SpiderWeb: A Spatial Data Generator on the Web

Puloma Katiyar
Computer Science and Engineering Department
University of California, Riverside
puloma.katiyar@email.ucr.edu

Tin Vu
Computer Science and Engineering Department
University of California, Riverside
tin.vu@email.ucr.edu

Ahmed Eldawy
Computer Science and Engineering Department
University of California, Riverside
eldawy@ucr.edu

Sara Migliorini
Computer Science Department
University of Verona, Italy
sara.migliorini@univr.it

Alberto Belussi
Computer Science Department
University of Verona, Italy
alberto.belussi@univr.it

ABSTRACT

This demonstration presents a web-based generator for spatial data. This generator allows users to choose from a wide range of spatial data distributions and configure the cardinality of the data and the distribution parameters. It then provides three functionalities. First, it provides a visualization of how the data will look like. Second, it allows users to download this data in several standard formats including CSV and GeoJSON. Third, it provides a permalink that users can bookmark or share with their team members to reproduce the same dataset later. This service is a step towards standardized benchmarking for spatial data systems.

CCS CONCEPTS

• **Information systems → Spatial-temporal systems**; **Web applications**.

KEYWORDS

synthetic data, generator, spatial data

ACM Reference Format:
Puloma Katiyar, Tin Vu, Ahmed Eldawy, Sara Migliorini, and Alberto Belussi. 2020. SpiderWeb: A Spatial Data Generator on the Web. In *28th International Conference on Advances in Geographic Information Systems (SIGSPATIAL '20), November 3–6, 2020, Seattle, WA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3397536.3422351

1 INTRODUCTION

In many published papers, researchers often need to test their implementations of new index structures or query execution methods on large scale spatial data. While some real datasets exist, the research community also needs to try datasets with specific characteristics to highlight how the proposed research behaves under certain circumstances. Synthetic data generation gives researchers full control over the data characteristics such as data skewness, complexity of geometries, or amount of overlap between datasets [5].

SIGSPATIAL '20, November 3–6, 2020, Seattle, WA, USA

ACM ISBN 978-1-4503-8019-5/20/11.
https://doi.org/10.1145/3397536.3422351

A potential tool for generating synthetic spatial datasets with various skewed distributions has been proposed recently [5]. The generator takes a dataset descriptor, which is a vector containing information about the data to be generated. This descriptor uniquely identifies the data and consists of two parts: (1) the distribution ID for 6 implemented distributions and (2) the model parameters depending on the chosen distribution. Additionally, an affine transformation can be applied to the data, allowing for scaling, rotating, and moving the data. This generator has been successfully used in existing research to evaluate index construction, query processing, spatial partitioning, and cost model verification [5].

In this demonstration, we implement a Spatial Data Generator on the Web (Spider-Web), available at http://spider.cs.ucr.edu. This generator focuses on minimizing the time and space complexity to allow the program to serve hundreds of concurrent users and generate terabytes of data with minimal resources [2]. We anticipate that this tool will be utilized to generate large datasets (over a billion data points) with sizeable download file sizes. In order to streamline the generator, we use the following strategies:

- Streaming the generated data to reduce the user's wait times for downloading files
- Developing a more efficient algorithm for the parcel distribution generator, which previously had the highest time and space complexity of all the other distribution types

Through these methods, we created a web-based tool to improve the reproducibility of experiments on synthetic data. The web generator provides a simple interface for users to enter the desired dataset parameters. Then, users can immediately visualize a sample of the data to confirm that this is the desired datasets. After that, users can ask to generate and download the full dataset and the download starts immediately regardless of the dataset size. In addition, the web generator generates a permalink that users can bookmark or share with their team members to regenerate the same dataset later.

2 DATASET GENERATION

As described in the introduction, this web-based generator was developed primarily with time and space efficiency in mind. In this section, we detail several techniques used to achieve this.

2.1 Streaming the Generated Data

Since the generator is expected to be used to generate large datasets, it is crucial that the file download of the data begins in less than a seconds of the user clicking a button on the web interface. Some existing systems, e.g., MNTG [3] and TAREEG [1], generate the dataset to disk before providing the user with a download link. This requires users to enter their email address, enter a queue of requests, and when their data is generated they receive an email with the download link. This might seem inevitable if the dataset is very large and would take several minutes to generate.

In this demonstration, we address this problem by designing the generator to stream the data from the back-end to the browser rather than generating the data, storing it in a file on the server, and then transferring it over. This relies on an HTTP feature called *chunked transfer* which generates the response one chunk at a time and sends it back to the requester. So, we design our generator to *stream* the generated data back to the client rather than saving all of them to a file and sending a link to that file.

This approach has multiple benefits for both the user and the server. First, and most notably, it saves resources on the server. With the streaming method, the server does not store any of the data as it is being generated. Second, if the user cancels the request for data while it is in progress, the server will be notified of the disconnection and will immediately free the compute resources required to generate the remaining data. Third, this approach prevents the system from being overwhelmed with multiple consecutive requests that might consume all the server resources.

On the user side, users do not have to wait for all the data to be generated before starting to receive it. By asking the user to select a download location on their computer upfront, we avoid requiring the user's attention during the remainder of the data generation and download. This gives the user instant confirmation that their request is valid and frees them up from waiting to begin a download. Additionally, if the generated dataset was intended for use as input for a software that accepts streaming input, this technique of streaming the generated data allows the software further down the pipeline to begin immediately. For example, this data can be piped through `curl` to a Python script that processes the data as it is generated by the server. If the script had an error, for example, it will be caught immediately rather than waiting for the entire file to download first.

The technique of streaming the data consists of two parts. First, the HTTP header "Content-Disposition: attachment" is sent to the browser well before any of the generated data is sent over. Upon receiving this header, the browser triggers the save dialog box, prompting the user to select the download location before the client can begin saving data to that file. Second, the generator program forces a flush of a small chunk of data at the beginning of the data generation. By doing this flush on the *stdout* buffer, which is used to send the data to the browser, we are choosing not to rely on the implementation details of Python or the host operating system of the server and its memory capacity and buffer sizes. In the generator program, we flush the output pipe after 10 data points have been generated so that the file writing on the client side can start immediately.

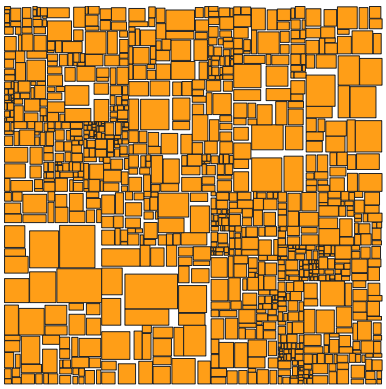

Figure 1: Parcel Distribution Model [5]

2.2 Streamed Parcel Generator

Among the six distributions currently supported by the spatial data generator [5], five of them are inherently designed for stream generation. They generate one record at a time with all records independent of each other. However, the parcel distribution is inherently sequential since it work by recursively splitting the space until the desired number of records are generated then the records are disturbed with some noise and written to the output. The final output of the parcel distribution includes geometries that represent boxes of different sizes as illustrated in Figure 1. This distribution can model land sections delineated in urban areas.

First, let us describe how the parcel generator works in its original implementation. In addition to the cardinality *card* of the generated dataset, the parcel generator takes two specific parameters r and d, where:

- $r \in [0, 0.5]$ is the minimum tiling range for splitting a box. $r = 0$ indicates that all the ranges are allowed while $r = 0.5$ indicates that a box is always split into half.
- $d \in [0, 1]$ is the dithering parameter that adds some random noise to the generated rectangles. $d = 0$ indicates no dithering and $d = 1.0$ indicates maximum dithering that can shrink rectangles down to a single point.

The original implementation of the parcel distribution algorithm relied on a *breadth-first traversal* of the tree of rectangles, as described in our previous paper [5]. However, we adapted this method to better fit with our streaming data technique.

The new parcel distribution algorithm proposed in this demo uses a depth-first traversal to start generating data at the leaf node level as quickly as possible. Compared to the other 5 distribution types, the parcel distribution is unique because generating each data point relies on the previously generated data point, or the parent bounding box. Originally, all the parent nodes would have to be generated before the first piece of data could be sent to the client. But by converting the algorithm to a depth-first traversal, we are able to generate the first data point much more quickly.

In the previous breadth-first traversal, a queue was used to store all the boxes from the second to the last level in the tree, which could take up a very large amount of memory. In the depth-first algorithm, the number of intermediately stored boxes is proportional only to

Algorithm 1: $G_{parcel}(r, d)$: Generate boxes of the parcel distribution

Input: $card, dim = 2, r, d$
Result: Stream of geometries: $\mathcal{G} = \{geom\}$

Initialize the random number generator with given seed;
$\mathcal{G} \leftarrow \{Box(0, 0, 1.0, 1.0, 0)\}$; `// Initial box at depth 0`
maxHeight =$\lceil \log_2 card \rceil$;
numSplit = 0;
numToSplit = $card$ - $2^{\max\{maxHeight-1,0\}}$;
boxesGenerated = 0;
while $boxesGenerated < card$ **do**
  $b \leftarrow \mathcal{G}$.pop;
  if $b.depth \geq maxHeight$ - 1 **then**
    if $numSplit < numToSplit$ **then**
      b_1, b_2 = split(b);
      numSplit += 1;
      Dither and output b_1, b_2;
      boxesGenerated += 2;
    else
      Dither and output b;
      boxesGenerated += 1;
      if $boxesGenerated == 10$ **then**
        Flush output;
  else
    $b_1, b_2 = split(b)$;
    $\mathcal{G}.push(b_1)$;
    $\mathcal{G}.push(b_2)$;
return $\mathcal{G}$

the log of the cardinality *card*. This vastly reduces the amount of memory needed to generate the data.

The depth-first traversal algorithm is described in Algorithm 1, with a helper function detailed in Algorithm 2. The algorithm begins with an initial 1 square unit bounding box. This box is continually split in two, generating a tree of boxes. The tree will be of the minimum height h needed to have the number of boxes on the leaf level be greater than or equal to the cardinality. If the number of boxes on the level of height 1 (second to last level) is n short of the cardinality, then n boxes will be split on that level to generate $n * 2$ boxes on the leaf level. The remaining boxes will be generated on the level of height 1. This method produces realistic variations in the sizes of the parcels.

3 DEMONSTRATION SCENARIO

This tool is best demonstrated in an interactive setting, where the audience can directly test the application. This web application will be hosted on a university web server at the domain http://spider.cs.ucr.edu/, making it available for public use. For our demonstration, we will set up a laptop for the audience to interact with the application. The audience may configure the data to their preferences, then see a visualization of the dataset and download it.

Algorithm 2: $Split(b, r)$: Split box using split range r

Input: b, r
Result: List of boxes: $\mathcal{B} = [boxes]$

if $b.width > b.height$ **then**
  splitSize = b.width $*U(r, 1-r)$;
  b_1=Box(b.x, b.y, splitSize, b.height, b.depth + 1);
  b_2=Box(b.x + splitSize, b.y, b.width - splitSize, b.height, b.depth + 1);
else
  splitSize = b.height $*U(r, 1-r)$;
  b_1=Box(b.x, b.y, b.width, splitSize, b.depth + 1);
  b_2=Box(b.x, b.y + splitSize, b.width, b.height - splitSize, b.depth + 1);
return $\mathcal{B}$

3.1 Web Interface

Figure 2 shows the current state of the main page of the data generator web interface. The top portion of the web page has multiple input fields for users to configure the dataset to their needs. The "distribution" drop-down field offers 6 different distribution types: uniform, diagonal, Gaussian, Sierpinski, bit, and parcel. The input of the "cardinality" field determines how many data points (or rectangles, in the case of parcel distribution) should be generated. "Dimension" indicates the dimensions of the data and the "format" field allows users to choose from 3 different download format types: CSV, WKT, and GeoJSON.

The next 6 input fields are specific to the type of distribution chosen, so not all will apply concurrently. These input fields are grayed out if they are unnecessary for the currently selected distribution type. Fields will become available to edit once the corresponding distribution is chosen. For instance, if "bit" is selected for the distribution type, the input fields for "probability" and "digits" will be enabled since those parameters are needed to generate "bit" distribution data. Finally, the "seed value" input field gives users the option to enter a value that will be used to seed the *random* function in the generator. This enables the user to generate a dataset that is both random and reproducible.

This generator offers a considerable degree of customization for the user, which increases its usability. Since the generator uses standard, widely accepted data distributions, it can be used in benchmarking tests.

3.1.1 Data Visualization. The third part of the web interface is a visualization space that displays how the chosen dataset will look like when generated. Once all the necessary initial inputs are given by the user, a graph will appear at the bottom of the web interface displaying the generated data as shown in Figure 2. As input values are changed, the visualization will immediately reflect the result of the change.

The graph is created using the OpenLayers API, which offers the flexibility to create multiple *vectorlayers* to display multiple datasets [4]. Users will be able to overlap different datasets and easily compare them. Additionally, users can see what the generated data will look like prior to downloading the full file. This makes it

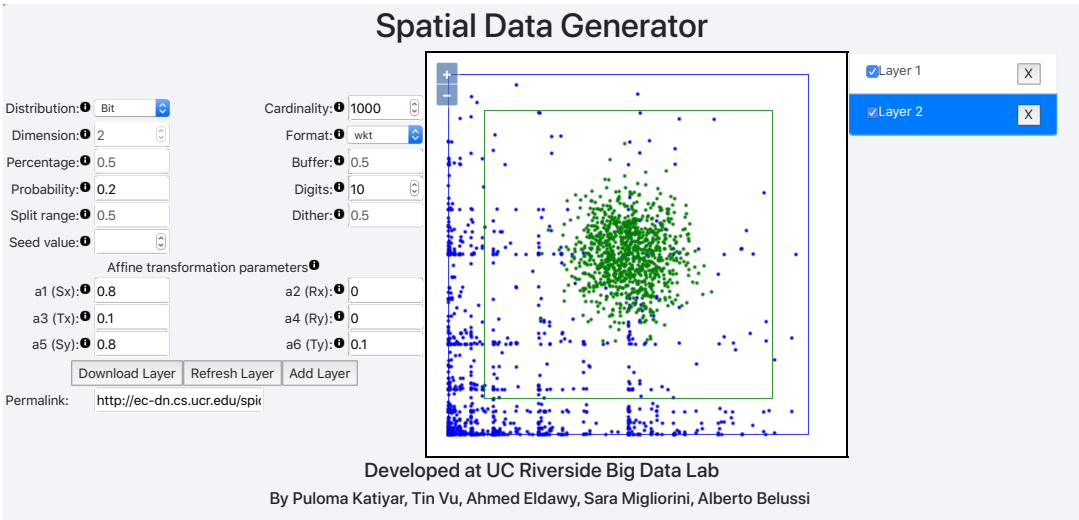

Figure 2: Spatial data generator web interface

simple for users to see how to alter the data characteristics to fit their needs.

To support this feature in the demo, the web interface sends a generation request in the background with the same parameters that the user is currently specifying but with a limited cardinality of 1,000 records. This upper limit ensures that the visualization would appear immediately on the web interface. The 1000 point limit was found to display within 1 second and still give users an accurate representation of the chosen dataset distribution.

This visualization also offers the option of transforming the data. As seen in Figure 2, the black dots in the corners of the graph can be clicked and dragged by the user to scale or rotate the data that is currently displayed. Users can also select individual datasets and move them around on the graph for comparison purposes. The values on the x-axis and y-axis will change accordingly to reflect the location of the newly transformed dataset.

3.1.2 Downloading and Accessing Datasets. The third user scenario involves bookmarking a dataset to retrieve it later. The proposed web generator automatically generates a unique ID for any dataset given its distribution and generation parameters. This unique ID consists of three parts:

- The distribution ID for 6 implemented distributions.
- The parameters dependent on the chosen distribution.
- The affine transformation applied on the generated dataset to scale, translate, or rotate it.

With these three components, a dataset can be exactly reproduced by anyone. This is the most impactful strength of this application. Since the dataset descriptor fully identifies the generated dataset, researchers who use this tool can simply cite the application's corresponding papers and list the descriptors they used. Others can then generate the exact same datasets for their own testing purposes. With this web-based tool, users can just share a link that directly generates the same dataset. Thanks to our streamed generator, no book keeping is needed on the server side and all these links are essentially provided for free.

Figure 2 shows that the application gives users access to a permalink consisting of all the dataset descriptors needed to replicate the synthetic data. This permalink updates as inputs are changed, giving users an easily accessible method of reproducing a certain dataset. Users can also share this permalink with others, such as a research group, so that they may all generate and utilize identical datasets.

4 FUTURE WORK

The simplicity and flexibility of this application makes it possible to easily expand it in the future. We plan to add more distributions to our generator. If researchers start to use the generator, we also plan to add citation information that includes the dataset ID and sample Latex code that authors can directly embed in their papers.

ACKNOWLEDGMENTS

This work is supported in part by the National Science Foundation (NSF) under grants IIS-1838222 and CNS-1924694 and by Agriculture and Food Research Initiative Competitive Grant no. 2019-67022-29696 from the USDA National Institute of Food and Agriculture. This work is also partially supported by the Italian National Group for Scientific Computation (GNCSINDAM) and by "Progetto di Eccellenza" of the Computer Science Dept., Univ. of Verona, Italy.

REFERENCES

[1] Louai Alarabi, Ahmed Eldawy, Rami Alghamdi, and Mohamed F. Mokbel. 2014. TAREEG: a MapReduce-based system for extracting spatial data from OpenStreetMap. In *SIGSPATIAL*. ACM, Dallas/Fort Worth, TX, USA, 83–92. https://doi.org/10.1145/2666310.2666403

[2] Puloma Katiyar. 2020. Spatial Data Generators. GitHub Repository. https://github.com/puloma-k/spatialdatagenerators

[3] Mohamed F. Mokbel, Louai Alarabi, Jie Bao, Ahmed Eldawy, Amr Magdy, Mohamed Sarwat, Ethan Waytas, and Steven Yackel. 2013. MNTG: An Extensible Web-Based Traffic Generator. In *SSTD (Lecture Notes in Computer Science, Vol. 8098)*. Springer, Munich, Germany, 38–55. https://doi.org/10.1007/978-3-642-40235-7_3

[4] OpenLayers. 2029. Class: VectorLayer. https://openlayers.org/en/latest/apidoc/module-ol_layer_Vector-VectorLayer.html

[5] Tin Vu, Sara Migliorini, Ahmed Eldawy, and Alberto Belussi. 2019. Spatial Data Generators. *1st ACM SIGSPATIAL International Workshop on Spatial Gems* (2019).