similarity between $u$ and all its $k$ neighbours equals to $s_u^t$. At the end of the $k$-NN creation phase, we collect the $s^t$ values of the last iteration and we remove all the $s^t = 1$. We call $\mathcal{S}$ (see Alg. 1 Line 12) the set of $s^t \neq 1$. We remove each node $u$ having all the neighbours identical to $u$ ($s^t = 1$) because such nodes must not affect the result to avoid biases. We set $\varepsilon = \overline{\mathcal{S}}$ the average of the values $\in \mathcal{S}$.

### 5.2 Adapt the metric to the data instead of the algorithm

One of the major characteristics of the clustering algorithm described before is its ability to handle arbitrary similarity metrics. This characteristic permits to adapt the similarity metric, used for the clustering algorithm, to the data instead of adapting the algorithm. Thanks to this, we define and use similarity metrics that are able to extract the most of the information from the data. In the following of this section, we describe the similarity metrics used on such data (Sect. 5.2.1), how we can analyse each individual (Sect. 5.2.2) and the metrics used for that (Sect. 5.2.3).

#### *5.2.1 Similarity metrics for ICPs*

In this section, we discuss the metrics that can be used for our data. As introduced before, each ICP is a 30-dimensional array representing the calling behaviour of an individual. We define the *shape* of an ICP equal to the positions of its array where the values are greater than 0. The shape enables the evaluation of the presence of an individual in the territory without considering the amount of calls performed. The Euclidean similarity (EUC) is unable to grasp similarities between ICPs having similar shapes. Due to this, our main idea is to introduce a metrics able to capture the similarities between individual sharing a common shape.

A metric able to capture the shape of the array is the Jaccard similarity (JAC). In order to use JAC, we modify each array in a boolean array where we set the value 1 in position $i$ if in position $i$ the data have a value greater than 0. However, the JAC takes into account exclusively the shape of the profiles, but it loses all the information about the weights in the array. Therefore, we combine the two similarities, the EUC and the JAC. We define the EUC+JAC similarity as follows:

$$\text{EUC+JAC}(a, b) = \alpha\text{EUC}(a, b) + (1 - \alpha)\text{JAC}(a, b) \quad (1)$$

Our goal is to identify the shape of the ICPs, due to this it is advisable to put more weight on the JAC. After a careful analysis, we identified in $\alpha = 0.4$ an acceptable configuration.

We provide an example supporting our idea in Table 2. Table 2 shows examples of the values of the presented similarity metrics for two residents and two commuters having similar shapes. Table 2 represents in the first two columns the ICPs selected and in the last three columns the similarity

**Table 2** Similar ICPs extracted by expertises. A comparison of similarity values using: EUC, JAC and EUC+JAC

| | | | EUC | JAC | EUC+JAC |
|---|---|---|---|---|---|
| Residents |  |  | 0.5 | 1 | 0.8 |
| Commuters |  |  | 0.78 | 1 | 0.91 |

values using different metrics. The ICPs have a very similar behaviour resulting in similar shapes. For instance, take in consideration the two residents in the first row of Table 2. Although some positions have different values, note the colour darkness representing the value on a single position of the array, they have an equal shape representing the same calling behaviour. With the EUC, we cannot assess that the two ICPs are similar (only 0.5 similarity); however, the JAC (giving value 1) suggests that the two ICPs have identical shapes. With our EUC+JAC, we can take the benefits of both the metrics and we obtain an high similarity of 0.8. Similar considerations can be applied also to the commuters example.

#### *5.2.2 Beyond ICPs: individual profiles for individuals analysis*

Once we have clustered and classified ICPs as Resident, Commuter or Visitor, it is possible to estimate the population in the region and in each municipality. Another interesting analysis is understanding the different typologies of individuals. Since one ICP is relative to an individual in a municipality, an individual may have multiple ICPs, one for each municipality where he travelled in the period under analysis. To this end, we think is of paramount importance to identify how each individual moves in the region.

We define an individual profile (IP) for each individual $i$. Each individual has an IP representing its profile in the territory under exam. It is constructed from the outcome of the clustering of the ICPs. An IP is a 3-dimensional array where each position represents the number of times $i$ is respectively considered a Resident, a Commuter and a Visitor in the region under exam. The aim of this characterization is to identify groups of individuals sharing a common behaviour. In particular, we should answer the following questions:

- how many individuals are just visitors of the region?
- how do residents of a municipality move to the region?
- do individuals exist visiting many places and performing many calls? (i.e. maybe some individuals are classified residents in multiple municipalities)?

To answer these questions, we cluster the IPs in order to aggregate similar individuals. Since we have specific questions to answer, we need to carefully choose also in this case the correct similarity metric to be used for the IPs. Again, thanks to our algorithm that supports arbitrary similarity metric, we can define a metric suitable for our data without warring about its suitability in the algorithm. In the following section, we define how we choose such metric.

#### *5.2.3 Similarity metrics for individuals profiles*

In this section, we define several metrics that can be used for clustering IPs. We think that the most important value in the IP is the number of times an individual may be considered a Resident. Note a value of 0 or 1 represents an individual that is respectively not a resident in the region under exam and a resident in one of the municipality under exam. However, it may happens that an individual has a value greater than 1. This means that such individual is moving in many municipalities and it is performing many calls in each of the municipalities. For instance, consider salespeople: individuals not having a fixed working place and whose work is mainly characterized on meeting people in different places, organizing such meetings by phone and keeping in touch with all the customers. Due to this, they are individuals that in our data will emerge having multiple ICPs and in some of them, where they are more present, having an high number of calls resulting in a Resident profile.

For all the above motivations, we need a metric capable of correctly identifying clusters keeping well separated individuals having a different value in the Resident slot. The euclidean distance is not enough to grasp such differences. For instance, it gives the same importance to the values in the resident and visitor slot. However, it is more important to differentiate between an individual being a resident in 2 municipalities from an individual resident in 3 with respect to 2 individuals being visitors respectively in 0 and 5 municipalities. Due to this, we defined a personalized metric. Such metric assigns a similarity equal to 0 to individuals having a different value for Resident. Instead, it assigns a value equal to the euclidean distance between the values of commuters and visitors for those individuals having the same value in resident.

## 6 Experimental setup

All the experiments have been conducted on a cluster running Ubuntu Linux consisting of 5 nodes (1 master and 4 slaves),

each equipped with 128 Gbytes of RAM and with two 16-cores CPU, interconnected via a Gigabit Ethernet network.

We implemented our approach using Apache Spark [18], and the source code we used for conducting our experiments is publicly available on GitHub.[1]

### 6.1 Alternative approaches

To study the performances of Muchness with respect to alternative existing approaches, we compared it against the following competitors:

- Sociometer [4] is the primary competitor, it is the most similar to Muchness; both the approaches are based on clustering and designed for the same case study;
- MP [6] targets the same problem; however, it is not based on clustering but relies on rules, such as the calling hours to identify if an individual is a resident. Such approach requires the knowledge of additional data as for instance the total amount of individuals in a region. Since such data may be affected by fluctuation or can be missing, we make use of a version of MP not requiring additional parameters;
- DBSCAN, we tried to conduct our experiments with an implementation[2] of MR-DBSCAN [14] on Apache Spark; unfortunately, we have not been able to cluster more than the 10% of the dataset due to memory errors related to the high dimensionality of the ICPs. Such algorithm has been originally conceived to work only in a two-dimensional space. At the beginning, the approach requires to partition the data points and this operation becomes difficult and more time consuming when the number of dimensions increases [19]. Due to this, we encounter memory errors when partitioning the space of the ICPs because they are arrays in 30 dimensions.

### 6.2 Evaluation metrics

We now discuss the metrics we use to analyse the performance of our approach and the most important parameters of Muchness+.

We study the comparison between Muchness and Muchness+ using well-known measures of quality [20,21]:

- **Compactness** measures how closely related the items in a cluster are. We obtain the compactness by computing the average pairwise similarity among items in each cluster. Higher values are preferred.
- **Separation** measures how well clusters are separate from each other. Separation is obtained by computing the average similarity between items in different clusters. Lower values are preferred

Note, the computation of the above metrics is computationally as hard as computing the clustering we intend to evaluate. For this reason, to perform a pairwise similarity between items, we pick items uniformly at random, with a sampling rate of 10%. Additionally, we also consider algorithm **Speed-Up**: this metric measures the algorithm runtime improvement $S_i$ where $i$ is the number of cores devoted to the computation. We set $S_i = T_b/T_i$ [22] where $T_b$ is the baseline and $T_i$ the computational time when using $i$ cores. We chose as baseline the computational time using 4 cores (one core for each machine).

## 7 Results

Through our experiments, we first study the optimizations introduced in Muchness+ with respect to the previous version called Muchness comparing both the clustering quality and the execution time. Then, we perform several considerations about the information that can be extracted using Muchness+; in particular, we study how the individuals travel from home to work and which places they visit. Finally, we perform a comparison with similar works and we evaluate the individual profiles introduced in Sect. 5.2.2 and the scalability of Muchness+.

### 7.1 How to configure Muchness+

With the optimizations introduced for Muchness+, we target to remove input parameters to facilitate the usage. Although in Sect. 5 we described how to remove all the parameters, that were previously required by Muchness, we introduced the optional parameter $\rho$ to avoid degenerate scenarios and to perform a trade-off between running time and quality. Figure 5 depicts the results when using a value of $\rho \in \{1, 2, 3, 6\}$ compared to Muchness. We found that $\rho = 1$ is a too strict configuration and does not permit to achieve a good result (i.e. the algorithm is not able to estimate correctly the number of residents). However, already with $\rho = 2$ the number of residents identified is comparable with the ones obtained with larger values of $\rho$. Also, keeping low $\rho$ permits to have a shorter running time because each node sends $\rho k$ messages. Due to this, we suggest to use $\rho = 2$ because this corresponds to the better running time and a quality similar to different configurations.

The parameter $j$ is always set equal to 10 in the experiments, and this is the default value in Muchness+. We leave the possibility to modify it; however, as described also in previous results [5,15,19], we identified in such configuration an optimal trade-off for many different scenarios. In addition,

[1] https://github.com/alessandrolulli/knnMeetsConnectedComponents.

[2] https://github.com/alitouka/spark_dbscan.

**Table 3** Road to Muchness+: adaptive $k$ optimization

| | Time (s) | #cluster | Residents | Compactness | Separation |
|---|---|---|---|---|---|
| Muchness | 3164 (±183) | 569 | 552829 | 0.89 | 0.76 |
| Adaptive $k$ | 1873 (±42) | 265 | 632393 | 0.86 | 0.72 |

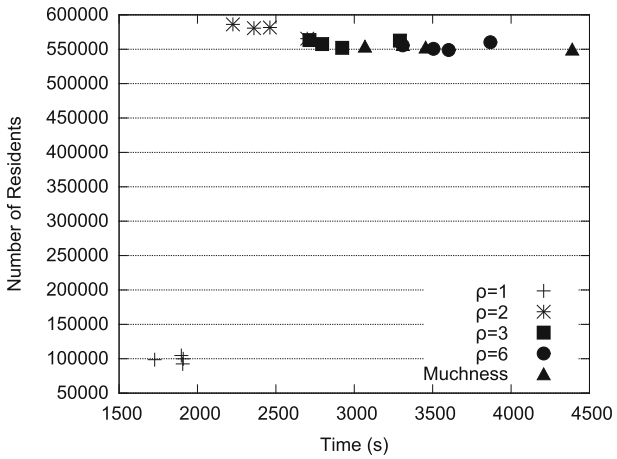


**Fig. 5** How to configure Muchness+: analysing the sampling parameter ($\rho$)

the impact of $j$ is mitigated by the adaptive $k$ feature introduced in Muchness+. This permits to automatically adapt the neighborhood of each node to the best needs.

### 7.2 Road to Muchness+: evaluating optimizations

In this section, we evaluate the impacts of the optimizations introduced in Sect. 5. We start evaluating each optimization separately. At the end of the section, we build Muchness+ and we compare it with Muchness. According to the result of the previous section, we set the sampling mechanism in all the experiments equal to $\rho = 2$. All the results are the average of 5 independent runs. For the time metric, we reported in parentheses also the 95% confidence interval.

#### *7.2.1 Adaptive k*

In the first set of experiments, we evaluate the adaptive $k$ optimization (Sect. 5.1.2). Table 3 depicts some comparison metric with respect to Muchness. In particular, with the adaptive $k$ optimization we gain two major results. First, the time to reach the solution is around the 40% less with respect to that of Muchness. This is motivated by two things. The sampling mechanism as seen in the previous section has a remarkable impact on the execution time. However, this is not the only cause of the large gain, note that in Fig. 5 all the values having $\rho = 2$ have execution time >2000 s. Here, we get an average execution time of 1873 suggesting that with respect to the execution time of $\rho = 2$ we obtain another gain thanks to the possibility to reduce the neighbour size of a large part of the nodes in the graph. This has no impact on the quality of the obtained clusters, in fact compactness and separation have similar values with respect to Muchness. The second gain regards the number of clusters. Keeping the number of clusters low has some benefits because it permits to analyse a lower number of clusters if a manual investigation is required.

#### *7.2.2 Early termination*

Next, we move to the analysis of the early termination mechanism. This optimization, described in Sect. 5.1.3, permits to avoid the need to set the $NumIter$ parameter of Muchness. Despite this advantage, it shows also execution time advantages. Table 4 describes the results and shows a comparison with Muchness. As in the previous section, we obtain a remarkable advantage in terms of execution time. Thanks to the early termination, we finish the computation in half of the time with respect to Muchness. To be more precise, the early termination ends the computation after 6 iterations. This suggests that the subsequent iterations performed by Muchness improve only marginally the result it obtains. Also, this is another confirmation that monitoring the results on iterative algorithms permits to have a deeper control about the quality of the results.

#### *7.2.3 Muchness+ versus Muchness*

Finally, we evaluate the Muchness+ algorithm inserting all the enhancements previously analysed. Here, we add also the optimization which allow to automatically select the threshold parameter for the pruning mechanism before running the connected components. Table 5 shows the final results. With all the optimizations, the execution time of Muchness+ is around 60% lower than the execution time of Muchness. In particular, it seems that the contributions of adaptive $k$ and early termination optimizations are additive and both contribute to reduce the execution time. We get also a more stable running time in the 5 executions, and we obtain only ±15 for what concerns the confidence interval of the time metric. The number of clusters is sensibly smaller in Muchness+, but this is not affecting the number of residents estimated by such approach.

**Table 4** Road to Muchness+: early termination

| | Time (s) | #cluster | Residents | Compactness | Separation |
|---|---|---|---|---|---|
| Muchness | 3164 (±183) | 569 | 552829 | 0.89 | 0.76 |
| Termination | 1697 (±80) | 293 | 586398 | 0.87 | 0.73 |

**Table 5** Muchness+ versus Muchness

| | Time (s) | #cluster | Residents | Compactness | Separation |
|---|---|---|---|---|---|
| Muchness | 3164 (±183) | 569 | 552829 | 0.89 | 0.76 |
| Muchness+ | 1309 (±15) | 161 | 634402 | 0.87 | 0.71 |

### 7.3 Studying individuals mobility

In this set of experiments, we aim to answer some questions about the indicators that can be extracted from mobile calls. For instance, it is reasonable to assume that call activity in commercial or business areas is an indicator for economic activity [10]. First of all, our method allows to use the calling behaviour to understand if an individual resides, works or is an occasional visitor in a certain place. Due to this, we answer to the following questions: (i) Is it possible to classify regions as residential, commercial or business? (ii) is it possible to show how individuals move to reach their working position? (iii) is it possible to check which are the most visited places?

motivo per

Using the outcome of Muchness+, we can analyze how individuals move into the territory under study. In particular, we analyse how individuals move from home to the working place or to visit a city. In the following figures, the thickness of the flows is proportional to the number of individuals travelling the path connecting the two municipalities.

#### *7.3.1 Individuals travelling from home to work*

One possible use of the outcome of Muchness+ might be to systematically analyse the movement from home to work. This may have multiple advantages such as observing what is the potential market for public transport. Also, this service can be very useful for statistical institutes. Figure 6 shows the main home to work flows for each municipality in Tuscany. Clearly, Florence, being the biggest city in Tuscany, is the centre of the working activity of the region. A large part of the individuals of the surrounding municipalities everyday move to Florence during the working hours. The figure also highlights the other cities of Tuscany, in order of importance: Pisa, Lucca and Livorno. Pisa, despite being smaller than other cities such as Livorno, is a centre of numerous activities and of prestigious universities. Due to this, it seems to be the second municipality, after Florence, to attract workers. From the figure, it is possible to obtain two other major insights. First, the larger centres of working activities seem to attract workers from their surrounding municipalities, note that the flows directed to the cities are from the surrounding municipalities (the edges have a clockwise direction). Second, flows exist also between the major cities. For instance, between Livorno and Pisa or Pisa and Florence. This may show the impact of rail transportation because a path connecting Livorno, Pisa and Florence by train exists.

#### *7.3.2 Individuals travelling from home to visit places*

While some statistics about systematic movements may be extracted also from census, this is not true for occasional visits. Due to this, it would be very helpful to know, for instance, who has attended an event and where they come from or how visitors are attracted in certain municipalities. This would enable to know the spread and importance of an event by measuring the attractiveness over the surrounding territory. Figure 7 depicts the flows between the municipality of residence and the visiting places. We can see that the amount of mobility that is created for occasional reasons is impressive and certainly greater than that happening systematically and/or due to working activities. Again, the figure shows that in particular the movements involving occasional travels are to the four largest Tuscan cities we have considered. These are important destinations for tourism by Italian and foreign citizens. A difference with the movements for working activity is that not only the surrounding municipalities but the individuals of quite all the municipalities travel occasionally to the major cities. For instance, from Camaiore, a small municipality in the top-left corner of the figure there are flows directed to all the cities. Also, in the figure many more municipalities are present with respect to Fig. 6. We can see that individuals travel occasionally to many more places than those they visit for working reasons.

### 7.4 Comparing with competitors and census data

In this section, we evaluate how Muchness+ is capable of being an indicator for measuring the amount of residents in a municipal area by comparing its results against MP, Sociometer and Muchness. In addition, we evaluate also the amount of estimated commuters against Sociometer and Muchness. Note the MP method is limited and specialized in

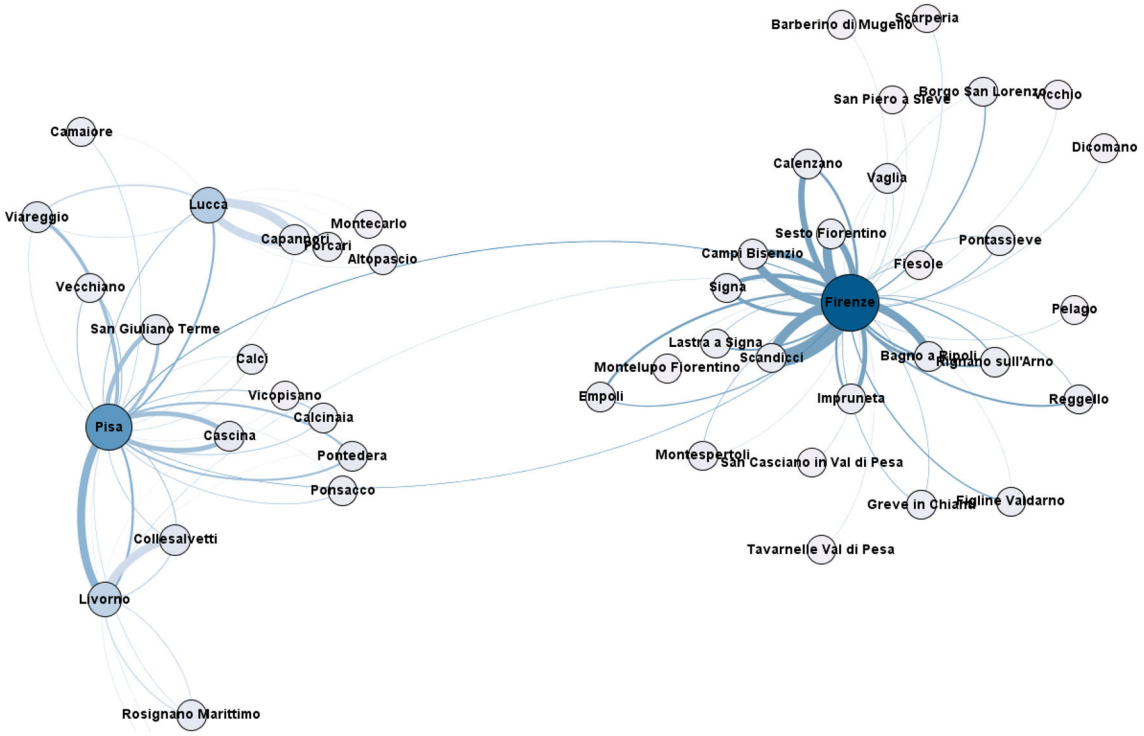


**Fig. 6** Studying individuals mobility: individuals travelling from home to work

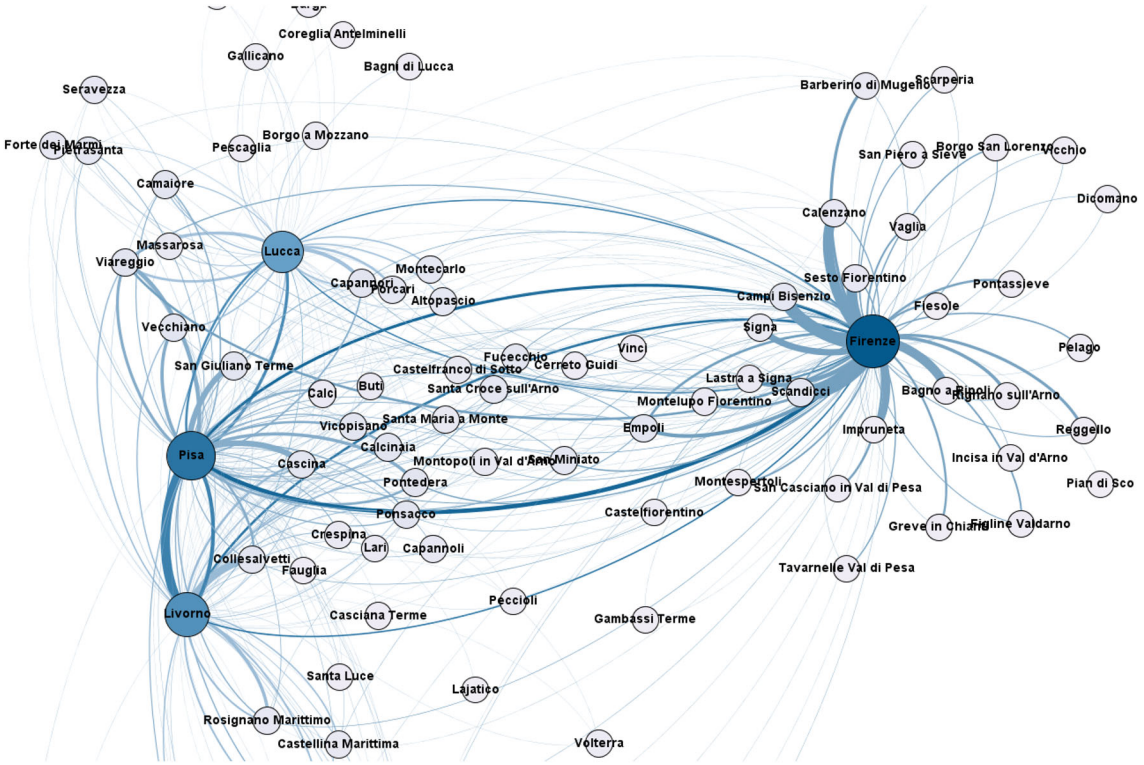


**Fig. 7** Studying individuals mobility: individuals travelling from home to visit places

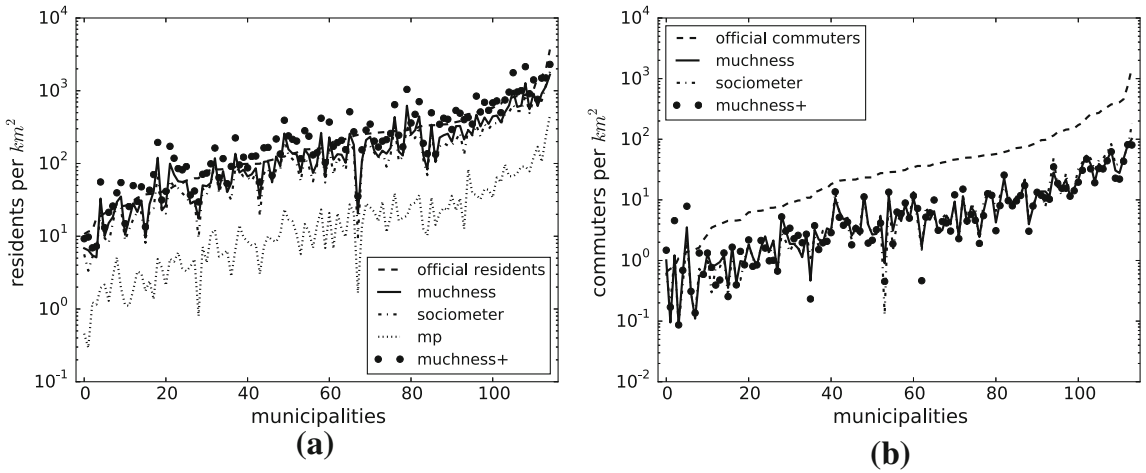


**Fig. 8** Comparing with competitors and census data. **a** Number of residents, **b** number of commuters

providing only the number of residents and does not provide any functionality to estimate commuters. It is worth to notice that all the estimations have been rescaled using the market share of our telco provider. The results are compared against official census statistics provided by Italian national institute of statistics (ISTAT). These data include the amount of residents and commuters belonging to the 115 municipalities we studied.

Figure 8a depicts the number of residents identified for each municipality. On the *Y* axis, we show the estimated population in log scale, whereas on the *X* axis the municipalities ordered from the lowest to the highest population density. The results are compared with the real census provided by ISTAT. As it can be noticed, all the methods have spikes in the same municipalities. This suggests that although the methods are based on different approaches (MP defines rules, Sociometer and ours are based on clustering), all identify similar behaviours on the data and may suggest that census data itself could under or over estimate population. It is evident that MP is always under estimating the density with an error that is greater than Sociometer, Muchness and Muchness+. Muchness+ seems the one closer to the real census data in particular for higher-dense municipalities.

To have a better insight into the errors performed on the estimations, Table 6 presents the median error on the estimations. We divided the error on the estimations in 4 areas having different population densities. Again, MP is providing the estimation affected by the larger error. Muchness and Sociometer provide similar results for the municipalities with a higher density where the volume of available data is large and the clustering can rely on a rich set of information. Instead, Muchness+, as we noted before, provides a better estimation in all the municipalities, and in particular it improves the result of Muchness on higher-dense municipalities. Finally, we compare the commuters estimations. Also in this case, the results are compared against real census data. All the approaches give approximately the same results in terms of estimation errors, for every density range.

**Table 6** Comparing with competitors and census data: median estimation errors

| | Residents × km$^2$ | | | |
|---|---|---|---|---|
| | < 50 | 50–100 | 100–150 | >150 |
| MP | 93% | 91% | 92% | 94% |
| Sociometer | 39% | 39% | 49% | 52% |
| Muchness | 24% | 29% | 42% | 47% |
| **Muchness+** | **16%** | **14%** | **15%** | **23%** |
| | Commuters × km$^2$ | | | |
| | < 50 | 50–100 | 100–150 | >150 |
| Sociometer | **83%** | 84% | 86% | 89% |
| Muchness | 84% | **83%** | **81%** | **87%** |
| **Muchness+** | 86% | 84% | 85% | 89% |

Bold values indicate the best results

### 7.5 Evaluating individual profiles

In Sect. 5.2.2, we defined the individual profile (IP) to identify different typologies of individuals and in Sect. 5.2.3 a metric capable of extracting useful information from such data. In this section, we compare two metrics for clustering individual profiles: the Euclidean distance and the one defined ad hoc for such data. Initially, we take the output of Muchness+ and we construct for each individual its IP. We obtained around 800k IP.

First of all, we compared the clusters obtained with the two metrics. Figure 9a, b depicts for each cluster a point in a 2D space where on the *X* axis is represented the number of times the exemplar of the cluster is a Visitor and on the *Y*

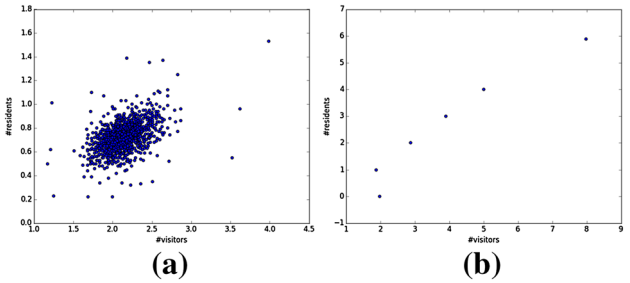


**Fig. 9** Evaluating individual profiles. **a** Euclidean distance, **b** personalized similarity metric

**Table 7** Evaluating individual profiles: exemplars

| Cluster Size | # Resident | # Commuter | # Visitor |
|---|---|---|---|
| 426767 (51%) | 0 | 0.01 | 1.98 |
| 265629 (32%) | 1 | 0.01 | 1.87 |
| 94412 (11%) | 2 | 0.02 | 2.88 |
| 32460 (4%) | 3 | 0.03 | 3.9 |
| 11001 (1%) | 4 | 0.03 | 5 |
| 6744 (< 1%) | 5.88 | 0.06 | 7.98 |

axis the number of times is a Resident. With the Euclidean distance, we obtained many more clusters with respect to the personalized metric specific for the data. With the personalized metric, we obtain a small number of clusters and each cluster is well defined and separated from the others. Thanks to this, the result is easier to be analysed and two clusters having similar characteristics do not exist.

We then proceed with a manual investigation of the clusters obtained with the personalized metric. Table 7 presents the 6 clusters obtained with their sizes and the values of the 3-dimensional array. We observe that more than the half of the individuals (51%) are not residents in the region under exam. This means that the majority of the people travelling in the region are living outside the region and visit Tuscany for tourism or for short periods of time. As expected, the second biggest cluster is the one where individuals are residents in only one municipality. However, some individuals exist that are considered Residents in more that one municipality. This is of paramount interest because highlight individuals use to travel a lot. Such individuals perform many calls in different moments of the day in different municipalities. Another interesting fact is that the individuals resulting resident in more than one place result to be also visitor of the highest amount of places. Note the value of Visitor is increasing when also the value of Resident is increasing. Also, as expected, the size of the clusters decreases when the values of resident increase. Finally, we noted that the values in the Commuter column are always close to 0. The motivation is that the number of commuters, as we found when comparing with real data, is low.

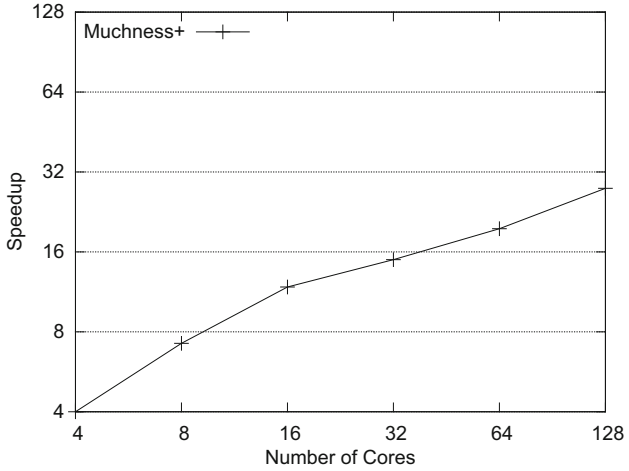


**Fig. 10** Scalability evaluation

### 7.6 Scalability

Finally, we tested the scalability of Muchness+, by varying the number of cores in the range [4,128]. Figure 10 depicts the results we achieved. Recall, Muchness+ has been built taking inspiration from two previous works for computing connected components [16,17] and text clustering [5]. In such works, similar patterns have been observed when evaluating the scalability using several different typologies of datasets. We obtained an almost linear scalability using 8 cores, a still good level scalability with 16 cores, and then the value tends to stabilize albeit it is always improving while adding more cores. These results can be motivated with several considerations about the testing environment. Spark allocates the cores according to a round robin policy: when using 4 cores, Spark exploits one core from each of the 4 machines. As a consequence, by using only 4 cores (of the 128 available) we exploit the total amount of memory available in the cluster. Considering that each machine has two CPUs, we reach the maximum available CPU-memory bandwidth, and thus linear scalability, when using 8 cores (one core per CPU).

## 8 Conclusions

This paper presents a framework for estimating the population making use of mobile calls. With respect to the existing solutions, we presented an unsupervised clustering algorithm that does not require an input from the user. In addition, it accommodates arbitrary similarity metric. We define personalized similarity metric able to capture similarities between individual call profiles, overcoming the limitations of state-of-the-art approaches that do not exploit the "shape" of the user profiles (in particular for Residents and Commuters). The ultimate aim of our research is to provide to the public

administration a tool, able to process a continuous stream of phone data to provide useful information for improving public services, such as transportation and security of the territory. To this end, we analyse how the individuals move in the territory for working reasons or to visit places. We empirically proved, through an extended experimental testbed, that our extended approach is able to provide a better estimation of the population and in less time with respect to the previous version. This is mainly due to its ability to early terminate the execution and to automatically limit the amount of messages when only marginal improvements can be performed. Furthermore, we give an experimental evidence that our approach provides a very good estimation of the population density within the Italian region of Tuscany. As future work, we plan to extend the work to provide a real-time estimation of the population, to perform event detection and to consider data associated with a larger region.

**Acknowledgements** This work is partially supported by the European Community's H2020 Program under the scheme 'INFRAIA-1-2014-2015: Research Infrastructures', Grant Agreement #654024 'SoBigData: Social Mining & Big Data Ecosystem'. (http://www.sobigdata.eu).

## References

1. Xu, K., Zou, K., Huang, Y., Yu, X., Zhang, X.: Mining community and inferring friendship in mobile social networks. Neurocomputing **174**, 605–616 (2016)
2. Ratti, C., Frenchman, D., Pulselli, R.M., Williams, S.: Mobile landscapes: using location data from cell phones for urban analysis. Environ. Plan. B Plan. Des. **33**(5), 727–748 (2006)
3. Calabrese, F., et al.: Real-time urban monitoring using cell phones: a case study in rome. IEEE Trans. Intell. Transp. Syst. **12**(1), 141–151 (2011)
4. Gabrielli, L., et al.: City users' classification with mobile phone data. In: 2015 IEEE International Conference on Big Data, pp. 1007–1012 (2015)
5. Lulli, A., et al.: Improving population estimation from mobile calls: a clustering approach. In: 2016 IEEE Symposium on Computers and Communication (ISCC). IEEE (2016)
6. Deville, P., et al.: Dynamic population mapping using mobile phone data. Proc. Natl. Acad. Sci. **111**(45), 15888–15893 (2014)
7. Ratti, C., et al.: Mobile Landscapes: Graz in Real Time. Springer, New York (2007)
8. Ahas, R., et al.: Using mobile positioning data to model locations meaningful to users of mobile phones. J. Urban Technol. **17**(1), 3–27 (2010)
9. Etter, V., et al.: Where to go from here? Mobility prediction from instantaneous information. Pervasive Mob. Comput. **9**(6), 784–797 (2013)
10. De Jonge, E., van Pelt, M., Roos, M.: Time patterns, geospatial clustering and mobility statistics based on mobile phone network data. In: Paper for the Federal Committee on Statistical Methodology research conference, Washington, USA (2012)
11. Terada, M., Nagata, T., Kobayashi, M.: Population estimation technology for mobile spatial statistics. NTT DOCOMO Techn. J. **14**, 10–15 (2013)
12. Furletti, B., et al.: Use of mobile phone data to estimate mobility flows. measuring urban population and inter-city mobility using big data in an integrated approach. In: Proceedings of the 47th Meeting of the Italian Statistical Society (2014)
13. Ester, M., et al.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Kdd, pp. 226–231 (1996)
14. He, Y., et al .: Mr-dbscan: an efficient parallel density-based clustering algorithm using mapreduce. In: 2011 IEEE International Conference on Parallel and Distributed Systems, pp. 473–480. IEEE (2011)
15. Lulli, A., et al.: Scalable k-nn based text clustering. In: 2015 IEEE International Conference on Big Data, pp. 958–963. IEEE (2015)
16. Lulli, A., Ricci, L., Carlini, E., Dazzi, P., Lucchese, C.: Cracker: crumbling large graphs into connected components. In: 2015 IEEE Symposium on Computers and Communication (ISCC), pp. 574–581. IEEE (2015)
17. Lulli, A., Carlini, E., Dazzi, P., Lucchese, C., Ricci, L.: Fast connected components computation in large graphs by vertex pruning. IEEE Trans. Parallel Distrib. Syst. **28**(3), 760–773 (2017)
18. Zaharia, M., Chowdhury, M., Franklin, M.J., Shenker, S., Stoica, I.: Spark: cluster computing with working sets. HotCloud **10**, 10–10 (2010)
19. Lulli, A., Dell'Amico, M., Michiardi, P., Ricci, L.: Ng-dbscan: scalable density-based clustering for arbitrary data. Proc. VLDB Endow. **10**(3), 157–168 (2016)
20. Desgraupes, B.: Clustering indices. Univ. Paris Ouest-Lab ModalX **1**, 34 (2013)
21. Liu, Y., Li, Z., Xiong, H., Gao, X., Wu, J.: Understanding of internal clustering validation measures. In: 2010 IEEE 10th International Conference on Data Mining (ICDM), pp. 911–916. IEEE (2010)
22. Patterson, D.A.: Computer Architecture: A Quantitative Approach. Elsevier, Amsterdam (2011)