ERMO-DG: Evolving Region Moving Object Dataset Generator

Berkay Aydin and **Rafal A. Angryk**
Georgia State University
Department of Computer Science
Atlanta, GA 30302-3994 USA
{baydin2, angryk}@cs.gsu.edu

Karthik Ganesan Pillai
Montana State University
Department of Computer Science
Bozeman, MT 59717 USA
k.ganesanpillai@cs.montana.edu

Abstract

It is often essential to create datasets with foreseeable characteristics. For the design and testing of advanced spatiotemporal pattern mining algorithms, adaptable and large datasets are needed. In this paper, we present a synthetic dataset generator, ERMO-DG, that is intended for creating spatiotemporal patterns. Generated datasets consist of spatiotemporal object instances of different feature types, where these instances are represented by spatial regions evolving over time. The generator allows researchers to systematically create spatiotemporal datasets with predictable characteristics such as number of patterns, cardinality of patterns, velocity, acceleration, lifetime and spatial areas of instances.

Introduction

As the volume of spatiotemporal data continues to grow unceasingly, researchers from academia and industry propose new algorithms or produce new tools to store, index, and mine spatiotemporal data. In recent years, there have been many spatial and spatiotemporal pattern mining algorithms proposed in (Cao et al. 2006), (Celik et al. 2006), (Wang et al. 2005), (Yang et al. 2005), and (Pillai et al. 2013) The motivation for these algorithms comes from different scientific domains such as astronomy, geology, meteorology etc. These algorithms can be utilized for the verification of a current scientific theory, the prediction of interesting relationships between different phenomena, or the discovery of new relationships. Real life spatiotemporal datasets have been used in order to test the correctness, completeness, and compare the efficiency of these algorithms. Some of these datasets include climate data, solar data and volcanic eruption data (Pillai et al. 2012), (Shekhar and Huang 2001), (Schuh et al. 2013). On the other hand, synthetic data generation is also useful, because synthetic data simplifies the experimental repeatability with well-defined data properties, which points out the the strengths and weaknesses of algorithms as researchers can test their algorithms with specific boundary conditions. Interestingly, quite a few spatiotemporal pattern mining algorithms were extensively tested using synthetic data along with real life data (Celik et al. 2006), (Cao et al. 2006), (Pillai et al. 2013).

In this paper, we present a new, highly parameterized spatiotemporal dataset generator, ERMO-DG, which can be utilized for testing the correctness, completeness, and comparing the efficiency and scalability of spatiotemporal pattern mining algorithms. Our motivation for creating ERMO-DG was to be able to generate consistent, pattern-generating datasets with foreseeable characteristics. ERMO-DG has been successfully used by Pillai et al. for testing the correctness and completeness of spatiotemporal co-occurence pattern mining algorithms (2013).

The instances created by ERMO-DG are moving spatiotemporal objects with evolving region-based representations. ERMO-DG does not generate point-based or network-based instances. Additionally, we present the instances with complete histories; however, the current or near future movements of spatiotemporal objects are not in the scope of this paper. To generate instances, ERMO-DG initially creates feature types and patterns. Then, it generates instances which will be grouped in randomly generated spatial neighborhoods in order to realize the generation of spatiotemporal co-occurence patterns.

The rest of this paper is organized as follows. In Related Work, we present early spatiotemporal dataset generators and demonstrate examples of synthetic data utilization in spatiotemporal pattern mining. Next, detailed information on the dataset generation process is given in Generation of Dataset. Lastly, we conclude the paper and discuss the future work issues in Conclusion and Future Work.

Related Work

Many topics that the spatiotemporal research community has been exploring, such as spatiotemporal pattern mining recently, access methods, query languages and indexing, most require either real or synthetic datasets for comparing the efficiency and analyzing the performance of algorithms. Many different approaches on the generation of synthetic spatiotemporal data have been presented in the last fifteen years. Oporto (Saglio and Moreira 2001) is one of the first well-known spatiotemporal dataset generators. Data from Oporto mimics the movement of fishing boats by following the principle that “real life instances are not chaotic”. GSTD algorithm is presented by Theodoridis et al. Theodoridis, Silva, and Nascimento in (1999). The generated data in GSTD is transaction-time oriented and memoryless; furthermore, the movement of data is controlled by duration, shift and resizing parameters. Tzouramanis et al. (2002) proposed a somewhat different dataset generator in , G-TERD (Generator for Time Evolving Regional Data). It is a highly parametrized generator where users can choose many characteristics such as rotational movement, enlargement (or reduction), and whether objects can go over other objects or not. Brinkhoff (2002) presented a network-based approach on the generation of spatiotemporal datasets with nine statements that are derived from real life principles. Düntgen et al. demonstrated a benchmark for moving object databases, called BerlinMOD (2009). Their benchmark is based on a simulation scenario, where they simulate a number of cars driving on the road network of Berlin; and it creates network-based point dataset. In addition, Chen et al. proposed a benchmark targeting the current and near-future indexing of moving spatiotemporal objects (2008). These dataset generators allow researchers to compare spatiotemporal access methods, query languages, and indexing techniques; however, they do not fulfill the needs of spatiotemporal pattern mining research since they do not offer any heuristics or spatial neighborhoods of instances, which can eventually lead to the generation of spatiotemporal co-occurence patterns. Very recently, Hermoupolis, which is a mobility pattern-aware trajectory generator, is introduced (Pelekis et al. 2013). It generates synthetic trajectories of moving objects which will result in mobility patterns such as flocks, convoys, clusters etc.

None of the above approaches present a stand-alone, highly parametrized, spatiotemporal dataset generator that can be used for benchmarking spatiotemporal pattern mining algorithms.

Generation of Dataset

Spatial co-location of objects occurs when moving objects have the same or close positions in common (Andrienko and Andrienko 2007). Shekhar and Huang (2001), described the spatial co-location patterns as the relationships among different types of instances occurring in different and possibly nearby (common) locations. On the other hand, spatiotemporal co-occurence can be seen as the temporal extension of spatial co-location, where instances of different types of objects occur both in space and time. ERMO-DG is intended for creating synthetic datasets that have certain characteristics which lead to the creation of spatiotemporal co-occurence patterns. Spatiotemporal co-occurence patterns represent the subsets of feature types that occur both in space and time (Pillai et al. 2013). Therefore, the crucial part of the data generation process is creating the spatial and temporal co-occurences of instances.

The spatiotemporal *patterns* are designed as a subset of different feature types. Feature types being in a pattern, signifies that the instances associated with these features occur together in both space and time. A *feature type* describes the spatial and temporal characteristics of associated instances. It can be considered as a set of meta-attributes delineating the attributes of an instance. *Instances* in ERMO-DG are spatiotemporal objects that change their location and shape over time. Each instance has a lifetime and a sequence of region polygons that shows the location of instances at each timestamp. All instances are associated with a specific feature type.

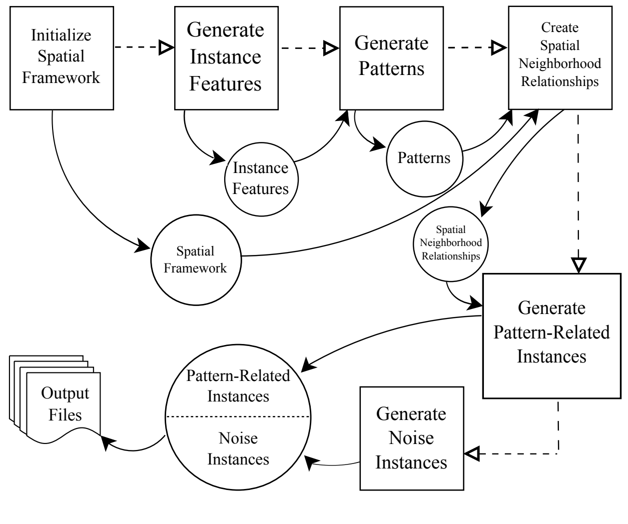
Initialize Spatial Framework
Generate Instance Features
Generate Patterns
Create Spatial Neighborhood Relationships
Instance Features
Patterns
Spatial Framework
Spatial Neighborhood Relationships
Generate Pattern-Related Instances
Output Files
Pattern-Related Instances
Noise Instances
Generate Noise Instances

Figure 1: Generation of instances

The overall generation process is outlined in Fig. 1. The rectangular shapes signifies the procedures and round shapes show the inputs and outputs of these procedures. Note that dashed arrows demonstrates order of the procedures and straight ones shows the flow of input and output among the procedures. The data generation process starts with randomly creating the feature types. Then, patterns are formed by arbitrarily grouping the features. It is followed by generating the instances for each pattern. Definitions of key concepts and the detailed explanation of each step is discussed in the following subsections.

Definition 1: *Spatial Framework*
Spatial framework is a rectangle whose upper-left corner is located at $(0,0)$ and lower-right corner is located at point (D_x, D_y). Instances are created within the spatial framework. A real life example for a spatial framework is the 2-dimensional map of a geographical area or the Sun. Note that the spatial framework size does not change; however, the contents (instances) of spatial framework is dynamic and their locations change over time.

Definition 2: *Feature Type*
A feature type is a list of attributes that will be utilized while creating the instances. It comprises shape, area, lifetime, velocity, and acceleration attributes. An individual feature type is denoted by F_i=$\{Min_{Vertex}$, Max_{Vertex}, Min_{Area}, Max_{Area}, $Min_{AreaEvol}$, $Max_{AreaEvol}$, $Min_{Lifetime}$, $Max_{Lifetime}$, $Min_{Velocity}$, $Max_{Velocity}$, $Min_{Acceleration}$, $Max_{Acceleration}\}$.

The explanations for these attributes are as follows: Min_{Vertex} - Max_{Vertex} specify the number of minimum and maximum vertices an instance polygon can have respectively. Min_{Area} and Max_{Area} indicate the minimum and maximum area of an instance polygon can have (respectively) through its lifetime. $Min_{AreaEvol}$ and $Max_{AreaEvol}$ indicate the minimum and maximum factor of increase or decrease in the polygon area of an instance during the its lifespan. $Min_{Lifetime}$ and $Max_{Lifetime}$ indicate the minimum and maximum lifespan of an instance respectively. $Min_{Velocity}$ and $Max_{Velocity}$ shows the minimum and maximum velocity of an instance respectively. $Min_{Acceleration}$ and $Max_{Acceleration}$ indicate the minimum acceleration of an instance respectively. A real life example of feature types can be the different classes of clouds in the sky. For instance, low level clouds such as cumulus and stratus, and high level clouds such as cirrus can be represented as different feature types, each characterized diversely with different minimum and maximum attribute values for velocity, area, lifetime etc.

Definition 3: *Instance*
A spatiotemporal instance is an ordered list of polygons denoted by $< polygon_i, t_i >$, each representing the spatial representation of the instance in distinct, consecutive timestamps. Each instance is associated with a feature type. The attributes specified in a feature type are used to generate the polygons. An instance is denoted by $Inst^j$, where j shows the associated feature type. A real life example of spatiotemporal instance would be solar flare, which erupts, moves, and evolves on the corona of the Sun.

Definition 4: *Pattern*
A pattern is a subset of distinct feature types. The cardinality (number of feature types involved in the pattern) of a pattern cannot be less than 2, nor can the cardinality be more than the total number of feature types. An existence of a pattern for a subset of feature types signifies that the instances associated with those features occur together in both space and time. An example of a pattern in real life would be the coexistence of sigmoids and active regions in nearby locations on the surface of the Sun.

We divide the patterns into *core* patterns and *overlap* patterns. Detailed explanation for core and overlap patterns is provided in the subsection, "Generation of Patterns".

Definition 5: *Spatial Neighborhoods*
Spatial neighborhoods are used for utilizing the spatial cooccurence of instances associated with a pattern. The instances associated with a pattern (intrinsically with a feature type in that pattern), are created very close to each other (spatially) at their birth time. Spatial neighborhoods represent the common (or close-by) locations where the instances of a pattern co-occur. We also align the birth time of instances in order to realize the temporal aspect of the cooccurence. A spatial neighborhood list is created for each pattern and they are denoted as $< SpN >$. A real life example for spatial neighborhoods is the location of an active volcano. The spatial neighborhood can be represented as the location of an erupting volcano and resulting eruption columns and ash clouds can be the co-occurring instances in that spatial neighborhood.

Detailed description of the dataset generation algorithm is shown in *Algorithm 1*, and the parameters related to each part of algorithm are described in the next subsections. In addition to those, the dimensions (D_1, D_2) of the spatial framework are also given as a parameter. Since the spatial framework initialization is trivial, we skip this part.

Algorithm 1: Dataset Generation Algorithm

```
Input  : $D_1, D_2, F_{Count}, Base_{Area}$,
         $GlobalMax_{Vertex}, Base_{AreaEvol}$,
         $Base_{Lifetime}, Base_{Velocity}$,
         $Base_{Acceleration}, N_{Core}, S_{Overlap}, S_{Inst}$,
         $M_{SpN}, L_{Noise}$
Variables: $SpF$: Spatial framework
           $F_{All}$: The set of all feature types
           $P_{Core}$: The set of core patterns
           $P_{Overlap}$: The set of overlap patterns
           $< SpN >$: Spatial neighborhood list
Output: Set of spatiotemporal instances with the list of
        <polygon, timestamp> pairs
Algorithm:
  $SpF \leftarrow$ InitializeSpatialFramework($D_1$, $D_2$)
  $F_{All} \leftarrow$ GenerateFeatureTypes($F_{Count}$, $Base_{Area}$,
        $GlobalMax_{Vertex}$, $Base_{AreaEvol}$,
        $Base_{Lifetime}$, $Base_{Velocity}$, $Base_{Acceleration}$)
  $P_{Core} \leftarrow$ GenerateCorePatterns($F_{All}$, $N_{Core}$)
  $P_{Overlap} \leftarrow$ GenerateOverlapPatterns($F_{All}$,
        $S_{Overlap}$, $P_{Core}$)
  foreach $P_i \in (P_{core} \cup P_{Overlap})$ do
    $< SpN > \leftarrow$ GenerateSpatialNeighborhood($SpF$,
          $S_{Inst}$, $M_{SpN}$)
    GenerateInstances($P_i$, $S_{Inst}$, $< SpN >$)
    GenerateNoiseInstances($P_i$, $S_{Inst}$, $L_{Noise}$)
  endfor
```

Generation of Feature Types

Parameters

F_{Count}: The total number of feature types to be created.

$Base_{Area}$: The base area parameter used for characterizing the area attributes of feature types.

$GlobalMax_{Vertex}$: The maximum number of vertices that a polygon can have.

$Base_{AreaEvol}$: The base areal evolution parameter used for characterizing the areal evolution attributes of features.

$Base_{Lifetime}$: The base lifetime parameter used for characterizing the lifetime attributes of features.

$Base_{Velocity}$: The base velocity parameter used for characterizing the velocity attributes of features.

$Base_{Acceleration}$: The base acceleration parameter used for characterizing the acceleration attributes of features.

The attributes of feature types in ERMO-DG are generated randomly. We use F_{Count} to determine the number of feature types to be created. After that, we use base parameters and global maximum vertex count to find maximum and minimum values described in Definition 2. The minimum number of vertices that a polygon can have is 3 by definition. We use $GlobalMax_{Vertex}$ parameter to decide the number of maximum vertices that a polygon can have. We generate two random integers between 3 and $GlobalMax_{Vertex}$ and we assign those two random values to Max_{Vertex} and Min_{Vertex} attributes of a feature. For area, lifetime, velocity, and acceleration attributes, minimum and maximum attribute values (Min_{Area}, $Min_{Lifetime}$, $Min_{Velocity}$, $Min_{Acceleration}$, Max_{Area}, $Max_{Lifetime}$, $Max_{Velocity}$, and $Min_{Acceleration}$) are:

$$Min_{Attr} = (Base_{Attr}) \times R_{Attr}$$
$$Max_{Attr} = Min_{Attr} + r_{Attr}$$
where $R_{Attr} \in \mathbb{Z}^+, R_{Attr} \leq 5,$
and $r_{Attr} \in \mathbb{Z}^+, r_{Attr} \leq Base_{Attr}.$

R_{Attr} is a random integer selected between 1 and 5 (standing for R_{Area}, $R_{Lifetime}$, $R_{Velocity}$, $R_{Acceleration}$ and generated seperately for each attribute), and serves as the degree of these attributes. For example, if $R_{Velocity}$ is set to 1 for feature type F_1 and 3 for feature F_2, then the instances created using F_1 will be slower. By multiplying the R_{Attr} values with the related base parameters ($Base_{Area}$, $Base_{Lifetime}$, $Base_{Velocity}$, and $Base_{Acceleration}$), we find the minimum attribute values. Then, we randomly generate and integer value r_{Attr} (stands for r_{Area}, $r_{Lifetime}$, $r_{Velocity}$, $r_{Acceleration}$ and generated seperately for each attribute), and add r_{Attr} to minimum attribute value to find the maximum attribute value. Additionally for acceleration, we randomly select a sign in order to characterize negative acceleration (deceleration). As an example, let $Base_{Velocity}$ be 20, $R_{Velocity}$ value is generated as 4, and r_{Attr} is generated as 11; then, $Min_{Velocity}$ is $4 \times 20 = 80$ and $Max_{Velocity}$ is $80 + 11 = 91$.

For areal evolution, we generate a random real number (ev) between -2 and 2 and we get largest previous (floor - $\lfloor ev \rfloor$) and the smallest following (ceiling - $\lceil ev \rceil$) of this number. Then,

$$Max_{AreaEvol} = (Base_{AreaEvol})^{\lceil ev \rceil}$$
$$Min_{AreaEvol} = (Base_{AreaEvol})^{\lfloor ev \rfloor}$$

Generation of Patterns

Parameters

N_{Core}: The number of core patterns to be generated.

$S_{Overlap}$: The number of overlap patterns to be generated for each core pattern.

For the generation of patterns, we have followed the heuristics described by Agrawal and Srikant (1994), and Huang et al. (2004). Agrawal and Srikant model the frequent itemsets as a retailing environment, assuming people tend to buy items together from a maximal itemset. We extended this idea into spatiotemporal context; and divided the patterns into two groups -*core* and *overlap* patterns. *Core* patterns can be seen as maximal frequent itemsets in classical data mining, they are designed to occur more frequently. The increase of frequency in core patterns is achieved by overlap patterns. An *overlap* pattern is associated with a core pattern, and it includes all the feature types that the core pattern has and one more different feature type. Hence, the core pattern's feature types recur in an associated overlap pattern with the addition of a new feature type. The instances created using a overlap pattern will eventually follow the core pattern too, but their occurance frequency will be smaller.

The number of core patterns to be generated is determined by parameter N_{Core}. For each k between 1 and N_{Core}, we create $(k+1)-cardinality$ core pattern. A $k-cardinality$ core pattern is formed by arbitrarily selecting k feature types and adding them to the feature set of this pattern. For each core pattern created, we generate $S_{Overlap}$ overlap patterns associated with that core pattern. In order to create an overlap pattern, we first copy all the features of the associated core pattern and add one more randomly selected feature to the feature set of the overlap pattern.

Generation of Spatial Neighborhoods and Instances

Parameters

S_{Inst}: The number of instances to be generated for each feature in a pattern.

M_{SpN}: The number of instances to be generated in a spatial neighborhood for each feature type in a pattern.

L_{Noise}: The ratio of noise instances to be generated.

After creating core and overlap patterns, the next step is forming spatial neighborhoods ($< SpN >$) and generating the instances in these neighborhoods. Spatial neighborhoods are related by a pattern; furthermore, they are implemented as points denoted $SpN^i_k = (X_k, Y_k)$, where i indicates the associated pattern, k is the index of the spatial neighborhood, and X_k and Y_k are the coordinates of the point. The instances that are created in the same spatial neighborhood have the same birth locations and birth times. The location of a spatial neighborhood is randomly generated in spatial framework. For each spatial neighborhood, we create M_{SpN} instances for each feature of a pattern. The total number of instances to be created is S_{Inst} for each future. Note that, for each pattern we create $\lceil \frac{S_{Inst}}{M_{SpN}} \rceil$ spatial neighborhoods.

Instance Attributes To generate instances, we use the attributes from the associated feature. The maximum and minimum attributes of a feature type specifies the range of the values for an asociated instance. For each instance, using the minimum and maximum attribute values from associated feature, we randomly pick the values for area (area of the birth polygon), number of vertices, lifetime, velocity (starting velocity), acceleration, and areal evolution factor. For example, let $Min_{Velocity}$ be 80 and $Max_{Velocity}$ is 91. We randomly select an integer between 80 and 91 and assign this value to as the starting velocity of the instance $Inst^i_j$. Same process is applied for area, number of vertices, lifetime, acceleration and areal evolution.

Representation of Instances An instance is a list of polygon and timestamp pairs. The number of vertices (denoted by n) is constant during an instance's lifetime. The vertices of a polygon are represented with a source point (denoted as src) and n vertex vectors representing the relative location of vertices to the source point. A source point is always inside the polygon. A polygon is represented as following:

$$Poly = \{(X_{src}, Y_{src}) | (x_1, y_1), \ldots, (x_n, y_n)\}$$

where (X_{src}, Y_{src}) is the coordinates of the source point and each (x_i, y_i) pair stands for a vertex vector.

Unit Polygon and Birth Polygon Generation To create random polygons with desired areas and number of vertices, a unit polygon generation algorithm is used. Given number of vertices (n), we initially partition a unit square into n regions as shown in Figure 2. Each of these regions cover $2\pi/n$ degrees and a random point is selected from each region. These points are utilized for the vertex vectors of the birth polygon by increasing the magnitude of these vectors by a growth factor (g). Lastly, we set the location of source point to the associated spatial neighborhood.

$$Poly_{Unit}=\{(0,0)|(x_1,y_1),\dots,(x_n,y_n)\}$$

$$g=\sqrt{\frac{DesiredArea}{AreaofUnitPolygon}}$$

$$Poly_b=\{(X_S,Y_S)|(gx_1,gy_1),\dots,(gx_n,gy_n)\}$$

where $Poly_{Unit}$ is the unit polygon, $Poly_b$ is the birth polygon, and X_S and Y_S is the x and y coordinates of associated spatial neighborhood relationship.

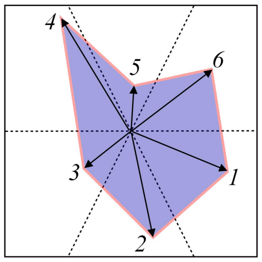

Figure 2: Unit polygon generation for $n = 6$: Each region covers $\pi/3(= 2\pi/6)$ degrees and six random points are selected from each of these six regions

Motion and Areal Evolution of Polygons The movement of instances in ERMO-DG are linear. The parameters for the movement are decided using the (initial) velocity (V_0) and acceleration (a) attributes of an instance. We randomly select these attributes using the maximum and minimum attribute values taken from associated feature type. For instance, to determine initial velocity, V_0, we select an integer between $Min_{Velocity}$ and $Max_{Velocity}$. The motion is implemented using the classical linear displacement function (*i.e.* $\Delta x = Vt + at^2$). We initially decide the direction of the movement by selecting a random angle (θ) between 0 and 2π. We then create a displacement function for x and y coordinates; those are following:

$$\Delta x_t = V_x t + a_x t^2$$

$$\Delta y_t = V_y t + a_y t^2$$

where Δx_t and Δy_t represent the displacements in x and y coordinates as two functions of time (denoted by t), similarly V_x and V_y shows initial velocities, a_x and a_y shows accelerations in x and y coordinates. Note that,

$$V_x = V_0 cos(\theta)\ and\ a_x = acos(\theta)$$

$$V_y = V_0 sin(\theta)\ and\ a_y = asin(\theta)$$

Therefore, for each timestamp we displace the source point of the region polygons by using the displacements in x and y coordinates seperately. For the areal evolution, we pick a real number between the $Min_{AreaEvol}$ and $Max_{AreaEvol}$ attributes of the associated feature, and use it as the ratio between the birth polygon area and death polygon area.

$$Ratio\ of\ area = \frac{Area\ of\ birth\ polygon}{Area\ of\ death\ polygon}$$

Then the amount of area change (called areal evolution factor - aef) at each timestamp:

$$aef = \sqrt[T]{Ratio\ of\ area}$$

where T is the lifetime of the instance. Hence, the representation of a polygon at time t_i (where $0 \leq i < T$) , given spatial neighborhood ($SpN = (X_S, Y_S)$), displacement functions (Δx_t and Δy_t), and areal evolution factor (aef) is

$$Poly_{t_i}=\{(X_S+\Delta x_{t_i}, Y_S+\Delta y_{t_i})|(aef^i x_1, aef^i y_1),\dots,(aef^i x_n, aef^i y_n)\}$$

Noise Instances The last part of instance generation is the creation of noise instances. In order to determine the number of noise instances to be generated L_{Noise} parameter is used. L_{Noise} is the ratio between the number of noise instances and the number of total pattern-related instaces. For a k-cardinality pattern P_i, we have k features; therefore, we create $k \times S_{Inst}$ pattern-related instances. Let $N_{i-Noise}$ denote noise instance count for P_i. Then,

$$N_{i-Noise} = k \times N_{Inst} \times L_{Noise}$$

In order to create a noise instance for a pattern, we randomly pick a feature that is not in the feature set of the pattern. After that, we select a random point (done for each noise instance) set it as the source point. The same instance generation process is applied after the selection of the source point.

Adjusting Parameters and Remarks

Parameters play a remarkable role in determining the characteristics of generated datasets; however, it is also important to mention immutable qualities of our work. The movement of instances is rectilinear with a variable speed (using acceleration); for simplicity, random, oscillatory or circular movements are not included. We also use hard thresholds (for R_{Attr} and ev values) while determining the maximum and minimum attribute values of feature types, this is done for simplicity. Additionally, areal evolution is incremental and evolution pace is continuous. The base parameters ($Base_{Duration}$, $Base_{Area}$, $Base_{AreaEvol}$, $Base_{Acceleration}$, and $Base_{Velocity}$) allows users to determine the instance-related characteristics of datasets. For example, increasing $Base_{Area}$ will result in generating larger polygons, setting $Base_{Acceleration}$ and $Base_{Velocity}$ values to 0 will result in generating immobile objects.

More importantly, pattern-related parameters (F_{Count}, N_{Core}, $S_{Overlap}$, M_{SpN}, S_{Inst} can be utilized for creating patterns with desired characteristics. One example is to change the maximum cardinality of patterns, using N_{Core} parameter. To create more repetitive, hence stronger patterns, fewer core patterns should be generated with more overlap patterns; hence, increasing $S_{Overlap}$ will result in having core patterns with higher support. The prevalence of patterns can also be adjusted in many ways. Increasing the instance areas (by increasing $Base_{Area}$) will result in creating more prevalent patterns. Users can generate long-lasting patterns by having a smaller number of spatial neighborhoods and more instances associated with such patterns. This can be done by increasing S_{Inst} or decreasing M_{SpN}.

Conclusion and Future Work

In this paper, we presented a new spatiotemporal dataset generator, named ERMO-DG, which produces instances with evolving regions over time. To our knowledge, it is the first approach to generate synthetic data specifically designed for spatiotemporal pattern mining algorithms. We have discussed the important concepts involved in the creation of a pattern generating synthetic dataset. These include creating patterns, utilization of spatial neighborhoods, and generating instances. The objective of this work is to provide a benchmarking enviroment for evaluating spatiotemporal pattern mining algorithms.

Parameter Name	Value
F_{Count}	9
N_{Core}	3
$S_{Overlap}$	4
S_{Inst}	50
M_{SpN}	10

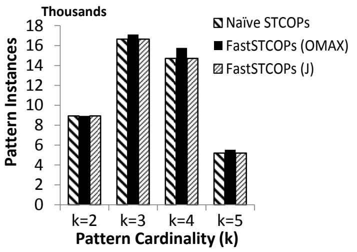

Figure 3: The parameters and the number of instances involved in pattern instance count vs. size of discovered patterns for three algorithms presented in Pillai et al. (2013)

ERMO-DG was successfully utilized for three algorithms presented in (Pillai et al. 2013). The parameters of the generated dataset and the number of pattern instances found using this dataset can be found in Fig. 3. In the chart shown in Fig. 3, *Pattern instance* demonstrates the co-occurence count of instances of different features, which participated in a spatiotemporal co-occurrence pattern (discovered by algorithms). The algorithms discovered all the patterns reported by generator. The variability in number of pattern instances are caused by interestingness measure (participation ratio) which incorporates spatiotemporal intersection and union volumes of instances. Note that, attributes of features have a strong impact on intersection and union volumes. The randomness effect created by minimum and maximum attribute selection can be observed, because without the existence of such, the number of pattern instances would be decreasing as pattern size increases. (See Pillai et al. (2013) for the details of algorithms and interestingness measures.)

In the future, we aim to enhance the features of ERMO-DG, to support rotational and curvilinear motion with different variable acceleration features for instances. Also, we plan to extend the generator to support different geometries (such as lines, circles and polygons with holes) in order to support different algorithms in pattern mining.

Acknowledgments

This work was supported by two National Aeronautics and Space Administration (NASA) grant awards, 1) No. NNX09AB03G and 2) No. NNX11AM13A.

Source Code

ERMO-DG is open-source and publicly available at (http://www.cs.gsu.edu/%7Ebaydin2/proj/ermodg.html).

References

Agrawal, R., and Srikant, R. 1994. Fast algorithms for mining association rules in large databases. In *Proc. of the 20th Int. Conf. on Very Large Data Bases*, VLDB '94, 487–499.

Andrienko, N., and Andrienko, G. 2007. Designing visual analytics methods for massive collections of movement data. *Cartographica* 42:117–138.

Brinkhoff, T. 2002. A framework for generating network-based moving objects. *GeoInformatica* 6(2):153–180.

Chen, S.; Jensen, C. S.; and Lin, D. 2008. A benchmark for evaluating moving object indexes. *Proc. VLDB Endow.* 1(2):1574–1585.

Düntgen, C.; Behr, T.; and Güting, R. H. 2009. Berlinmod: A benchmark for moving object databases. *The VLDB Journal* 18(6):1335–1368.

Huang, Y.; Shekhar, S.; and Xiong, H. 2004. Discovering colocation patterns from spatial data sets: a general approach. *Knowledge and Data Engineering, IEEE Transactions on* 16(12):1472–1485.

Pelekis, N.; Ntrigkogias, C.; Tampakis, P.; Sideridis, S.; and Theodoridis, Y. 2013. Hermoupolis: A trajectory generator for simulating generalized mobility patterns. In *Machine Learning and Knowledge Discovery in Databases*. 659–662.

Pillai, G. K.; Angryk, R. A.; and Aydin, B. 2013. A filter-and-refine approach to mine spatiotemporal co-occurrences. In *ACM SIGSPATIAL Int. Conf. on Advances in Geographic Information Systems*, 114–123.

Pillai, K. G.; Angryk, R. A.; Banda, J. M.; Schuh, M. A.; and Wylie, T. 2012. Spatio-temporal co-occurrence pattern mining in data sets with evolving regions. In *Data Mining Workshops, 2012 IEEE 12th Int. Conf. on*, 805–812. IEEE.

Saglio, J.-M., and Moreira, J. 2001. Oporto: A realistic scenario generator for moving objects. *GeoInformatica* 5(1):71–93.

Schuh, M. A.; Angryk, R. A.; Pillai, K. G.; Banda, J. M.; and Martens, P. C. 2013. A large-scale solar image dataset with labeled event regions. *Int. Conf. on Image Processing*.

Shekhar, S., and Huang, Y. 2001. Discovering spatial co-location patterns: A summary of results. In Jensen, C.; Schneider, M.; Seeger, B.; and Tsotras, V., eds., *Advances in Spatial and Temporal Databases*, volume 2121 of *Lecture Notes in Computer Science*. 236–256.

Theodoridis, Y.; Silva, J. R.; and Nascimento, M. A. 1999. On the generation of spatiotemporal datasets. In *Advances in Spatial Databases*, 147–164.

Tzouramanis, T.; Vassilakopoulos, M.; and Manolopoulos, Y. 2002. On the generation of time-evolving regional data*. *Geoinformatica* 6(3):207–231.