Mining Constrained Regions of Interest: An Optimization Approach

Alexandre Dubray(✉), Guillaume Derval, Siegfried Nijssen, and Pierre Schaus

UCLouvain - ICTEAM/INGI, Louvain-la-Neuve, Belgium
{alexandre.dubray,guillaume.derval, siegfried.nijssen,pierre.schaus}@uclouvain.be

Abstract. The amount and diversity of mobile and IoT location and trajectory data are increasing rapidly. As a consequence, there is an emerging need for flexible and scalable tools for analyzing this data. In this work we focus on an important building block for analyzing location data, that is, the problem of partitioning a space into regions of interest (ROIs) that are densely visited. The extraction of ROIs is of great importance as it constitutes the first step of many types of data analysis on mobility data, such as the extraction of trajectory patterns expressed in terms of sequences of ROIs. However, in this paper we argue that unconstrained ROIs are not meaningful and useful in all applications. To address this weakness, we propose the problem of constraint-based ROI mining, and identify two types of constraints: intra- and inter-ROI constraints. Subsequently, we propose an integer linear programming formulation of the task of discovering a fixed number of constrained ROIs from a binary density matrix. We extend the approach to discover automatically the number of ROIs by relying on the Minimum Description Length Principle. Our experiments on real data show that the approach is both flexible, scalable and able to retrieve constrained ROIs of higher quality than those extracted with existing approaches, even when no constraints are imposed.

Keywords: Data mining · Constrained optimization · Integer linear programming · Regions of interest · Constrained clustering

1 Introduction

The number and diversity of tracking devices are constantly increasing and so does the volume of recorded location data. Innovative applications exploiting these data can be imagined if some form of meaningful aggregated information can be discovered. An important building block for summarizing trajectory data is the extraction of regions of interest (ROIs). Informally, a ROI is a densely visited space. The discovery of ROIs is of practical importance as it can be instrumental for other tasks. Examples of such tasks related to trajectory mining are:

– In [8], the authors propose to discover trajectory patterns expressed in terms of ROIs. They first rewrite the trajectories as a sequence of the extracted

ROIs. A frequent sequence mining algorithm [1,4] can then be applied on the sequence database to extract sequential patterns with a minimum support.
- Another possible use of ROIs is location prediction. This task consists in, given a database of trajectories and the start of a new trajectory of a moving object, predicting what will be the next location of the moving object [12,15].
- In the area of urban management [18], the authors proposed a system relying on ROIs to help taxis to wait in a region likely to contain their next trip request.

However, not all ROIs are equally useful and meaningful in all applications. For example, in the case of tourist spot recommendation, it may be desirable that the extracted ROIs are close to public transport access; in an application suggesting visiting a city by bicycle it is useful to impose a constraint that extracted ROIs are close enough to bike paths, and are within reasonable distance from each other. No existing approaches for identifying ROIs take such constraints into account. For this reason, in this paper we introduce the problem of *constraint-based ROI mining*. We categorize these constraints into two types: *intra-ROI constraints*, which impose requirements on the individual ROIs, and *inter-ROI constraints*, which impose requirements on the relationships between ROIs. In this work, we propose an Integer Linear Program, that can directly incorporate the two types of constraints, to solve this problem.

2 Preliminary Concepts

A well-known algorithm for identifying ROIs is the *PopularRegion* algorithm [8] that is both easy to implement and scalable. This algorithm extracts non-overlapping rectangular ROIs from a 2D grid of density values $\mathcal{G}$ of size $N \times M$ (N rows and M columns). This grid-based approach enables application dependent density definitions. For analyzing trajectory data, the density of a cell can be the number crossing trajectories with or without interpolation between consecutive points. If one is rather interested to detect geographic regions where users stay for a significant amount of time (Stay Points) [11], one can define the density as the relative fraction of time spent in the cell by a trajectory.

The *PopularRegion* algorithm works as follows. Starting from a small ROI, it greedily expands the rectangle ROI in one of the four directions as long as the average density of the rectangle remains above a given threshold. Using the same notation as in [8], c_{ij} is the cell at row i and column j ($1 \leq i \leq N$, $1 \leq j \leq M$), θ is a user defined minimum density threshold and $\mathcal{G}^* = \{c \in \mathcal{G} \mid \text{density}(c) \geq \theta\}$ is the set of all dense cells. The algorithm works as follows:

1. Take the cell in $\mathcal{G}^*$ with the highest density that is not already in a ROI. If there is none, return the set of ROIs.
2. Create a ROI with this single cell.
3. While there is a direction in which we can extend the ROI, extend it in the direction that gives the highest average density.
4. Add the ROI to the set of ROIs and go to 1.

The main advantage of the *PopularRegion* algorithm is its scalability and ease of implementation. However, it clearly also has a number of weaknesses. First its output is ill-defined; there is no clear characterization of an objective function that is minimized. Furthermore, as explained in [9], it is easy to create examples where the greedy algorithm ends up finding very large ROIs that may hinder the creation of interesting subregions. This is illustrated in Fig. 2 and 3, which show the initial dense cells as well as the regions discovered by *PopularRegion*, for two different data sets. As can be observed, for both data sets *PopularRegion* identifies regions that cover large part of the city, which is not satisfying. Finally, *PopularRegion* does not allow constraining the discovered ROIs and it can only generate rectangular ROIs.

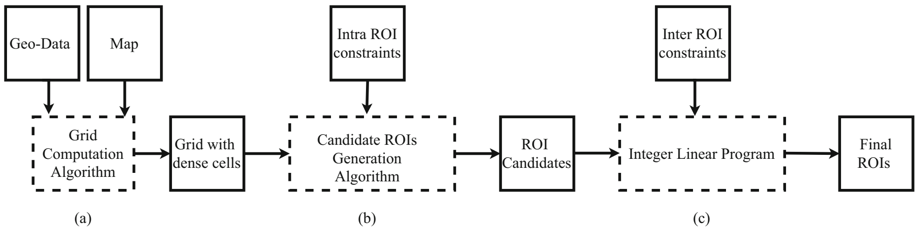


Fig. 1. Our approach is decomposed into consecutive steps: (a) The grid $\mathcal{G}$ is created from the geolocalized data; (b) A set of candidate ROIs is generated satisfying intra-ROIs constraints; (c) The set of ROIs is selected among the candidate ROIs solving the Integer Linear Program taking into account inter-ROI constraints.

Our contribution is the process for discovering constraint-based ROIs given in Fig. 1:

– The grid and the dense cells are computed based on the map and the geo-localized data. Many alternatives are possible depending on the application.
– A set of candidate ROIs are computed. The final ROIs will be selected from this set. These ROIs must satisfy the intra-ROI constraints such as the minimum distance to public transportation, the shape constraints, etc.
– An Integer Linear Program (ILP) selects the final ROIs. It consists in finding the most parsimonious representation of all the dense cells. Two variants are proposed: one with a fixed number K of regions and one in which this number is chosen automatically by relying on the *minimum description length* (MDL) principle [14]. The ILP can easily accommodate inter-cluster constraints such as the minimum distance between any two selected ROIs.

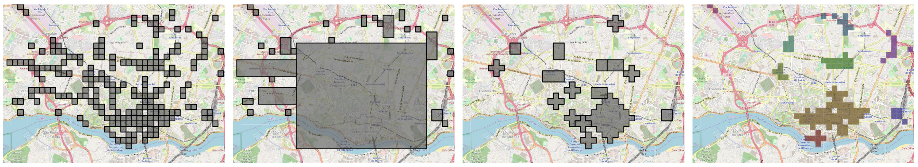
This paper focuses mainly on the generation of candidate ROIs and the ROI selection algorithm (the ILP). The grid and dense cells generation is left to the user: it is an orthogonal task which must be adapted to the task at hand.

This approach for detecting ROIs addresses a number of weaknesses of the *PopularRegion* algorithm. In particular, it can easily accommodate constraints on ROIs and the optimization problem for discovering the ROIs is well-defined.

We evaluate the new approach qualitatively and compare it with the *Popular-Region* and OPTICS [3] algorithms on real-life data. As alternative approaches do not support constraints, we also evaluate our approach without constraints.

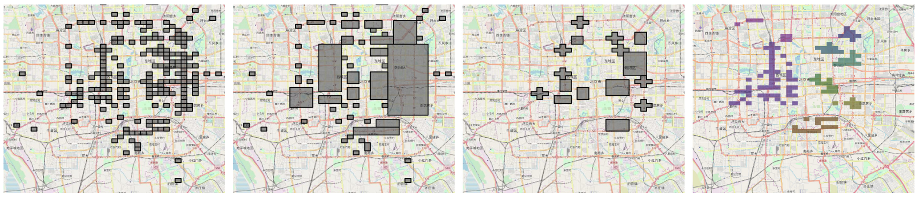
An example of regions discovered by our method is illustrated in Figs. 2c and 3c. As can be seen, our method finds more fine-grained ROIs and avoids selecting all the isolated cells.

Related work is discussed in Sect. 3. Our optimization model is introduced in Sect. 4. The candidate ROIs generation is discussed in Sect. 5, as it is dependent on the optimization model, and the addition of constraints is describe in Sect. 6. The experiments are described in Sect. 7. We conclude in Sect. 8.



(a) Dense cells on a 100 × 100 grid (b) *PopularRegions* (c) Our method with ratio constraints on the rectangles (d) OPTICS, each color represents a cluster

Fig. 2. Visualization of the output of the different methods for the Kaggle data set.



(a) Initial set of dense cells on a 200× 200 grid (b) *PopularRegions* (c) Our method with ratio constraints on the rectangles (d) OPTICS, each color represents a cluster

Fig. 3. Visualization of the output of the different methods for the T-Drive data set.

3 State of the Art and Related Work

Like *PopularRegions*, Gorawski and Jureczek [9] proposed a grid-based approach to identify ROIs. The algorithm is essentially *PopularRegions* with a limit on the size of the rectangles during the extension process. It requires additional parameters and does not permit constraining the ROIs.

The approach of [5] is not grid-based. Starting from geo-tagged locations on the map, it discovers dense convex polygons around predefined points-of-interest

(PoI). The fixed PoI setting limits the use cases of the approach and the fact that it is not grid-based also limits the possible applications. Furthermore, shape constraints on the ROIs are not possible.

The task of finding ROIs on a grid is similar to clustering. Starting from a grid of dense cells, any clustering method can be used to group dense cells close to each other. However, the problem is not exactly the same. Clusters of dense cells are not necessarily connected regions. DBSCAN [7] is one of the most popular density-based clustering algorithms. It does not require to specify the number of clusters and is also able to identify outlier points. OPTICS [3] is another well-known method to perform density-based clustering that is able to deal with clusters of varying density. Examples of output of OPTICS are shown in Figs. 2d and 3d. OPTICS identifies clusters of various forms since they are not constrained by the algorithm.

In [6], the authors propose a clustering method computing connected component sets of dense cells starting from the rectangular regions found by *Popular-Region*. This method is not able to filter outlier cells like DBSCAN or OPTICS and does not accept constraints on the ROIs.

4 An Optimization Model for ROIs

This section describes the optimization model used in step (c) of Fig. 1. The model is in charge of selecting the final ROIs from a set of precomputed candidate ROIs denoted $\mathcal{S}$ (shapes). We formalize the problem as an *integer linear program* (ILP). For simplicity we first assume that $\mathcal{S}$ is composed of rectangles and that the desired number of ROIs to select is fixed to K. Subsequently, we will extend the approach to discover automatically the number of regions K, by using the Minimum Description Length Principle [14]. We will first introduce our approach when no constraints are given; how to deal with constraints is discussed in Sect. 6.

4.1 Selection of K ROIs

Assuming that the set of candidate rectangles is composed of all the possible rectangles, our approach aims to find K non-overlapping rectangles that cover the dense cells well and avoid covering the non-dense ones. For a $N \times M$ grid, there are less than $N^2M^2 = |\mathcal{G}| \times |\mathcal{G}|$ such rectangles, that is, the total number of possible pairs of coordinates.

The approach can be interpreted as discovering a classification model for predicting the dense-non-dense status of a cell solely based on its coordinates. The prediction function to discover is chosen from a hypothesis space composed of the power-set of non-overlapping shapes from $\mathcal{S}$. Of course, such a prediction model will make a number of errors: the non-dense cells contained in some selected rectangle and the dense cells not covered by any selected rectangle. In the example of Fig. 4, the model has selected two rectangles and makes four prediction errors: the cells (4, 3) and (6, 7) are non-dense cells covered by a rectangle, and the cells (6, 2) and (7, 8) are dense cells not covered by a rectangle.

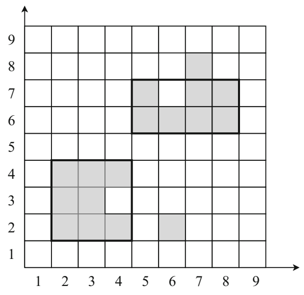


Fig. 4. Model example

The Integer Linear Program. The selection status of every candidate $R_i \in \mathcal{S}$ is modeled with one binary variable $x_i \in \{0,1\}$. The set of selected shapes is noted $\mathcal{R} = \{R_i \in \mathcal{S} \mid x_i = 1\}$. By abuse of notation, we also use $\mathcal{R}$ to denote the set of covered cells $\bigcup_{x_i=1} R_i$. (Un)covered cells are captured in the model using binary variables $cov_c \in \{0,1\}$, where $cov_c = 1 \iff c \in \mathcal{R}$.

Given that $\mathcal{G}$ ($\mathcal{G}^*$) is the set of (dense) cells, the dense cells not covered by any rectangle are denoted as $error^+ = \{c \in \mathcal{G}^* \mid cov_c = 0\}$ and the non-dense cells covered by some rectangles are denoted as $error^- = \{c \in \mathcal{G} \setminus \mathcal{G}^* \mid cov_c = 1\}$. We hence wish to discover the set of rectangles that minimizes the error: $\arg\min_{\mathcal{R}} |error^+| + |error^-|$. The complete model is given next.

$$\text{minimize} \sum_{c \in \mathcal{G}^*} (1 - cov_c) + \sum_{c \in (\mathcal{G} \setminus \mathcal{G}^*)} cov_c \tag{1a}$$

subject to

$$\textstyle\sum_{R_i \in \mathcal{S}} x_i \leq K \tag{1b}$$

$$\textstyle\sum_{R_i \in \mathcal{S} \mid c \in R_i} x_i \leq 1 \qquad \forall c \in \mathcal{G} \tag{1c}$$

$$x_i \leq cov_c \qquad \forall R_i \in \mathcal{S}, \forall c \in R_i \tag{1d}$$

$$cov_c \leq \textstyle\sum_{R_i \in \mathcal{S} \mid c \in R_i} x_i \qquad \forall c \in \mathcal{G} \tag{1e}$$

$$x_i \in \{0,1\} \qquad \forall R_i \in \mathcal{S} \tag{1f}$$

$$cov_c \in \{0,1\} \qquad \forall c \in \mathcal{G} \tag{1g}$$

The constraint (1b) limits the number of selected rectangles to K. The constraints (1c) prevent selecting overlapping rectangles. The constraints (1d) and (1e) ensure $cov_c = 1 \Leftrightarrow \exists x_i = 1 : c \in R_i$.

We further improve this model to get rid of the $|G| \times |\mathcal{S}|$ constraints (1d) and (1e), and the binary variables cov_c. This new model relies on the next theorem stating that the value $|error^+| + |error^-|$ can be inferred solely based on the number of dense and non-dense cells covered by the rectangles.

Theorem 1. *By denoting d_i (resp. u_i) the number of dense (resp. non-dense) cells covered by the rectangle R_i, it follows that*

$$\arg\min_{\mathcal{R}} |error^+| + |error^-| \Leftrightarrow \arg\min_{\mathcal{R}} \sum_{R_i \in \mathcal{R}} (u_i - d_i).$$

Proof. The term $|error^+|$ can be written as $|\mathcal{G}^*| - \sum_{R_i \in \mathcal{R}} d_i$. The term $|error^-|$ is $\sum_{R_i \in \mathcal{R}} u_i$. It follows that

$$\begin{aligned}\arg\min_{\mathcal{R}} |error^+| + |error^-| &= \arg\min_{\mathcal{R}} |\mathcal{G}^*| - (\sum_{R_i \in \mathcal{R}} d_i) + (\sum_{R_i \in \mathcal{R}} u_i) \\ &= \arg\min_{\mathcal{R}} \sum_{R_i \in \mathcal{R}} (u_i - d_i)\end{aligned}$$

□

The linear program to solve is then the following:

$$\text{minimize} \sum_{R_i \in \mathcal{S}} x_i \cdot (u_i - d_i) \tag{2a}$$

subject to

$$\textstyle\sum_{R_i \in \mathcal{S}} x_i \leq K \tag{2b}$$

$$\textstyle\sum_{R_i \in \mathcal{S} | c \in R_i} x_i \leq 1 \qquad \forall c \in \mathcal{G} \tag{2c}$$

$$x_i \in \{0, 1\} \quad \forall R_i \in \mathcal{S} \tag{2d}$$

with Eq. (2b) limiting the number of regions and Eq. (2c) enforcing non-overlap between the regions. The problem of ROI selection is thus reduced to an instance of the Maximum Weighted Independent Set problem with an additional cardinality constraint [10] that is generally solved with integer programming solvers.

ROIs of Arbitrary Shape. The integer linear model (2) does not require that candidate regions are rectangular. Any shape that covers a set of cells can be included in the candidate set $\mathcal{S}$. In particular, a circular region $Circ = (row, col, radius)$ defined by its center and radius is a natural ROI candidate. The circular region covers the cells $Circ = \{c_{ij} \in \mathcal{G} \mid |i - row| + |j - col| \leq radius\}$ (assuming Manhattan distance). We further discuss the generation of candidate regions in Sect. 5.

4.2 A Parameter-Free Approach

Fixing the limit K for the maximum number of ROIs can in some cases be an arbitrary decision. We can use the *Minimum Description Length* (MDL) principle [14] to determine the size of a model in a principled manner. MDL trades off the description length of the data given the model, and the description length of the model itself. More precisely, let us assume that we have a set of models (hypothesis) $\mathcal{H}$. The description *length* of the model $L(H)$ is the number of bits needed to encode the model; the description length of the data $L(D \mid H)$ is the number of bits needed to encode the data given the model H. The MDL principle tells us to prefer the model that minimizes $L(D, H) = L(H) + L(D \mid H)$.

The model described in the previous sections is composed of a choice of multiple ROIs, indicating where the cells must be dense, along with errors of the model, that is, coordinates of cells which are included in a selected ROI but are non-dense, and dense cells outside the selected ROIs. Each of these can be encoded using a different number of integers:

- 4 integers per rectangle (top-left and bottom-right corners' coordinates)
- 3 integers per circle (center coordinate and radius)
- 2 integers per wrongly classified cells (coordinates of the cell)

A fixed number of bits are used for every integer.

The length to encode the prediction model $\mathcal{S}$ (selected ROIs), and the input cells in this model is:

$$L(\mathcal{S}) = \sum_{R_i \in \mathcal{S}} size(R_i) \qquad\qquad L(\mathcal{G} \mid \mathcal{S}) = 2 \cdot (|\mathcal{G}^*| + \sum_{R_i \in \mathcal{S}} (u_i - d_i)),$$

where $size(R_i)$ is the number of integers required to encode R_i (3 if R_i is a circle and 4 if it is a rectangle, for example). The integer $|\mathcal{G}^*| + \sum_{R_i \in \mathcal{S}} (u_i - d_i)$ counts the number of errors made by the model and the factor 2 accounts for encoding the two coordinates of each exception cell.

We can use the MDL criterion to discover the ROIs without fixing their number in advance. To find regions that minimize the MDL criterion, we solve the following ILP model:

$$\text{minimize} \sum_{R_i \in \mathcal{S}} x_i \cdot (2(u_i - d_i) + size(R_i)) \tag{3a}$$

subject to

$$\textstyle\sum_{R_i \in \mathcal{S} \mid c \in R_i} x_i \leq 1 \qquad \forall c \in \mathcal{G} \tag{3b}$$

$$x_i \in \{0, 1\} \quad \forall R_i \in \mathcal{S} \tag{3c}$$

Note that we do not exactly minimize $L(\mathcal{G}, \mathcal{S})$. Indeed we removed the constant $2 \cdot D$ as it does not impact the optimization. This problem is an instance of the Maximum Weighted Independent Set problem.

5 Generation of Candidate ROIs

In this work we assume a generic generate and filter approach based on a set of predicates for the candidate regions. The time needed to solve the ILP grows with the number of candidate shapes as each one requires the introduction of one binary decision variable. We show how to reduce the number of candidates while ensuring that the solution found is still optimal.

Table 1. Impact of the filtering on the set of possible candidates before and after the filtering.

Min. density threshold	Grid Size	# candidates	# remaining candidates
2	100	25 502 500	17 218
	150	128 255 625	7 703
	200	404 010 000	3 330
5	100	25 502 500	2 523
	150	128 255 625	1 255
	200	404 010 000	448

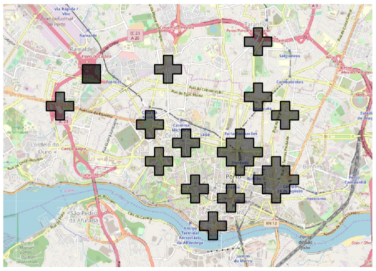


Fig. 5. Output of our method with a minimum distance constraint of 2 and a maximum diameter constraint of 5.

In the worst case, without any filtering, the number of possible rectangles is still polynomial in the size of the grid; more precisely, for a $N \times M$ grid, there are less than N^2M^2 possible rectangles (all the coordinates $(x^1, y^1), (x^2, y^2)$). There is also a polynomial number of circles. Fortunately, one can avoid generating all the candidates. Obviously, we can directly filter out all the candidates R_i for which $2(u_i - d_i) + size(R_i) > 0$. Indeed, in that case the cost of taking the candidate $(2u_i + size(R_i))$ is higher than the cost of not selecting it $(2d_i)$. Moreover if a rectangle contains a set of contiguous rows (or columns) that cover u non-dense cells, d dense cells and the inequality $u > d + 2$ holds, then this rectangle is not part of the optimal solution. The intuition behind this property is that by removing the rows (or columns) from the rectangle, we create two rectangles that yield a smaller description length. Indeed, the gain in description length $(2u)$ is higher than its increase $(2d + 4)$.

As an example of the effectiveness of the filtering, Table 1 shows, for multiple configurations on a Kaggle data set, the total number of distinct rectangles before and after the filtering.

6 Finding Constrained ROIs

As explained in Fig. 1 the ROIs can be constrained in two different ways: with intra- or with inter-ROI constraints.

Intra-ROI Constraints are the ones that must be satisfied independently by each ROI such as "a ROI contains at least one bus stop" or "is at a distance less than 100 m from a train station". These constraints define predicates that must be satisfied by each region. These constraints are taken into account by the algorithm that generates the set of candidates; in its most basic form, this algorithm generates candidates which are filtered using the constraints.

Inter-ROI Constraints are the ones that involve more than one ROI. For instance, “two ROIs must be separated by a minimum distance to ensure diversity in a tourist recommendation system”. Such constraints can be modeled in the integer linear program within the non-overlapping constraint (3b) by also including in the sum range all the candidate ROIs within a given radius distance from the cell. The ILP model can also be extended to accommodate constraints that concern only a subset of the regions that cover a cell; in principle, any constraint that can be modeled using linear equations can be added to the model.

Figure 5 shows the output of our method with a minimum distance constraint of 2 between the ROIs and a maximum diameter of 5, and illustrates how the introduction of constraints allows the ROIs to be more diverse.

7 Results and Comparison

Our experiments compare the PopularRegion algorithm with our new approach on both real and synthetic data. Clustering techniques are not producing ROIs with predefined shapes and thus explore an incomparable hypothesis space. We nevertheless include the OPTICS clustering algorithm as an optimistic baseline in our comparisons, assuming that the clusters discovered constitute the prediction function for the density status of cells.

We did not include in this experiments the works of [2,13] since they use application dependent semantic information. The method proposed in [6] also is not evaluated as it finds an exact cover of all the dense cells without generalizing with regions excluding outlier cells like OPTICS.

For the rest of this section, we denote by *ILP* our full model (i.e. Eqs. (3a)–(3b) that includes rectangular and circular ROIs while *ILP-rectangles* denote a restricted model containing only rectangular ROIs[1]. In both models we impose a ratio constraint on the width and height of the rectangles (one can not be more than two times the other) to avoid pathological solutions.

In this section we will address the following questions: i) How well does our method perform compared to *PopularRegion* and OPTICS? ii) How efficient is our approach and what is the computation bottleneck ? iii) Is our method robust to noise in data?

7.1 Performances with Respect to the MDL Criterion

We first describe an experiment performed on two real-world data sets. The first one comes from the taxi destination prediction challenge that was organized by the 2015 ECML/PKDD conference and proposed as a Kaggle competition. This data set contains more than 1.6 million trajectories from taxis of the city of Porto[2]. The second data set is the T-Drive data set from Microsoft and contains

[1] The Python code of our model is accessible here https://github.com/AlexandreDubray/mining-ROI.

[2] The data set can be downloaded at this link https://www.kaggle.com/crailtap/taxi-trajectory/home. We filtered out incomplete trajectories and the few trajectories that went too far away from Porto.

GPS traces from taxis of Beijing [16,17]. For the Kaggle data set, the density threshold will be expressed as a percentage of the total number of trajectories and we used a 100×100 grid. For the T-Drive data set, we used a 200×200 grid and, since we do not have separate trajectories, the density threshold is a percentage of the maximum density in the grid.

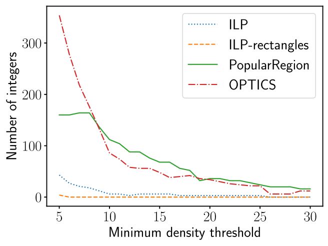
OPTICS requires two parameters: *minPts*, a threshold to be a core point, and ξ, a distance ratio to separate the clusters. Details about the parameters can be found in [3]. In our experiments, we set $minPts = 3$ since it is the threshold at which our method considers a candidate interesting. We set $\xi = 0.05$, but this parameter has almost no effect on the results in our experiments.

Figures 6b and 6d show the number of integers needed to encode the errors made by the models (2 per cell wrongly classified), in function of the minimum density threshold. As explained before and illustrated in Figs. 2b and 3b, for low-density thresholds, *PopularRegion* tends to create large regions, which results in a high number of errors since it covers many non-dense cells.

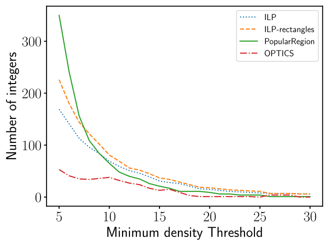
OPTICS selects in its clusters all the cells not considered noise; it is thus expected that it will make few errors, at the expense of a larger model length. Recall that OPTICS does not explore the same hypothesis space. It can thus only be interpreted as a baseline when comparing the errors. Our method discovers regions that generalize well the initial distribution of the dense cells, and allows some non-dense cells in the ROIs. The number of errors is generally between the ones of *PopularRegion* and OPTICS. When the minimum density threshold increases, OPTICS and our approach perform slightly worse than *PopularRegion*. The reason is that *PopularRegion* will overfit perfectly the isolated dense cells by creating one region for each, which is obviously not the expected behavior of an algorithm for detecting ROIs. As expected, the addition of circular shapes permits decreasing slightly the number of errors over the rectangle model since it augments the capacity of the prediction function.

Figures 6a and 6c show the number of integers needed to encode the ROIs (i.e. the first part of the MDL criterion, excluding the values needed to encode errors). Our method always gives a smaller value, with and without circular regions. It can be seen that for a high density threshold, the number of ROIs tends to zero as it is more advantageous to store the exceptions directly rather than using ROIs (the number of dense cells decreasing). When the minimum density threshold becomes larger, the dense cells become sparse over the map and OPTICS considers them as noise without identifying any cluster.

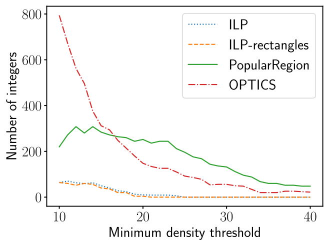
Figure 6 shows that our method outperforms *PopularRegion* on low threshold values by having less errors and nonetheless using fewer ROIs. For a higher value, our methods maintain a similar number of errors as *PopularRegion* while using at least four times less ROIs. Compared to OPTICS, we have a more errors due to the inclusion of non-dense cells in the ROIs, but our ROIs require fewer integers for their encoding. This is only valid due to the balance imposed by the usage of MDL: in general, for a fixed number of ROIs our method will have a smaller error than PopularRegion, and for a fixed error it will have a smaller number of ROIs, by design.



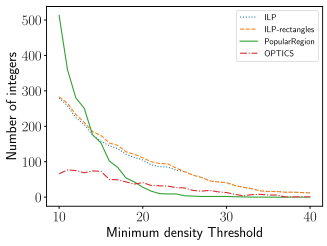
(a) Model length on the Kaggle data set



(b) Errors encoding length on the Kaggle data set



(c) Model length on the T-Drive data set



(d) Errors encoding length on the T-Drive data set

Fig. 6. Error percentage and length of the models in function of the minimum density threshold on the Kaggle data set (a)–(b) and the T-Drive data set (c)–(d).

7.2 Execution Times

Table 2 shows the run time of the methods for two minimum density thresholds and three grid sizes for the Kaggle data set. We limit the size of the grid to 200×200, which corresponds a cell size of 50×50 m. Working beyond this limit seems unreasonable given the accuracy of GPS data. For the ILP model, we show the time needed to solve the optimization problem defined in Eqs. (3a)–(3b). The table also shows the total number of dense cells in the grid as well as the number of candidate shapes.

Table 2. Run time of the methods for different grid sizes and minimum density thresholds for the Kaggle data set.

Minimum density threshold	2%			5%		
Grid side size	100	150	200	100	150	200
Number of dense cells ($\|\mathcal{G}^*\|$)	571	597	537	230	178	137
Number of ILP candidates	23 814	7 779	3 399	2 880	1 232	434
ILP optimization time (s)	4.328	0.464	0.109	0.113	0.044	0.029
PopularRegion run time (s)	0.003	0.005	0.006	0.002	0.003	0.004
OPTICS run time (s)	0.209	0.222	0.200	0.084	0.065	0.051

With its greedy behavior, *PopularRegions* obtains the best run time for all configurations. The run time of our method is mostly determined by the number of candidate shapes as these correspond to the number of variables in the model. We see that when the number of candidates becomes low enough our method has a run time that is similar to OPTICS. In a more constrained application, the set of candidates is expected to be smaller and the constraints stronger, which makes our method practical for identifying constrained regions of interest.

7.3 Robustness to Noise

To evaluate the robustness of the approaches to noise, we start from the Kaggle dataset, which consists of trajectories (i.e. series of points in space and time), and generate the grid by dividing the space in 100×100 cells of uniform size. The dense cells are chosen as being the ones with a minimum density threshold of 0.05 (i.e. at least 5% of the trajectories visit these cells). By running the methods, we obtain for each of them a set of selected ROIs $\mathcal{R}$. We then introduce noise by modifying the trajectory data points: for a level of noise p, each element of a trajectory has a probability p to be moved; if it is moved, its new position is chosen randomly in the square of 10×10 cells around the initial point. By running the methods again, we obtain new sets of selected ROIs under noise $\mathcal{R}'$.

We compute the recall $|\mathcal{R} \cap \mathcal{R}'|/|\mathcal{R}|$, the precision $|\mathcal{R} \cap \mathcal{R}'|/|\mathcal{R}'|$ and the F1-measure $(2 \cdot \mathit{precision} \cdot \mathit{recall})/(\mathit{precision} + \mathit{recall})$. Figure 7 shows how these metrics evolve with the level of noise.

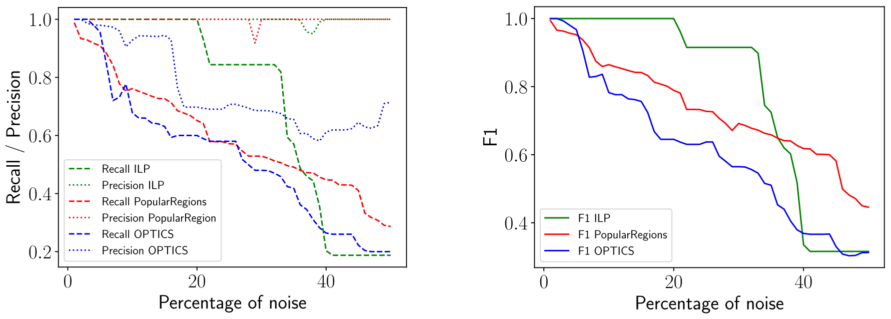


Fig. 7. Recall, precision an F1-measure w.r.t the original data in function of the percentage of noise, on the Kaggle data set.

Both *PopularRegions* and our method obtain almost always a precision of 1.0. This means that these methods do not cover areas that were not covered before. However, their recalls decrease, meaning that the found regions will tend to shrink as the amount of noise increases.

While for *PopularRegions*, the recall decreases smoothly with the level of noise, it decreases stepwise for our method. The reason is that our method uses a threshold to define the binary density status of the cells. It thus requires enough noise in order to flip the status of a cell. For OPTICS it can be seen that its precision and recall are lower than for the other two methods, for most of the noise levels. In the beginning, as for the ILP-based method, it still produces the same solution since it only considers the state of the cells. But unlike our method, it is not able to generalize well as it can only cover dense cells. As a consequence, its recall drops faster. For the same reason, it will never return a non-dense cell that was initially dense, thus causing a drop in recall. However, it will return dense cells that were non-dense (and thus not in the initial solution), decreasing its precision. At the light of the F1-Measure, these combined effects are in favor of our method. On any of the metrics, the ILP provides better results as long as the noise remains reasonable. When the noise level becomes significant ($\simeq 40\%$), the dense cells become very sparse and the results are much less relevant to interpret.

8 Conclusion and Future Work

Mining approaches for discovering regions of interest (ROIs) are an important building block for any application wishing to extract knowledge from location data. In order to be useful, the extracted ROIs generally need to satisfy application dependent constraints. This last requirement was missing in existing approaches. Inspired by the approach introduced in [8], we introduced an alternative approach for discovering constrained ROIs. It relies on an efficient Integer Linear Program (ILP) to extract the ROIs from a set of predefined ROIs candidates. The model can be used in a setting where the number of ROIs is fixed, or it can work in a parameter free setting by relying on the *minimum description length principle*. Our approach is flexible as it can discover ROIs satisfying various types of constraints that can be enforced either at the step of the candidate ROI generation, or directly in the integer linear programming model. We have reported various experiments showing the flexibility of the proposed approach on both real and synthetic data sets. The results have shown that it was able to retrieve constrained ROIs of higher quality than those extracted with existing approaches such as the PopularRegion algorithm [8] and clustering techniques. Despite the larger computation time, we showed that the approach is able to scale on real-world data sets using fine-grained grids.

As future work one could solve the candidate generation problem using a custom constraint-based search algorithm rather than with a generate and filter one. Although less generic, this could be more efficient if many regions need to be filtered out. The ILP does not require the shapes to be defined on the grid. As future work, it could be interesting to extend our work with ROIs defined in the continuous space. Finally, it would be interesting to extend the approach to work with continuous density values rather than binary ones that require a threshold parameter.

References

1. Agrawal, R., Srikant, R.: Mining sequential patterns. In: IEEE International Conference on Data Engineering (ICDE), vol. 95, pp. 3–14 (1995)
2. Alvares, L.O., Bogorny, V., Kuijpers, B., de Macedo, J.A.F., Moelans, B., Vaisman, A.: A model for enriching trajectories with semantic geographical information. In: GIS (2007)
3. Ankerst, M., Breunig, M.M., Kriegel, H.P., Sander, J.: OPTICS: ordering points to identify the clustering structure. ACM SIGMOD Rec. **28**(2), 49–60 (1999)
4. Aoga, J.O.R., Guns, T., Schaus, P.: An efficient algorithm for mining frequent sequence with constraint programming. In: Frasconi, P., Landwehr, N., Manco, G., Vreeken, J. (eds.) ECML PKDD 2016. LNCS (LNAI), vol. 9852, pp. 315–330. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46227-1_20
5. Belcastro, L., Marozzo, F., Talia, D., Trunfio, P.: G-ROI: automatic region-of-interest detection driven by geotagged social media data. TKDD **12**(3), 1–22 (2018)
6. Cai, G., Hio, C., Bermingham, L., Lee, K., Lee, I.: Sequential pattern mining of geo-tagged photos with an arbitrary regions-of-interest detection method. Expert Syst. Appl. **41**(7), 3514–3526 (2014)
7. Ester, M., Kriegel, H.P., Sander, J., Xu, X., et al.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: KDD (1996)
8. Giannotti, F., Nanni, M., Pinelli, F., Pedreschi, D.: Trajectory pattern mining. In: SIGKDD (2007)
9. Gorawski, M., Jureczek, P.: Regions of interest in trajectory data warehouse. In: Nguyen, N.T., Le, M.T., Świątek, J. (eds.) ACIIDS 2010. LNCS (LNAI), vol. 5990, pp. 74–81. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-12145-6_8
10. Kalra, T., Mathew, R., Pal, S.P., Pandey, V.: Maximum weighted independent sets with a budget. In: Gaur, D., Narayanaswamy, N.S. (eds.) CALDAM 2017. LNCS, vol. 10156, pp. 254–266. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-53007-9_23
11. Li, Q., Zheng, Y., Xie, X., Chen, Y., Liu, W., Ma, W.Y.: Mining user similarity based on location history. In: SIGSPATIAL. ACM (2008)
12. Monreale, A., Pinelli, F., Trasarti, R., Giannotti, F.: Wherenext: a location predictor on trajectory pattern mining. In: SIGKDD. ACM (2009)
13. Palma, A.T., Bogorny, V., Kuijpers, B., Alvares, L.O.: A clustering-based approach for discovering interesting places in trajectories. In: SAC (2008)
14. Rissanen, J.: Modeling by shortest data description. Automatica **14**(5), 465–471 (1978)
15. Ying, J.J.C., Lee, W.C., Weng, T.C., Tseng, V.S.: Semantic trajectory mining for location prediction. In: SIGSPATIAL. ACM (2011)
16. Yuan, J., Zheng, Y., Xie, X., Sun, G.: Driving with knowledge from the physical world. In: SIGKDD. ACM (2011)
17. Yuan, J., et al.: T-drive: driving directions based on taxi trajectories. In: SIGSPATIAL. ACM (2010)
18. Yuan, N.J., Zheng, Y., Zhang, L., Xie, X.: T-finder: a recommender system for finding passengers and vacant taxis. IEEE TKDE **25**(10), 2390–2403 (2013)