Find Your Way Back: Mobility Profile Mining with Constraints

Lars Kotthoff[1], Mirco Nanni[2], Riccardo Guidotti[2], and Barry O'Sullivan[3]

[1] University of British Columbia, Canada
larsko@cs.ubc.ca
[2] KDDLab – ISTI-CNR & CS Department, University of Pisa, Italy
{mirco.nanni,riccardo.guidotti}@isti.cnr.it
[3] Insight Centre for Data Analytics, University College Cork, Ireland
barry.osullivan@insight-centre.org

Abstract Mobility profile mining is a data mining task that can be formulated as clustering over movement trajectory data. The main challenge is to separate the signal from the noise, i.e. one-off trips. We show that standard data mining approaches suffer the important drawback that they cannot take the symmetry of non-noise trajectories into account. That is, if a trajectory has a symmetric equivalent that covers the same trip in the reverse direction, it should become more likely that neither of them is labelled as noise. We present a constraint model that takes this knowledge into account to produce better clusters. We show the efficacy of our approach on real-world data that was previously processed using standard data mining techniques.

1 Introduction

Clustering is one of the fundamental tasks in data mining whose general aim is to discover structure hidden in the data, and whose means consist in identifying the homogeneous groups of objects (the clusters) that emerge from the data [7]. Typically, this translates into putting together objects that are similar to each other, and keep separated as much as possible those that are different. The applications of this basic task are many and varied, ranging from customer segmentation for business intelligence to the analysis of images, time series, web search results and much more.

This paper focuses on the application domain of mobility data mining, which is concerned with the study of trajectories of moving objects and other kinds of mobility-related data [6]. Trajectories are sequences of time-stamped locations (typically longitude-latitude pairs) that describe the movements of an object. A most relevant example, which will also be the reference context for this work, is the sequence of GPS traces of cars collected by on-board devices, such as a satellite navigation system or an anti-theft platform [5].

In this context, clustering can be used to identify popular paths followed by several moving objects, such as common routes adopted by car drivers in a city. A different perspective to the problem was proposed in recent years in [8]: the

objective of the analysis is not directly the collective mobility of a population, but instead passes through an intermediate phase where the mobility of the single individual is studied first. Clustering the trajectories of an individual has the main purpose of highlighting which movements repeat consistently in his/her life, and therefore can be considered an important and representative part of the user's mobility. In the realm of urban traffic and transportation engineering such movements constitute the *systematic* component of the mobility of a person – and then of a city, as a consequence.

Identifying such *mobility profiles* is an important current research direction that has many applications, for example in urban planning [4]. If trips that individuals make frequently in their cars can be identified, the schedule and routes of the public transport system could be adjusted to provide people with an incentive to leave the car at home and do something for the environment.

In this paper, we study the problem from a constraint programming point of view. The trajectories that form part of mobility profiles have to conform to certain restrictions that are hard to incorporate in traditional data mining algorithms, but easy to express as constraints. We present a constraint programming model and demonstrate its benefits over previous approaches.

There are a number of approaches that model data mining problems with constraints (e.g. [1, 2, 9]), but to the best of our knowledge, this is the first time that this particular problem is studied in this context.

2 Drawbacks of the Pure Data Mining Approach

Existing data mining approaches to extract mobility profiles are usually customised algorithms that have been developed specifically for this purpose. The approach proposed in [8] (schematically depicted in Figure 1) consists of two ingredients: a trajectory distance function that encapsulates the context-specific notion of *similar trajectories* and a clustering schema that takes care of grouping similar trajectories together (center of Figure 1). It also ensures that the resulting groups have a certain minimum size, e.g. the cluster composed of the single dashed trajectory is removed (center of Figure 1). Once clusters have been formed, a representative trajectory for each of them is simply extracted by selecting the most central element of the cluster, i.e. the one that minimizes the sum of distances from the others (see the selected trajectories on the right of Figure 1).

We impose the minimum size constraint because we aim to identify the trips that are made regularly and are not interested in trips that occur regularly, but infrequently (such as a trip to a holiday home). The clustering procedure yields a list of trip groups, each of them essentially representing a mobility routine followed by the user. The set of routines of a user form the mobility profile of that user.

A very typical (and expected) result is that an individual's mobility contains at least two routines, one for the home-to-work systematic trips and one for the symmetric work-to-home return journey. Indeed, symmetric routines (the

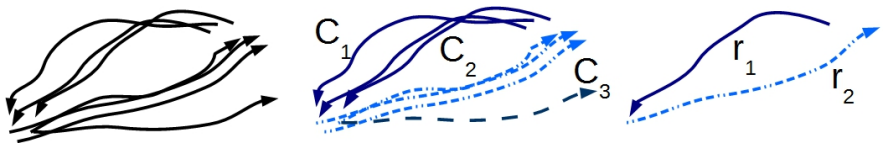

Figure 1. The three steps of profile extraction: identify the trajectories (single trips) from the GPS data, discover clusters by grouping trajectories that are close, refine the clusters to remove the noise and extract representative routines.

home-work-home cycle being the most typical example) appear very naturally in real life, and they are a very important component of an individual's mobility. Therefore, while it is simply *important* to discover all the routines hidden in an individual's mobility data, it is *crucial* to discover symmetric routines if they exist.

From the viewpoint of a traffic management application for example, symmetric routines represent simple and regular mobility cycles that might neatly fit a public transportation offer. Asymmetric routines on the other hand are generally a symptom of a more complex daily mobility, for instance involving bring-and-get activities interleaved within a home-work-home cycle, which are much more difficult to serve by a public transportation service.

These observations lead to conclude that it would be advisable to equip the routine extraction procedure with mechanisms that boost the discovery of symmetric routines. Unfortunately, the plain clustering-based approach provided so far in [8] follows a local perspective, focused on guaranteeing properties of each single cluster – namely, mutual similarities of its members and a certain minimum size of the cluster – without considering the relation between different clusters or trajectories in different clusters.

Therefore, symmetric routines, which correspond to separate clusters even though they are connected by this particular property, are treated in isolation. What we would need to stimulate the discovery of symmetric routines, is a way to link such separate clusters, and change the cluster construction criteria accordingly. The purpose of this paper is to explore this way by exploiting a constraint formulation of the clustering problem, which can then be easily enriched and customized to include the application-specific requirements we mentioned above.

Setting the Minimum Cluster Size

Being able to take the symmetry of trajectories into account provides an additional benefit when clustering. The minimum size of a cluster is currently enforced by providing a static value to the algorithm that applies to all clusters. This value needs to be chosen carefully – if it is too high, no clusters at all may be identified and if it is too low, too many clusters will be identified. In particular, the value should be set such that trajectories that are noise as far as a mobility profile is concerned are not identified as clusters.

In practice, symmetric trajectories are often part of clusters of different sizes. There are many reasons for this. Consider for example the home-to-work routine, where an individual drives straight from home to work every day, but on the way back goes to the gym on certain days and shopping on others. Therefore, the size of the home-to-work cluster will be larger than that of the symmetric work-to-home cluster.

If the threshold for the minimum size of a cluster is set higher than the size of the work-to-home cluster, it will be categorized as noise even though there is a symmetric equivalent that supports it. Indeed, there could be cases like this with symmetric clusters of different sizes where no fixed value for the minimum size threshold will ensure that all symmetric clusters are identified while the noise trajectories are not.

In a constraint model, this can be taken into account easily. Instead of considering only the size of a cluster to justify its existence, we can explicitly model symmetric clusters as supports as well.

3 Constraint Model

We present the constraint model that takes all the considerations mentioned above into account. The constraints encode when two trajectories should be in the same cluster, when they should be in different clusters, when they are noise and the conditions that apply when there are symmetric trajectories.

For the sake of simplicity, trajectories are represented by a start and an end point. For our purposes, the route taken to get from the start to the end point is secondary, as we are interested in the extraction of mobility profiles that do not contain this information. For example, the way from home to work could be different from the way from work to home because of traffic conditions, one-way streets, or similar, but both are still part of the same routine. If the user stops, e.g. to do some shopping, the trajectory is split into two parts with two separate start and end points.

The model itself is general enough to accommodate routes represented by more than two points as well, but would likely reduce the number of symmetric trajectories as pairs would look less similar because of additional (irrelevant) information. Overall, the model would become larger and more complex.

Individual trajectories $x \in X$ are modelled as variables whose domains consist of the clusters they can be assigned to. That is, each trajectory can be assigned a cluster ID, including a special value which denotes that the trajectory is noise. We use $dist(x, y)$ to denote the distance (both spatial and temporal) between two trajectories x and y. Note that for the temporal distance, only the time of day is relevant, not the date. The reason for this is that we want to cluster trips that occur regularly, e.g. the daily commute to work. The distance of two trajectories is defined as the spatial and temporal distance between their start points and their end points, so $dist(x, y) < A$ is short hand for $dist(x_{start}, y_{start}) < A \wedge dist(x_{end}, y_{end}) < A$. T denotes the distance above which two trajectories are considered to be not *close* and R the distance above

which they are considered *far*, $T < R$. If a trajectory is noise and should not be part of any cluster, it is denoted $x = noise$.

Our formulation consists of two complementary parts. The first part is a set of constraints that specify under which conditions two trajectories should belong to different clusters or be labelled as noise. The second one is a multi-objective optimization function that requires to cluster as many trajectories as possible and to fit them into as few clusters as possible. This requires the identified clusters to be as large as possible.

Our optimization function can be formulated as follows, where we use $\|X\|$ to denote the cardinality of the set X:

$$\begin{array}{ll} \text{minimize } \|unique(\{x \mid x \in X\})\| & \text{(no. distinct clusters)} \\ \text{minimize } \|\{x \mid x \in X, x = noise\}\| & \text{(no. noise trajectories)} \end{array} \tag{1}$$

Our constraint model includes three kinds of constraints. First we specify when two trajectories should *not* be in the same cluster:

$$\forall\, x, y \in X : dist(x, y) > R \rightarrow x \neq y \vee x = noise \vee y = noise \tag{2}$$

If two trajectories are far apart, they should be in different clusters or at least one of them should be noise. In our application, we are trying to identify mobility patterns which are defined by clusters of trajectories that are all close together. We therefore want to avoid clustering trajectories that are close to a set of intermediate trajectories, but far apart themselves.

Symmetric trajectories x and y are denoted as $symm(x, y)$ and defined as follows:

$$\begin{aligned} \forall x, y \in X : symm(x, y) \equiv\ & dist(x_{start}, y_{end}) \leq T \wedge \\ & dist(x_{end}, y_{start}) \leq T \end{aligned} \tag{3}$$

We now introduce the symmetry constraints that represent the main advantage of the constraint model over existing data mining algorithms. We denote the threshold for the minimum number of trajectories in a cluster S. This is a fixed value specified by the user and potentially prevents identifying clusters that are small, but supported by a symmetric cluster. We therefore introduce the threshold size for a pair of symmetric clusters S', where $S' > S$. If a trajectory x is symmetric to another trajectory y, they should belong to different clusters if they are not noise and the sum of the sizes of the clusters they belong to must be greater than S'.

$$\begin{aligned} \forall\, x, y \in X : & symm(x, y) \wedge x \neq noise \wedge y \neq noise \\ & \wedge (\|\{z \mid z \in X, z = x\}\| + \|\{z \mid z \in X, z = y\}\| > S') \rightarrow x \neq y \end{aligned} \tag{4}$$

This set of constraints encodes most of the background knowledge on symmetric trajectories. We both take symmetry into account explicitly and mitigate the

drawbacks of having a fixed minimum cluster size. The fixed threshold S is the only means of separating signal from noise in the data mining application. In the constraint model, we can relax this requirement in the presence of symmetric clusters – if a symmetric pattern is identified, we require the sum of the cluster sizes to be greater than the threshold for a pair of symmetric clusters, which is greater than the threshold for a single cluster.

If, on the other hand, there is no trajectory that is symmetric to x, then x is either noise or the size of the cluster that it is assigned to is greater than the threshold S for the size of a single cluster.

$$\begin{aligned} \forall x \in X : \|\{y \mid y \in X, symm(x,y)\}\| = 0 \rightarrow \\ x = noise \vee (x \neq noise \wedge \|\{z \mid z \in X, z = x\}\| > S) \end{aligned} \tag{5}$$

These constraints describe the requirements we place on trajectories to be part of clusters as well as considering the relation between clusters by taking the symmetry into account. We do not rely on any special constraints or esoteric constructs – our model is generic and can be implemented and solved with almost any constraint solver. In contrast, the data mining algorithm requires a specialised implementation that, while being able to leverage existing work, needs considerable efforts.

4 Model Implementation

We implemented the constraint model described above in the Minion constraint solver [3]. Minion is a general-purpose constraint solver. Our choice of solver was motivated by nothing but the authors' familiarity with this particular solver and its free availability.

4.1 Minion Model

Our Minion model is an almost direct translation of the model presented in the previous section. For each trajectory, we add one variable whose domain values represent the clusters the trajectory can be assigned to. We also add the dual representation where each cluster is represented by an array of Booleans, one for each trajectory. If a particular trajectory is in a particular cluster, the corresponding Boolean value is set to 1, else 0. We also add the channelling constraints between the one-variable-per-trajectory and the one-array-per-cluster representations. We require the dual representation to be able to post constraints on cluster sizes.

The noise trajectories are assigned to a special cluster that is represented by the value 0 in the domains of the trajectory variables. Modelling noise this way allows us to treat it in the same way as other clusters.

The Minion input language is flat and does not support quantifications over the variables. We therefore instantiate the free variables in Equations 2 to 5 and add constraints for each grounding. Note that the left hand side (before the

implication) in all of these equations except 4 can be computed statically before solving the problem as it only depends on the trajectory data. We add constraints corresponding to the right hand side of the implications to the Minion model only if the left hand side is true. This preprocessing step helps to reduce the size of the Minion model, which is potentially very large because of the nested quantifications.

The implication on the right hand side of Equation 4 is modelled with the help of auxiliary variables that represent the left hand side of the implication being true. We need auxiliary variables here because Minion does not support implications between constraints, but only between a constraint and a variable. All variables and constraints are treated the same way by Minion and arc consistency is enforced during search. We use Minion's default parameters.

The thresholds T, R, S and S' are specified by the user. In the experiments section we will see the effect of these parameters on the results.

4.2 Optimising the Clustering

The optimisation objective of mobility profile mining is to cluster as many trajectories as possible, subject to the constraints, while minimising the overall number of clusters. We cannot easily express this multi-objective optimisation in our Minion model, but it turns out that we do not have to. We instead solve the simpler satisfaction problem that is defined as follows. Given a number of clusters, we assign values to trajectory variables in a descending manner. That is, we start with the highest cluster and assign a trajectory to the noise cluster (value 0) only if all other possibilities have been exhausted. This ensures that as many trajectories as possible are clustered as non-noise. In Minion, we simply specify a descending value ordering for the search.

We now minimise the number of clusters as follows. We start by specifying only two clusters in addition to the noise cluster. If the constraint satisfaction problem (CSP) has a solution, we have found our clustering. If there are more symmetric trajectories that need to be in different non-noise clusters however, this first CSP will have no solution. In this case, we increase the number of clusters by one, generate a new CSP and try to solve it. This way, we identify the solution with minimum number of clusters.

In practice, the number of clusters is usually small. The number of routine trips an individual makes that we have observed empirically was fewer than five in most cases, such that our method is likely to find the solution after only a few iterations.

4.3 Scalability

The main reason why data mining and machine learning applications use specialised approximate algorithms instead of complete optimisation approaches such as constraint programming is the scalability of such methods. Approximate methods are almost always orders of magnitude faster than corresponding complete methods while delivering solutions of similar quality.

While scalability is a concern for our application as well, it is not an issue in practice, due to the high modularity of the problem: indeed, each user can be analyzed separately, making the complexity of the problem linear in the number of users. The critical factor is, instead, the number of trajectories of a single user. However, the vast majority of individuals have a relatively low number of trips (in the dataset used for our experiments, which covers a time span of 5 weeks, most users have fewer than 100 trajectories), and therefore the constraint models can be solved in reasonable time. Our application is not time-critical, but puts the main emphasis on the quality of the solutions. We are happy to spend more time solving the problem as long as we get a better quality result.

The main factor that contributes to the size of the CSP apart from the number of trajectories is the number of clusters. Each additional cluster increases the size of the domains of the trajectory variables, requires an additional array of Boolean values and additional constraints to account for the possibility of a trajectory being assigned to that cluster. The iterative approach for setting the number of clusters described above keeps the size of the CSP as small as possible while not compromising on the quality of the solution.

We have found that this approach has a very significant impact on the efficiency with which we can solve this clustering problem. Often, increasing the number of clusters beyond the optimal number increases the solve time of the model by orders of magnitude.

5 Experimental Evaluation

We evaluated the quality of the clusters that the constraint model finds on real-world trajectories collected in Italy. We will show how our constraint model improves on the existing solution for extracting mobility profiles, thus evaluating the impact of taking symmetric clusters into account on the results. We briefly describe the dataset used in the experiments, then summarize the results, and finally show an example of profiles found on a single user with the different approaches.

5.1 Dataset

Our dataset contains the trajectories of the cars of 150 users living around the city of Pisa, Italy, collected over a period of 5 weeks (Figure 2). We cannot make the actual data available because of privacy reasons; an anonymized version is available at `http://www.cs.ubc.ca/~larsko/downloads/cp2015_anonymized_dataset.tar.gz`.

The users were selected in such a way to have a good representation of different mobility complexities, i.e. different numbers of trajectories. In our stratified sample, the number of trajectories per user roughly follows a uniform distribution that ranges from 1 to 100 (see Figure 3).

Experiments were carried out adopting a spatio-temporal distance $dist(x, y)$ between two points that is computed as the standard Euclidean distance between

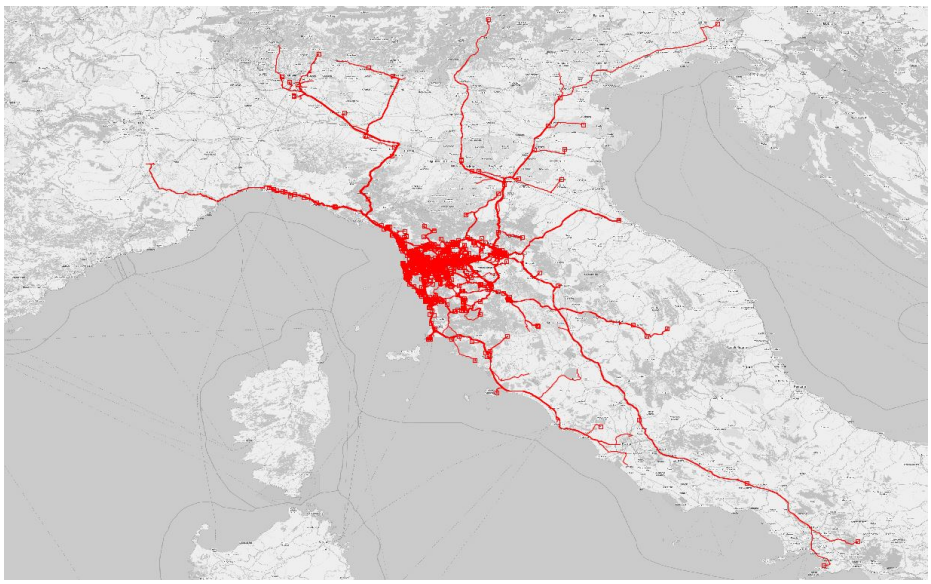

Figure 2. Dataset of trajectories used in the experiments, containing 150 users centered around Pisa, Italy.

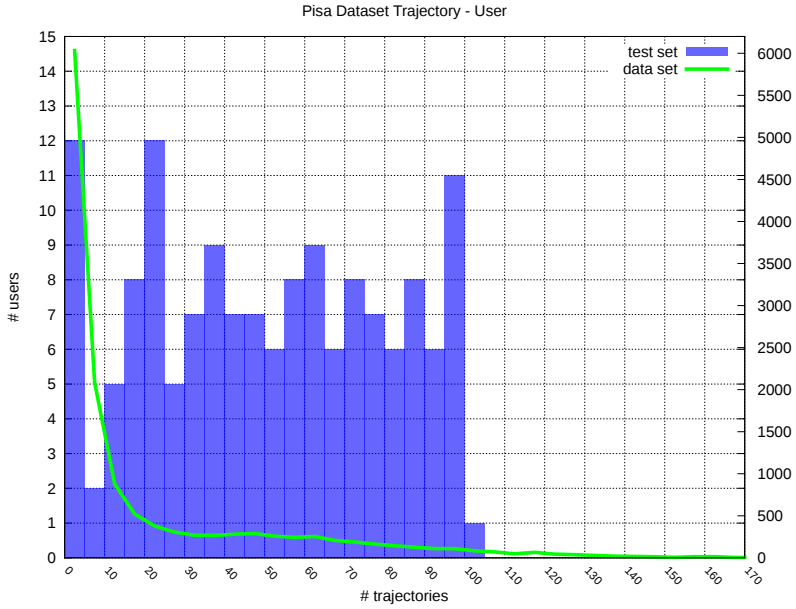

Figure 3. Distribution of number of trajectories per user used in the experiments (blue bars) vs. distribution of original dataset (green line).

the spatial components of x and y if their temporal gap is less than a threshold $\tau = 1hr$, and ∞ otherwise. Therefore, we used the following ranges of values for parameters S (the minimum number of trajectories in a cluster) and T (the distance threshold between trajectories, expressed in meters): $S \in \{3, 4, 5, 6\}$, $T \in \{100, 200, 300, 400, 500\}$. Moreover, the parameters S' (the threshold for the combined size of two symmetric clusters) and R (the maximum distance between trajectories in a cluster) are set as functions of S and T, respectively: $S' = 1.5 \cdot S$, $R = 3 \cdot T$. The coefficients used to compute S' and R have been chosen empirically through a set of preliminary experiments over the dataset. Different datasets may exhibit different properties and a recalibration of these values may be required.

Computation over a medium-low-end desktop machine (i3 CPU at 3GHz, with 4GB RAM, running Ubuntu 12) took an overall time below 15 minutes. In particular, single users with 20 or less trajectories (which are the most common in the original dataset) are dealt with in less than one second, while those with 50 to 100 trajectories (less common, yet over-represented in our sample set, to evaluate their impact) took up to 15 seconds each.

5.2 Evaluation of Results

The main advantage of our constraint model over previous approaches is that it allows to take symmetric clusters into account. Without the constraints that do this, we should achieve results that are equivalent to the data mining solution proposed in [8]. However, the constraint model should find an optimal solution instead of a heuristic one.

In our first set of experiments, we compare both approaches without taking symmetric clusters into account to get a baseline for the performance of our new approach. This also allows to evaluate the impact of taking the symmetric clusters into account later.

Figures 4 and 5 show a comparison of the two approaches for the different parameter values (S and T) in terms of the number of non-noise trajectories (Figure 4) and of number of clusters found (Figure 5).

Both the number of non-noise trajectories and clusters increases as we relax the constraints, i.e. allow fewer trajectories that are further apart in a cluster. This is true for both the data mining and the CP approach. In general, both the number of non-noise trajectories and clusters is larger for the CP approach, in some cases significantly so. For example, the number of clusters for $S = 3$ that the CP approach finds is almost twice the number the data mining approach finds. As S is increased, the difference becomes smaller. The reason for this phenomenon is that the data mining approach partitions the set of trajectories into clusters through a heuristics that considers the distribution of data only locally, and therefore can create small clusters that are later discarded, as shown in the example of Figure 1. This kind of situations are avoided by the CP approach, which instead considers the assignment of trajectories to clusters from a global viewpoint.

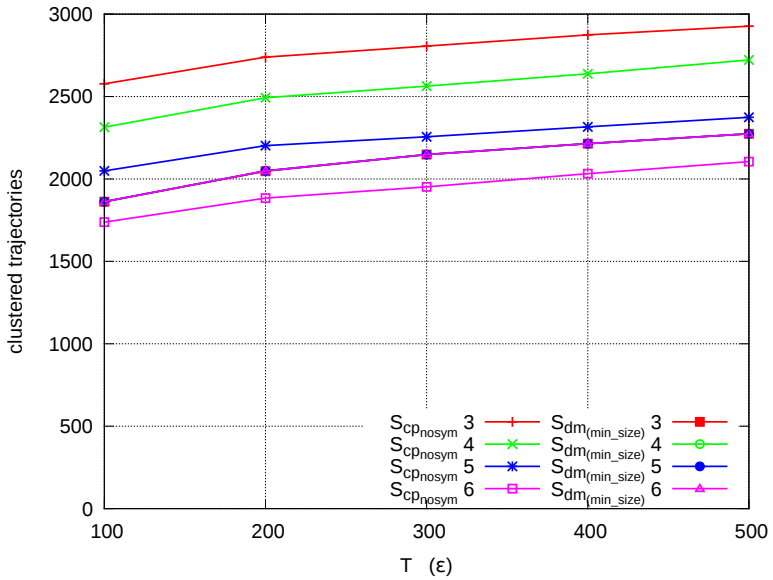

Figure 4. Number of clustered trajectories found by the data mining approach (dm) and the constraint model without symmetry constraints (CP_{nosym}) for different distance threshold and cluster size values T and S.

In our second set of experiments, we focus on evaluating the impact of taking symmetric clusters into account. We adopt two measures for comparing the complete constraint model with all constraints to the one that does not take symmetric clusters into account as above.

- the *trajectory coverage* of method A over method B, defined as:

$$tr\_cov(A,B) = \frac{\|\{x \mid x \in X_A, x \neq noise\} \cap \{x \mid x \in X_B, x \neq noise\}\|}{\|\{x \mid x \in X_B, x \neq noise\}\|} \quad (6)$$

  where X_A (resp. X_B) represents the overall set of clustered trajectories obtained with method A (resp. B). When A completely covers B, $tr\_cov(A,B) = 1$.
- the *cluster coverage* of method A over method B, defined as:

$$cl\_cov(A,B) = \frac{\sum_{u \in U} (C_u \cdot cl\_cov_u(A,B))}{\sum_{u \in U} C_u} \quad (7)$$

  where U is the set of users and C_u the number of clusters found by A for user u. Function $cl\_cov_u(A,B)$, in turn, is computed over each user u as the average trajectory coverage of each cluster of u found by A w.r.t. those found by B. The matching between clusters is performed by taking the best match, i.e. maximizing the *trajectory coverage* limited to the two clusters compared.

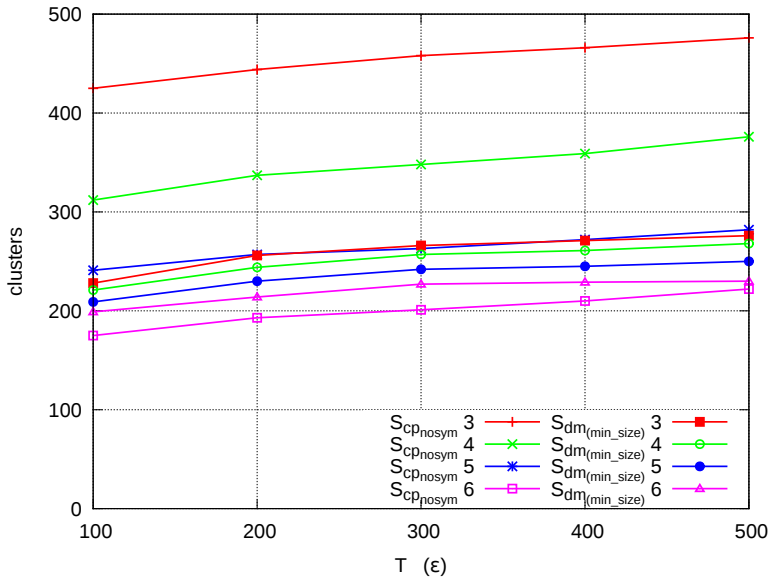

Figure 5. Number of clusters found by the data mining approach (dm) and the constraint model without symmetry constraints (CP_{nosym}) for different distance threshold and cluster size values T and S.

Clusters left unpaired are assigned to an empty set. When A completely covers B, $cl\_cov(A, B) = 1$.

While the trajectory coverage compares the overall ability to recognize systematic trips, the cluster coverage looks at how much each single cluster is discovered by the other method. Figures 6 and 7 report the results obtained on our dataset. Lines labelled with $S_{CP_{sym}}$ represent the coverages of the model with symmetry constraint (CP_{sym}) over that without symmetry (CP_{nosym}), and $S_{CP_{nosym}}$ the opposite.

The plots clearly show that the model that takes symmetric trajectories into account completely covers the other one, both in global terms of trajectory coverage (Figure 6) and in terms of (the more precise) cluster coverage (Figure 7), since the corresponding values are always very close to 1. The model that does not take symmetric trajectories into account on the other hand loses at least 50% of trajectories and 70% of cluster coverage. These results are consistent across the range of values for S and T that we consider here, with little variation.

Our results show that considering the symmetry of clusters has a large impact on results. We are able to cluster a lot more trajectories that were mistakenly identified as noise before, allowing to recover several *backward* trips that would otherwise not be recognized as routines.

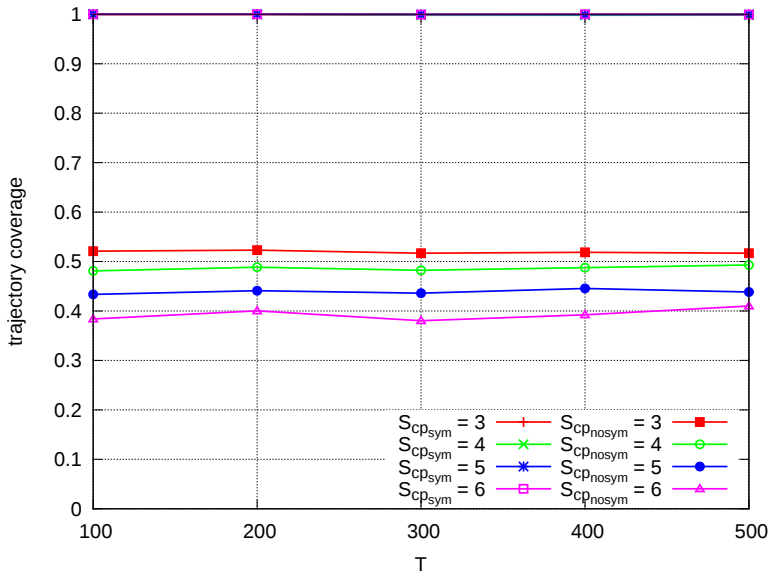

Figure 6. Trajectory coverage for constraint models with symmetry constraint (CP_{sym}) and without (CP_{sym}) for different distance threshold and cluster size values T and S.

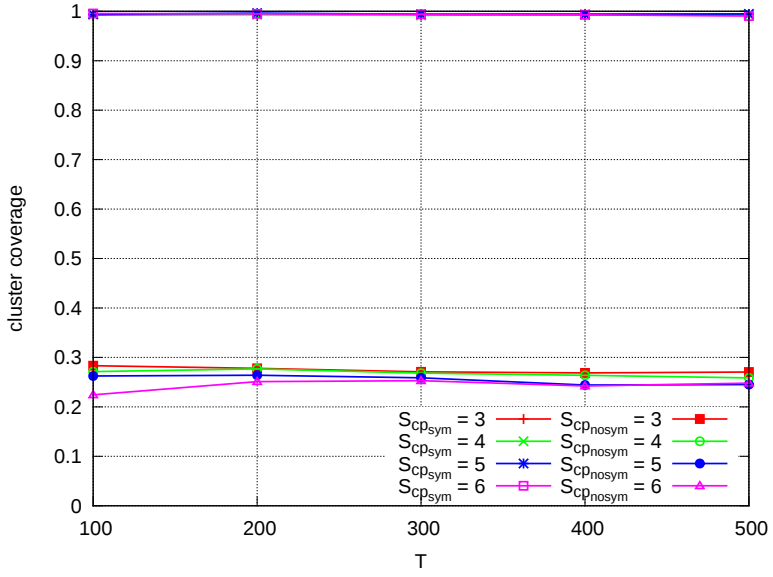

Figure 7. Cluster coverage for constraint models with symmetry constraint (CP_{sym}) and without (CP_{sym}) for different distance threshold and cluster size values T and S.

5.3 Qualitative Evaluation

The quality of the clustering is more difficult to assess, as there are no automated means to determine whether a set of clusters "makes sense". In practice, clusterings are evaluated through assessment by a human expert. We obviously cannot do this for all of the users we consider in this paper and consider the clusters for a single user as an example here.

While not all users have symmetric trajectories, the following example demonstrates the motivation of our approach and the case that is difficult to solve for specialised data mining methods. A clustering example that shows the benefit of the constraint model is presented in Figure 8. The image was obtained by using $T = 100$ and $S = 6$.

The red and the blue trajectories represent symmetric parts of the same routine. However, there are fewer red than blue trajectories; in particular the number of red trajectories is smaller than the threshold S. If the symmetry is not taken into account, only the blue trajectories are clustered, whereas the red ones are discarded as noise. The CP approach that does consider symmetry identifies the red trajectories correctly as a non-noise cluster.

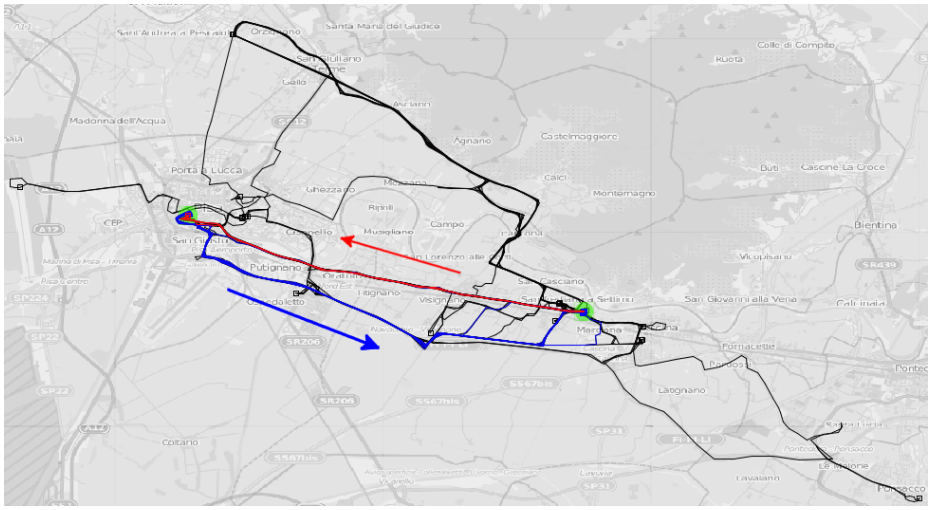

Figure 8. Trajectory clusters found on a sample user (best seen in colors).

For the eastward journey, our user mainly follows the blue routine, but for the return journey he/she often changes path, either for traffic reasons or for personal needs, and the red trajectories represent the most frequent choice. The figure clearly shows the benefit of being able to take symmetric trajectories into account. Indeed, there are only three trajectories in the red cluster, which has been discarded as noise by the other method.

Lowering S to 3 would have clustered the red trajectories as non-noise, but at the same time erroneously identified a number of additional clusters that are in fact noise for our purposes. Therefore, the model that does not consider symmetric trajectories (as well as the specialised data mining algorithm discussed above) could not have identified the same set of clusters for this user.

6 Conclusion and Future Work

We have presented a constraint programming formulation of the problem of mining mobility profiles from GPS trajectories. This is an important research direction with many real-world applications that, to date, has been the domain of specialised data mining algorithms. We have described the drawbacks of these specialised algorithms and how they are alleviated with constraint programming.

The implementation of our model and its experimental evaluation on real-world data showed that the model stands up to its promise and delivers clusterings of a higher quality than those that the previously-used specialised data mining algorithms achieve. In particular, we are able to identify a much larger number of non-noise trajectories and clusters. The experiments further demonstrate that it is feasible to use complete optimisation methods for this problem.

The main benefit of constraint programming over similar technologies for this application is the succinct representation of the model. In particular SAT models would be much larger to accommodate the potentially large domains encoding the possible cluster assignments. Similarly, ILP formulations would be more complex because of nested constructs.

There are several avenues for future work. We did not add any symmetry-breaking constraints to our model even though symmetries occur – for example in the numbering of the clusters. Formulating and adding symmetry-breaking constraints may increase the scalability of our approach further.

On the data mining side, there are additional problem characteristics that are not taken into account in our current model. The day that particular trips were taken for example is not considered, but it would be beneficial to do so. We could for instance boost symmetric trajectories that occur on the same day, as they are more likely to represent a recurring activity.

There are many additional extensions to the model presented in this paper. Even our relatively basic formulation shows the promise of the approach by delivering results of a higher quality than specialised approaches while being easier to maintain and modify.

Acknowledgements

This work was supported by the European Commission as part of the "ICON - Inductive Constraint Programming" project (contract number FP7-284715). The Insight Centre for Data Analytics is supported by Science Foundation Ireland under grant number SFI/12/RC/2289. We thank the anonymous reviewers for their helpful feedback.

References

1. Davidson, I., Ravi, S.S., Shamis, L.: A SAT-based Framework for Efficient Constrained Clustering. In: SIAM International Conference on Data Mining. pp. 94–105 (2010)
2. De Raedt, L., Guns, T., Nijssen, S.: Constraint programming for data mining and machine learning. In: AAAI. pp. 1671–1675 (2010)
3. Gent, I.P., Jefferson, C.A., Miguel, I.: MINION: a fast, scalable, constraint solver. In: ECAI. pp. 98–102. IOS Press (2006)
4. Giannotti, F., Nanni, M., Pedreschi, D., Pinelli, F.: Trajectory pattern analysis for urban traffic. In: Proceedings of the Second International Workshop on Computational Transportation Science, IWCTS 2009. pp. 43–47 (2009)
5. Giannotti, F., Nanni, M., Pedreschi, D., Pinelli, F., Renso, C., Rinzivillo, S., Trasarti, R.: Unveiling the complexity of human mobility by querying and mining massive trajectory data. The VLDB Journal 20(5), 695–719 (2011), `http://dx.doi.org/10.1007/s00778-011-0244-8`
6. Giannotti, F., Pedreschi, D. (eds.): Mobility, Data Mining and Privacy - Geographic Knowledge Discovery. Springer (2008)
7. Tan, P.N., Steinbach, M., Kumar, V.: Introduction to Data Mining. Addison-Wesley (2005)
8. Trasarti, R., Pinelli, F., Nanni, M., Giannotti, F.: Mining mobility user profiles for car pooling. In: KDD. pp. 1190–1198 (2011)
9. Wagstaff, K., Cardie, C.: Clustering with Instance-level Constraints. In: Seventeenth International Conference on Machine Learning. pp. 1103–1110. ICML '00, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2000)