PARTITIONING WIDE AREA GRAPHS USING A SPACE FILLING CURVE

Cyprien Gottstein[1], Philippe Raipin Parvedy[1], Michel Hurfin[2], Thomas Hassan[1] and Thierry Coupaye[1]

[1]TGI-OLS-DIESE-LCP-DDSD, Orange Labs, Cesson-Sevigné, France
[2]Univ Rennes,INRIA, CNRS, IRISA, 35000RENNES, France

ABSTRACT

Graph structure is a very powerful tool to model system and represent their actual shape. For instance, modelling an infrastructure or social network naturally leads to graph. Yet, graphs can be very different from one another as they do not share the same properties (size, connectivity, communities, etc.) and building a system able to manage graphs should take into account this diversity. A big challenge concerning graph management is to design a system providing a scalable persistent storage and allowing efficient browsing. Mainly to study social graphs, the most recent developments in graph partitioning research often consider scale-free graphs. As we are interested in modelling connected objects and their context, we focus on partitioning geometric graphs. Consequently our strategy differs, we consider geometry as our main partitioning tool. In fact, we rely on Inverse Space-filling Partitioning, a technique which relies on a space filling curve to partition a graph and was previously applied to graphs essentially generated from Meshes. Furthermore, we extend Inverse Space-Filling Partitioning toward a new target we define as Wide Area Graphs. We provide an extended comparison with two state-of-the-art graph partitioning streaming strategies, namely LDG and FENNEL. We also propose customized metrics to better understand and identify the use cases for which the ISP partitioning solution is best suited. Experimentations show that in favourable contexts, edge-cuts can be drastically reduced, going from more 34% using FENNEL to less than 1% using ISP.

KEYWORDS

Graph, Partitioning, Graph partitioning, Geometric partitioning, Spatial, Geography, Geometric, Space Filling Curve, SFC, ISP

1. INTRODUCTION

In the last decade, graphs have seen extreme attention from IT industry and Computer Science researchers. Because graphs can fit many use cases induced by the development of services from Google, Facebook or Amazon, and new marking opportunities based upon graph analysis and recommendation engine, they have reached an unprecedented scale. Along with the explosion of social networks graph knowledge and analyse about people has become a very profitable resource and represent a massive subject of interest. Yet, graphs effective range of application is much wider. From security and fraud detection to financial risk management, graphs analysis is used extensively in many domains. As performing such analyses requires heavy computation, it is essential for those graphs to be widely distributed across a large set of machines, i.e. partitioned. As K. Andreev and H. Räcke shown, graph partitioning is a NP-Complete problem [1].

At this day, a plethora of graph partitioning solutions have been proposed to solve this problem. The large range of strategies even makes it difficult to choose the appropriate graph partitioning solution for a given graph. It has been shown in recent surveys on graph partitioning [2], graph

DOI:10.5121/ijdkp.2021.11102

research domain intensely focuses on the Online Analytic Processing (OLAP) Context and the infamous scale-free graphs. Scale-free graphs, including social graphs, are graphs whose degree distribution follows a power law. The OLAP graph partitioning standard, when it comes to large scale graphs (including scale-free graphs), is FENNEL [5].We describe FENNEL in section 2.

Nevertheless, scale-free graphs have additional characteristics contextual to their use cases. Very few scale-free graphs hold precise spatial information onto their nodes and, as a direct consequence, data spatiality has been left aside in these works. Geometric graphs, graphs whose nodes and edges are associated to spatial elements placed in a plane, and geometric-aware partitioning strategy have received little interest in the past decade.

Regardless, we believe geometry's popularity in graph partitioning will swiftly be rebuild through the growth and construction of the Internet of Things. Through giant companies like Amazon or Microsoft which have already launched new services such as Microsoft Azure Digital Twins and other companies working on smart-cities or smart-industry, new digital uses are being conceptualized and enabled by the IoT. To answer those new use cases, we anticipate a new kind of graphs will emerge through the aggregate of multiple types of infrastructure. From roads, power or water supply chain to buildings and equipment’s and ultimately connected object from the IoT, each modelled as graphs with their own set of properties and combined together would appear, what we will refer until the rest of the paper, Wide Area Graphs (WAG).

Some major characteristics will be common to all WAGs. WAG is a sub family of geometric graphs. Recall that geometric graphs are graphs for which vertices or edges are placed on a plane. Therefore, each node of a WAG has an associated location. Also, for a graph to be considered as a WAG it must be composed of millions of nodes and edges and cover an area as large as a country’s surface or even wider. At last, a WAG should be infrastructure related e.g. mixing power grid networks, road networks, buildings, equipment’s or even supply chains. Furthermore, it should have layers associating local objects interacting with each other and connected to a larger network. We do not include anything regarding connectivity and density properties into this high level characterisation of a WAG because these aspects may vary significantly from one WAG to another. However we envisiona WAG as a highly polarized version of graphs issued from Finite Element Meshes (FEM).

FEM are used to simulate real world experiences as wind or water movement (see Figure 1). A planar graph induces a low connectivity ratio, compared to scale-free graphs, and a low edge Euclidean distance as edges only exists between nodes that are close to each other on the plane. A mesh is an aggregate of non-intersecting elements like triangles or tetrahedron. When converted into a graph either nodes are the elements and edges binds adjacent element together or nodes are the intersection points between the elements and edges represents their border. Because of this creation rules, FEM graphs are planar graphs with varying density gradually peaking usually within a single continuous area. WAGs connectivity does not have such limits and may possess high connectivity nodes with edges going further than their direct neighbours leading to crossing edges and higher edge Euclidean distance. In Wide Area Graphs, the covered space may not be square and embeds lots of empty space with multiple scattered high density points. Alongside harsh and smooth node density gradation, WAGs would then match more the not-so-well shaped meshes [3]. At last, in FEM a given location is always matched to a single element so in graphs issued from meshes there may never be overlapping node, this rules does not hold true in WAGs.

In this paper we tackle the problem of WAG partitioning. Geometry is inherent to WAGs and makes a high differentiation property to partition the nodes of a WAG. Geometric strategies have already been studied in the graph partitioning context and we deem appropriate to use geometry to partition Wide Area Graphs. Such kind of strategy comes with a price, raw geometric partitioning without heuristics may end very costly. To mitigate this problem, Pilkington and Baden published in 1994 Inverse Space-Filling curve Partitioning (ISP) [4]a much cheaper geometric partitioning solution relying on a Space-Filling Curve.A space-filling curve is a mapping from a multidimensional plane to a single dimension line. A curve construction algorithm is based on recursion and self-similarity through the levels, each curve is associated to a level of zoom to fix its granularity. The interesting perk about space-filling curve is their aptitude to translate the spatial proximity from the multidimensional plane into the single dimension line. ISP is central to our research and we essentially propose a context extension of the original paper to cope with possibly upcoming Wide Area Graphs. Our foremost objective is to figure out for which kind of Wide Area Graphs ISP is relevant.

This research belongs to Thing'in the future, an IoT oriented research project direct by Orange, France largest ISP provider and Telecommunication Company. Thing'in the future main idea is to develop a platform able to map connected and unconnected objects of the real world into a unified graph representing their interactions. Because the large majority of data directly represent physical objects, the resulting graph is highly connected to geography and can be interpreted as a WAG prototype. As our objective is to study the WAG family and not only the single specific Thing’in the future graph, we will also consider synthetic graphs during our experiments.

Through the content of this paper we bring the follow contributions. First, we give an update to ISP by confronting it to the context of Wide Area Graphs which are basically a tougher graph partitioning problem than FEM graphs. Next, we expose to what extent edge Euclidean distance is critical to ISP performance. We propose and establish analysis metric and produce a matrix showing the limits of ISP helping decide whether or not to use ISP to partition a graph. Finally we measure its benefits and drawbacks through a set of metrics, mainly comparing our results against FENNEL algorithm [5].

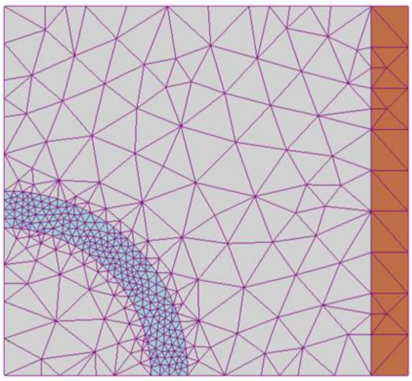

Figure 1. Illustration of a graph issued from a mesh.

2. RELATED WORK

Graph partitioning is a humongous field of research. The graph partitioning, or the balanced k-partitioning, problem is described as follows: given a graph G and a number k corresponding to the number of partitions, we seek to cut G into k balanced pieces while minimizing the number of edges cut. Edge-cut is the default performance measure to any graph partitioning strategy; it represents the ratio of edges for which both ends are stored on separate partitions. Since concretely, each partition is stored in different servers, it is important to have a minimum amount of edge-cuts as it dramatically increases edge traversal time (due to network latencies): a basic operation for a graph algorithm. Obviously, **Load-Balancing** is also mandatory; a fairly distributed load is needed to provide horizontal scalability.

In this section, we depict some of the state-of-the-art strategies in the field of graph partitioning. We do not aim to cover the whole field of research, but instead we will limit ourselves to the most relevant research related to our target, e.g. partitioning strategies able to partition a WAG. For example, a strategy such as METIS [6], the most popular example of multi-level strategy, is considered to be extremely costly if it is intended to run on WAG scale. While METIS can be applied on any type of graphs, some strategies are specialized for geometric graphs.

A well-known approach for geometric partitioning is Recursive Coordinate Bisection [7] (RCB).The RCB method sorts the graph nodes based on the most expanded dimensional axis and then recursively bisects that axis to distribute the load evenly. This approach then evolved with random sphere [8] and partitioning based on space filling curve: Inverse Space-Filling Partitioning [4] (ISP). Random sphere strategy performs geometric sampling over the mesh to find an efficient circle center point to bisect the mesh with an even load; the algorithm is then executed recursively to reach the given number of partitions. ISP also relies on geometry but with a much cheaper approach based on SFC. Nodes which are close on the multi-dimensional space are mapped close on the single dimension, then they are grouped together and form a partition. We describe how ISP works in Section 3.

Both RCB and Random sphere are geometric strategies which could have been used to partition WAG. But given that RCB is known to yield partitioning of poor quality [9] and random sphere works best on “well-shaped” graphs, which is not what should be expected from Wide Area Graphs, we will not consider those strategies for our comparisons.

More recent works aim to partition graphs based on geometry [10]–[12]. Akdogan proposes to partition geometric graphs using Voronoi tessellation whereas Volley places data on a 3D sphere representing the earth and groups data spatially close to reduce edge-cuts. These solutions work relatively well but require a lot of computations. Delling et al. strategy is based on the presence of natural cuts within a graph that minimize edge-cuts. Their solution detects such cuts to build partitions. Unfortunately the natural cut hypothesis seems unlikely to hold within a WAG.

Today, graph partitioning is considered to be divided into two categories: offline which requires to store the whole graph in memory to perform global partitioning decision and online with low requirement strategies based on local decision making. Most online strategies are based on streaming. Streaming is a powerful approach as it handle well the giant scale induced by scale-free graphs or Wide Area Graphs. To partition a graph using a streaming strategy is in general done in a single pass and partitions are built through the streaming process. Every nodes of the graph are streamed exactly once and their placement is decided with a scoring method. A score is generated with each partition paired with a given node, the partition with the highest score obtains the streamed node. Upon stream completion, nodes have all been affected and may stay in their partition, the partitioning is done.

As far as we know, the first scientific study of streaming graph partitioning was realised by Kliot and Stanton [13]. In their research they tried various nodes placement strategy with different heuristics. They concluded in their paper that the best heuristic they could establish for graph streaming partitioning was Linear Deterministic Greedy (LDG). Later on, Tsourakakis et al. [5] greatly extended their research with FENNEL. FENNEL basic approach is very similar to LDG, in fact it is a generalized framework capable of reproducing LDG while bringing subtle configuration possibilities. Streaming strategies all share the same simple base, yet they offer a large number of declinations each ending with different results.

3. INVERSE SPACE-FILLING CURVE PARTITIONING EXTENSION

In this section we discuss how we extend ISP previous work. As said before, geometric partitioning within a multi-dimensional space is very expensive. Alternatively, ISP [4] proposes to map the multi-dimensional space on a single dimension thanks to a space filling curve leading to a simplified one dimension (1D) partitioning of the curve. The curve is divided into cells and their order is defined by the curve algorithm. Each cell is a point on the curve mapped to a bounding box on the multi-dimensional space. A cell is to be assigned to a single partition and has an associated weight defined by the number of nodes it contains. Thus, the weight of a partition is equal to the sum of the weights of its cells.

When ISP was first proposed, graph partitioning streaming strategies have not yet been identified as a particular class of strategies. Therefore, ISP is classified as a geometric partitioning strategy. We believe it is fully streaming compliant and should therefore be considered as such, the only requirement is to stream the nodes in the order defined by the cells of the curve used to perform the partitioning. In our implementation of ISP, based on the total number of nodes (denoted n) and the number of partitions (denoted k), we assign beforehand a capacity to each partition. We assume for simplicity they are uniform with a capacity of roughly n/k. Uniform partition capacity is not mandatory, ISP only requires a total capacity large enough to store the whole graph. We define the load of a partition as its weight divided by its capacity.

A partition corresponds to a set of contiguous cells of the SFC. Partitions are created sequentially. To create the first partition, we start from the first cell of the SFC and we consume the cells in the order defined by the curve. We greedily assemble cells until the partition is fully loaded (i.e., its load is no longer lower than its expected capacity). Then we move onto the next partition and the next cell. As a result, the curve is segmented into k intervals and each interval is corresponding to a partition. Partitions are continuous segment over the curve which corresponds to continuous area over the multi-dimensional space thanks to the space filling curve properties.

We have respected the original design choice of ISP: even if measuring the load of a partition has seen some evolution like using the number of edges [13] or other custom metric[5], the load metric of a partition is still based on the number of nodes it actually stores. We view the number of nodes as the most suited metric because we are targeting the On Line Transaction Processing (OLTP) context. We also kept the Hilbert curve[14] as it has been proven to be the optimal space filling curve for spatial proximity preservation by Knoll and Weis[15]. At this point, whenever we mention a Space Filling Curve (SFC), we will implicitly assume it is the Hilbert SFC.

Our main contribution is about the design of a dedicated ISP [4]to partition WAG which are harder to handle than graphs issued from meshes. Besides the properties we described for WAG, we also mentioned that nodes may overlap in a WAG. During the partitioning process partitions will be assigned on average a few more nodes than expected (except for the last one which will be less weighted). Indeed, the last cell assigned to a partition may have more nodes than necessary and, therefore, its assignment to the partition can potentially overload it. On the contrary, the last partition will consist of the remaining cells and will be potentially be under loaded. The possible imbalances are due to the fact that cell unit is the granularity adopted when defining the partitions: whatever the weight of a cell, it is never subdivided to be shared between two consecutive partitions. This directly relates to how density and node overlapping is handled in a SFC.

SFCs are usually combined with the adaptive refinement technique which allows the SFC to zoom in and out depending on current needs e.g. local density for graph partitioning. With graph where nodes never overlap e.g. graph issues from meshes, such method ensures that each cell will store at most one node. However, in WAGs nodes may overlap e.g. a building with multiple-storeys stored in a 2D plane in which case the cell's weight will remain unsplittable even with adaptive refinement. This is a non-trivial issue as instead of assembling cells which have a weight of exactly one node to build a partition, we may have to cope with high weight cells tougher to balance.

Figure 2 illustrates this risk. If the cells have different weights (1, 2, 3 or 5 in the example), the consumption of the cells in the order defined by the SFC sometimes leads to achieve exact load balance (at the bottom of the figure) or, on the contrary, leads to exceed partition capacity threshold by adding a last cell with too many nodes (at the top of the figure). As such, the existence of super cells (SFC cells with extreme load) may or may not be problematic. High weight cells are the reasons why partitions tend to have slightly more nodes than expected.

4. EDGE DISTANCE AND DENSITY, ISP DECIDING FACTOR

ISP can potentially be applied to any kind of graph as long as each of its nodes has position on aEuclidean space. We argue in this section that there are precise conditions for ISP to perform well. The decisive factor is the distance covered on average by an edge in proportion to the surface covered on average by a partition, we defined it as:Edge Distance To Partition Square Size(**EDTPS**).

The idea is the following: the more we increase the number of partitions, the more the space covered by each partition decreases as the space is finite and the likelier we are to produce edge-cuts using a geometric partitioning strategy such as ISP. The higher the EDTPS, the closer is an edge from being as large as the average square surface of a partition. Geometric partitioning with low EDTPS should therefore behave well.

Let G(V,E) be a directed geometric graph placed on a 2D plan. The graph G has a surface englobing polygon P composed of points $\{(x_i, y_j)\}$ so that P= $((x_0, y_0), (x_1, y_1), \ldots (x_{n-1}, y_{n-1}))$. The points in P are all sorted in order. Let D(e) be the Euclidean distance between the source and destination for each edge e of E. We first define the function used to convert a polygon to a surface area.

$$Area(P) = \frac{1}{2}\sum_{i=0}^{n-1}(x_i y_{i+1} - x_{i+1} y_i) \quad \text{Where } x_n = x_0 \text{ and } y_n = y_0$$

Equation 1. Area formula

Then, the EDTPS formula is as follows:

$$EDTPS(G,k) = \frac{\sum_{e \in E} D(e)}{|E|} * \frac{1}{\sqrt{\frac{Area(P)}{k}}}$$

Equation 2. EDTPS Formula

A graph partitioning solution is evaluated over its ability to preserve edge from being cut and how well it maintains load balance. ISP, as mentioned in Section 3, may have trouble handling graph with high local density peak, especially without adaptive refinement. To measure those potential troubles we introduce our second metrics:Cell Density To Partition Capacity(**CDTPC**).

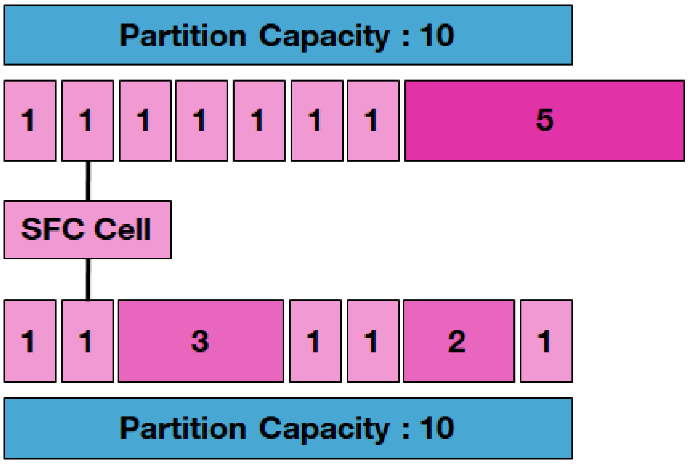

Figure 2. ISP Imbalance scenario

As shown in the second example of Figure 2, a hyper dense SFC cell may not automatically trigger imbalance, it needs to be consumed within a specific interval. This interval is the ratio between the weight of the cell and the space left in the partition. The heavier the cell, the larger the interval is to build imbalance partition. CDTPC is designed to evaluate such interval and detect in advance potential load imbalance. The higher it is, the larger the interval will be. While a high CDTPC does not necessarily imply an imbalance problem, a small ratio most likely guarantees a very sane balance.

In order to define CDTPC we need first to compute maximum cell density which we will refer as **MCD**. To do so we map every nodes position to the SFC and retain the SFC cell which includes the most nodes. The CDTPC formula is then the following:

$$CDTPC(G,k) = \frac{MCD}{\frac{|N|}{k}}$$

Equation 3. CDTPC Formula

Note that in both formulas we use k, the number of partitions, as EDTPS and rely on it. It is expected since the performance of the ISP partitioning algorithm vary depending on the number of partitions.

5. GEOMETRIC GRAPH GENERATOR

In front of the limitation of crawling graphs with interesting geometric properties and complete absence of datasets that could be close to represent future Wide Area Graphs (a geometric graph, covering a large area and with a high number of nodes and edges), we resorted early to use a graph generator. There already exists geometric graph generators [16], [17], yet, none can provide a WAG-like graph (with the desired density or connectivity characteristics).They partially allow some edge Euclidean distance configuration but not in absolute terms and lacks the input parameters to specify how density should behave across the surface of the generated graph. Such features are needed to produce synthetic Wide Area Graphs with precise EDTPS and CDTPC variation. For all these reasons we have designed our own geometric graph generator.

The generator is designed to create graphs similar to infrastructure or what we call IoT graphs. Also, we could not incorporate the preferential attachment model [18], therefore, the generator is unable to produce scale-free graph. The generator is written in Java and integrates its own R-Tree [19] to spatially index the nodes and enable fast geospatial query on the generated graph, needed for the edge-binding step. The generation process is performed in three steps: process population density through a SFC, then create and assign the nodes on the plane and finally build the edges.

The following parameters are needed to produce a graph: the number of nodes and edges, a surface plane, node density distribution behaviour in the plane and expected edges Euclidean length. The surface plane must be a square composed of GeoJSON coordinates. Density distribution is configured through a set of subcategories composed by a surface unit and a density factor. The surface unit sets the total portion of the plane which will be affected to this density factor. Then, densities factors determines, relatively to the others, how dense is a subcategory. At last, edge distance distribution is also defined through categories, each with a ratio, and a minimum and maximum possible distance. During generation the same number of starting edges are assigned to all nodes. The edge ratio is used to determine how many of those edges belong to a given distance category. At the end, nodes will have the same number of outgoing edges but a different number of incoming edges.

To generate the node density distribution across the surface of the graph, we apply a grid on the plane and map each cell to a SFC. The definition of the grid (number of columns and rows) is defined with the SFC level of granularity. Because each density subcategories are affected surface units we can determine, relatively to the total of surface units, how much of the SFC (and consequently of the underlying plane) will be assigned to each subcategory. Again, through relative ratio, surface units combined with density factors allow us to process the amount of nodes which will end in a given density subcategory and by extension its local cell density (the number of nodes populated inside a single SFC cell mapped to the plane).

For the purpose of generating a density distribution somewhat representative of the real world, we split the reserved portion of the plane assigned to a given density subcategory into multiple segments scattered across the SFC. Starting with the densest subcategory, the generator will create a segment and for this purpose it first starts at a random available cell on the curve. Then, a random Pareto function is used to calculate the segment length: number of contiguous cells of the curve. Density itself is used as a parameter in the Pareto function which leads to create small hyper dense segments against long established low density segments. Upon completion, the curve is composed by a multitude of density subcategory segments. All that remains is to populate those segments with nodes and proceed to the edge binding. We provide a visual example at Figure 3, the left part show the construction order of density segments with red being the densest subcategory. On the right we see the curve composed of density segments mapped to the 2D plane.

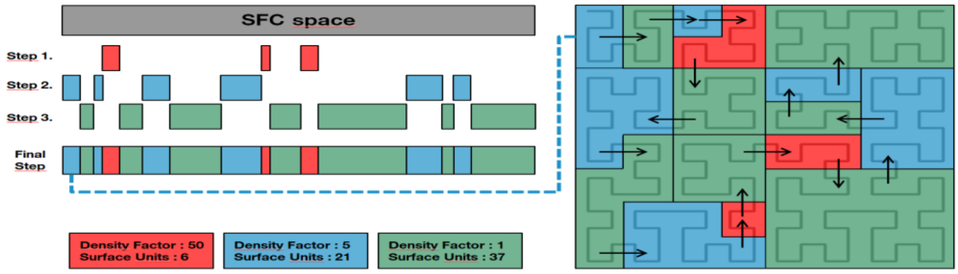

Figure 3. Density distribution of the node population

Populating the nodes only consists in taking a cell from the SFC, checking its local cell density determined earlier in the generation process, and producing the corresponding number of nodes. A random GPS location constrained within the cell bounding box is generated and assigned to each node populated within that cell.

Edge binding is much more computation-heavy. To satisfy the constraint of an edge Euclidean category we perform oversized geometric square query over the embedded R-Tree. Square have a diameter twice as large of the maximum edge Euclidean distance required for current distance category, it gives the generator nodes sufficiently spaced to create even the longest edge required. A square shape allows us to cover the surface of the generated graph through a simple sliding strategy. It also enables the generator to work with only limited chunks of the graph and with appropriate offsets to perform multiple edge binding passes on the same area, we make sure no connected component is formed because of our approach. At last, to avoid abnormal behaviours in edge connectivity distribution, the edge building process heavily relies on random selection. While random selection slows down the edge building process we deem it effective for our current needs: a stable edge connectivity distribution.

6. Experimental Setup

6.1. Setup

We evaluate our solution on a high processing computing machine with 92 GB of RAM and 48 CPU cores. The spatial graph generator and the metric processor were written in Java 8, everything related to visualization was done using Python 3.6. We choose to rely on the Graph X library of Spark (https://spark.apache.org/) to perform parts of our metrics; Spark is an expensive overhead in both development and computation but brings high scaling potential. Because Spark is built to work over Hadoop (See https://hadoop.apache.org/) and HDFS, we deployed a standalone HDFS service to handle the data read and produced by the metrics jobs. To evaluate the different partitioning solutions, we measure the amount of edge-cuts produced and the imbalance of the partitions. Regarding load imbalance, we use the **normalized maximum load** metric [5] (defined Section 4.1). We also include the custom heuristics based on EDTPS and CDTPC defined in section 4.

6.2. Dataset

We used two kinds of graphs in our experimentation: the graph of Thing'In whose characteristics are presented in the following Section 6.2.1 and synthetic WAGs produced using the graph generator already described in Section 5. We consider different WAGs with distinct characteristics as it is impossible to reduce WAGs to a single set of properties. Precisely, we aimed to first produce WAGs close to Thing'In properties and then gradually deviate to obtain a wide range of results for the application of ISP. We also wanted to find an inflexion point where ISP becomes better or worse than concurrent strategies.

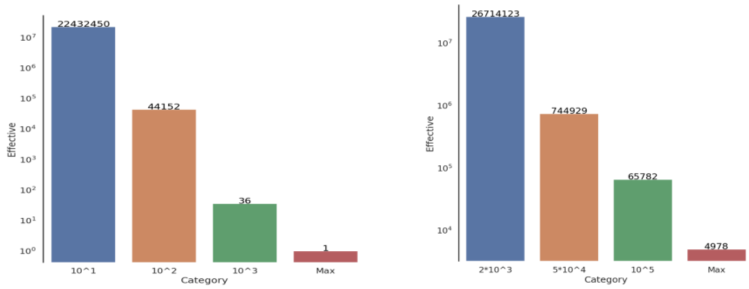

Figure 4 and 5. (Left) Thing'In Degree Distribution, blue column is for nodes with degree less than 10. (Right) Thing'In edge euclidean distance distribution

6.2.1. Thing'In Graph

The Thing'in the futuregraph is generic, it relies on web semantic to describe its content. It is the resulting aggregate of open data sources focusing on real world objects crunched into our platform. Thing'in the future is not an industrial project yet and contains inoperable data for our experiments, mainly nodes without GPS location. As we need the graph to be totally geometric, we removed about half the nodes of Thing'in the future. The rest (~ 27.2 millions of nodes and ~27.5 millions of edges) is a mix of transportation network like trains, buses, roads... and buildings information.

Given its current content, Thing'in the futureis a spatial graph. It is confirmed with Figure 4, we see a connectivity typical of a graph in which road data represents a big chunk of the graph. Therefore, Thing'in the future graph cumulative degree connectivity does not follow a power-law. Super-nodes (nodes with hyper connectivity) are absents and the most connected node maps to Rennes (the city where the Thing'in the future main developer team is located. They performed many experimentations related to this city and, consequently most edges of the graph are related to this city).The average edge Euclidean distance shown in Figure 5 is extremely low: 97.03% of the edges are shorter than 2km and very few edges are longer than 10 km (~ 0.0001%).

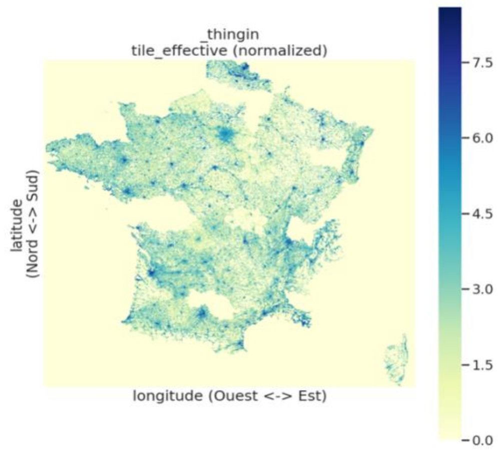

Figure 6. Thing'in the future node population distribution

Nodes are distributed in a uniform manner across the territory of France (See Figure 6). There are still some regions where no data has been injected, thus with a density of 0, whereas all large cities show progressive density towards the town center. Based on its properties, we consider Thing'in the future as a WAG. Consequently, it represents an interesting graph to be partitioned with ISP.

6.2.2. Synthetic datasets

We created this specific synthetic dataset for the purpose of showing potential correlations between EDTPS and CDTPC (see Section 4) and ISP performances in edge-cuts and load-balance. As we believe those measures are intertwined, our strategy is to generate graph similar to Thing'in the future and iteratively change edge Euclidean distance and density. Thing'in the future is not a fully fledge WAG thus and its density is not guaranteed to be representative of WAG. However, we think density in WAG should relates to human density to some degree with equivalents of deserts, empty area, and metropolises, area with extremely concentrated population.

Table 1. Density configuration

Category	**0**	**0.5**	**5.0**	**20.0**	**50.0**	**500.0**
HD	1000	1000	100	0	10	1
MD	1000	1000	100	0	10	0
LD	1000	1000	100	10	0	0

Table 1 contains the density configuration we used in our experiments. Columns represents a type of density characterised by a density strength factor in the header, the rows values are surface units e.g. how much of this density type will be present in current row (density distribution). For instance, the High Density distribution (HD) is a mix of 47.37% empty surface, 47.37% scarcely populated surface, 4.73% low density surface, 0.47% high density and 0.05% extreme density. If no surface units are allocated in a given type of density then it is absent from this density distribution. We explain how density distribution are used within our generator in Section 5. The low density distribution (LD) was designed to approach Thing'in the future density whereas HD is based on a simplified Zipf's (https://en.wikipedia.org/wiki/Zipf\%27s_law) distribution of human density across the Earth. At last, the medium distribution (MD) act as an intermediary point to provide more CDTPC variation and hopefully load-balance variation.

Our density distributions are merely relative. To obtain varying effective density on the generated graphs we also need to set graph volumetry (number of nodes and edges) and plane size. We mainly kept a fixed volumetry of roughly 50 million nodes and 150 million edges and used two plane sizes (See Table 3). With the same volumetry and density distribution, the large plane (EU) is supposed to smooth out density peaks. Instead, those peaks will be accentuated within a small plane (FR) in which we expect the presence of hyper dense hot spot, especially with HD density. With this method we produced graphs with varying density and CDTPC metrics allowing us to identify CDTPC and ISP's load balance correlation.

Table 2. Synthetic plane configuration

Type	**Latitude**	**longitude**
FR (Country)	[41.15:50.33]	[-5:9.85]
EU (Continent)	[41.15:65.33]	[-5:40.85]

Relevant to identifying ISP performances and EDTPS correlation, Table 3 lists the 8 edge Euclidean distance setup we applied to create our synthetic dataset. Just like density distribution and plane size, our edge distance settings were designed to force out a wide range of EDTPS values to evaluate its accuracy. Also, having two kind of planes allows us to prove absolute plane size is impactless on ISP. Overall, as we wanted to have tough load-balance to manage HD was set as the default density distribution with the exception of TINY- which combined with medium density distribution expressly targeted at producing graph similar to Thing'in the future. At last, the edge distance category LONG was tested with all 3 densities.

Table 3. Average edge distance and EDTPS of synthetics WAGs

Category	**AVG_Dist**	**EDTPS**	
		FR	**EU**
TINY-_MD	1041.49	0.00192	0.00076
TINY	13521.19	0.02123	0.00967
SHORT	25860.80	0.04773	0.01765
MEDIUM	48902.51	0.09026	0.03395
MEDIUM+	55766.50	0.10293	0.03562
LONG	84392.80	0.15608	0.0562
LONG_MD	86869.69	0.16034	0.05508
LONG_LD	89290.85	0.14142	0.06008
LONG+	176261.23	0.32547	0.12684
HUGE	391988.63	0.7235	0.2865

Once we finished the generation of our synthetic datasets, we had to verify if it was satisfying our initial objectives: producing graphs with varying EDTPS and density. Again, as shown in Table 3, we obtained interesting EDTPS values ranging from 0.1% to 72.35% with 4 partitions on FR plane. With 16 partition, on FR plane and HUGE category the EDTPS even reaches more than 100% percent and represent the default case where ISP is bound to perform poorly.We failed to match exactly the EDTPS of Thing'in the future (See the row TINY-) as Thing'in the future is even lower, however it remain satisfying in absolute terms as EDTPS values are extremely small.Additionally, we checked on cumulative connectivity distribution and density. As our generator does not embed the features to produce graph with scale-free properties, graphs connectivity remained low with no nodes above one hundred connectivity degree.

7. RESULTS

We seek in this experiment to compare existing state of the art streaming graph partitioning method to ISP. We applied ISP, FENNEL and LDG on each dataset with the same streaming approach. The only difference aside the decision algorithm is the streaming order. In ISP nodes were streamed ordered by the cells of the SFC whereas nodes were streamed randomly for FENNEL and LDG because, given our scale, DFS (Depth First Search) and BFS(Breadth First Search) were too expensive to apply. We first look at how strategies behave in terms of edge-cuts and secondly how they handle balance across their partitions. Note that LDG results are not showed in the tables: LDG is always worse than FENNEL regarding edge-cuts and equivalent in load-balance.

7.1. Edge-cuts

As it was our objective, we managed to reach a breaking point where the performance of FENNEL finally exceeds ISP. It is shown in Table 5 (See LONG+) that on a plane with the size of a country like France, with edge Euclidean distance past 175 kilometres on average and 16 partitions, it becomes better to apply FENNEL rather than ISP.

We expected while building our datasets that the performances of ISP would heavily correlates to EDTPS, it has been confirmed through the results. Each time EDTPS increases so does the number of edge-cuts produced by ISP without exceptions. We also managed to reach a scale where ISP is bound to perform extremely poorly (HUGE), in that case up to **95%** of the edges are cut. On the contrary, when edges are extremely short (e.g. TINY-), ISP outperforms **112** times the results of FENNEL in synthetic graphs and up to **125** times when applied to Thing'In (See Table4 and6).

We used the same space-filling curve based strategy for graph generation and ISP. Yet this has no impact on the results for two reasons. First, both SFCs are not matching. For two curves to match, those needs to use the same algorithm, granularity and map the same multidimensional plan. Although algorithm and granularity used are identical, the multidimensional plan are different. ISP covers the whole world map whereas the generator cover only the bounding boxes given in Table 3 changing both SFC cell bounding box surface and SFC cell ordering. Figure 7 is a visual example of both curves, their respective mapping and ordering are impossible to match or correlate. Second, the generator edge binding step ignores the curve presence, edges are created randomly around a node with respect to edge Euclidean distance configuration. Even if the curves were matching, it still would not give any form of advantage to ISP.

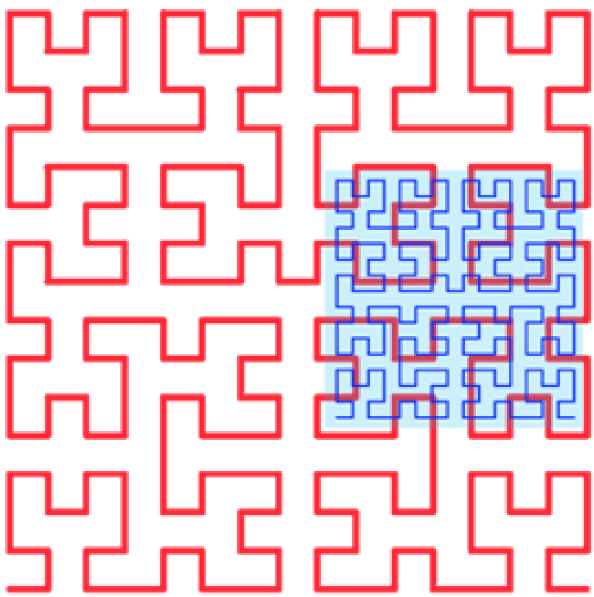

Figure 7. Visual example of generator (Blue SFC) and ISP (Red SFC) mismatch

Interestingly, we see multiple EDTPS and ISP results correlation, it must however be weighted with the amount of borders induced by the partitions which is currently ignored by our metric. Each new partitions produces additional border which may cut edges. As EDTPS does not include those borders in its estimation, we use the following interpretation rule: two graphs with identical EDTPS but different number of partitions should have different behaviour with ISP. The one with less partition should yield better results. We joined a part of the results obtained for the EU plane size in Table 6 and this interpretation rule does hold true.

Table 4. EDTPS and performances of ISP and FENNEL over the generated graphs for FR plane

Category	**K**	**EDTPS**	**FENNEL**	**ISP**
TINY-_MD	4	0.00192	38.232±0.002	0.34±0.0
	8	0.00272	45.811±0.002	0.55±0.0
	16	0.00384	50.194±0.002	0.83±0.0
TINY	4	0.02123	47.733±0.0	2.61±0.003
	8	0.03002	56.241±0.0	5.1±0.007
	16	0.04245	61.074±0.0	8.05±0.006
SHORT	4	0.04773	47.872±0.0	5.89±0.008
	8	0.0675	56.337±0.0	9.86±0.017
	16	0.09546	61.129±0.0	15.21±0.007
MEDIUM	4	0.09026	48.014±0.001	10.9±0.018
	8	0.12764	56.545±0.001	17.9±0.032
	16	0.18052	61.345±0.001	27.59±0.021
MEDIUM+	4	0.10293	47.908±0.001	11.35±0.013
	8	0.14556	56.372±0.001	19.45±0.014
	16	0.20586	61.16±0.001	27.37±0.01
LONG	4	0.15608	47.914±0.0	16.77±0.013
	8	0.22074	56.379±0.0	29.39±0.017
	16	0.31217	61.175±0.0	41.18±0.006
LONG_MD	4	0.16034	47.917±0.0	18.12±0.026
	8	0.22675	56.38±0.0	28.21±0.031
	16	0.32067	61.171±0.0	41.06±0.04
LONG_LD	4	0.14142	47.897±0.0	15.5±0.05
	8	0.2	56.361±0.0	25.99±0.082
	16	0.28285	61.159±0.0	38.21±0.116
LONG+	4	0.32547	48.07±0.0	30.06±0.017
	8	0.46028	56.564±0.001	47.71±0.013
	16	0.65093	61.364±0.001	67.48±0.004
HUGE	4	0.7235	48.056±0.001	63.88±0.017
	8	1.02318	56.557±0.002	86.02±0.007
	16	1.44699	61.359±0.002	95.49±0.004

If we compare the results of EU LONG+ and FR MEDIUM with k = 8 and 16 (see Table 4 and 5), they have similar EDTPS but EU LONG+ has consistently better edge-cuts results as it uses fewer partitions to reach the same EDTPS. The same pattern can be detected comparing EU LONG_LD to FR MEDIUM k = 8.It can also be applied to graphs within the same plane: take FR TINY with k = 16 and FR MEDIUM+ k = 4 or EU HUGE k = 4 and EU LONG_LD k = 16. At last, ISP performances are similar when both EDTPS and number of partitions are matching (See FR TINY and EU SHORT). The EDTPS values are interleaving and so do the edge-cuts results with ISP. Finally, it is easy to see EDTPS and edge-cuts correlation (See Figure 8). The only unmatched curve corresponds to HUGE which is perfectly acceptable as there is a ceiling effect for the edge-cuts. EDTPS may go further than 100% but edge-cuts are ultimately limited to 100%.

Table 5. Partial EDTPS and edge-cuts (EC) of ISP and FENNEL over the synthetic EU graphs

Category	K	EDTPS	FENNEL (EC)	ISP (EC)
SHORT	4 8 16	0.01765 0.02495 0.03529	47.279±0.001 55.79±0.001 60.601±0.001	2.49±0.003 4.18±0.004 7.24±0.004
LONG	16	0.1124	61.167±0.001	18.36±0.014
LONG_MD	16	0.11017	61.163±0.0	17.81±0.01
LONG_LD	16	0.12016	61.159±0.0	18.65±0.002
LONG+	4 8 16	0.12684 0.17937 0.25367	48.365±0.001 56.899±0.001 61.709±0.001	14.37±0.022 25.49±0.018 36.19±0.007

Table 6. EDTPS and edge-cuts (EC) of ISP and FENNEL over Thing'in the future

	K	EDTPS	FENNEL (EC)	ISP (EC)
Thing'in the future	4 8 16	0.00006 0.00008 0.00011	6.349 34.636 37.883	0.132 0.277 0.435

While EDTPS seem appropriate to evaluate ISP performance, it ignores other potential factors like connectivity and volumetry, we argue that it does not affect its veracity. We did run tests to check the volumetry impact and we couldn't observe anything significant, the problem is more about connectivity. Connectivity is ignored in both ISP and EDTPS and our dataset does not present much connectivity variation. Still, in essence, as connectivity degree rise for a given node, it has to reach further and further to connect to new nodes as there is a finite number of nodes in its neighbourhood. Consequently, edge distance tends to be longer. But as edge distance is already used in EDTPS, ignoring connectivity should not be a problem.

We expected ISP results to be a bit dissonant; nonetheless we are surprised by FENNEL extreme stability. Again, our most serious guess is based on the fact that all the synthetics graphs share the same low connectivity with absence of super nodes which FENNEL relies on to build its partitions.

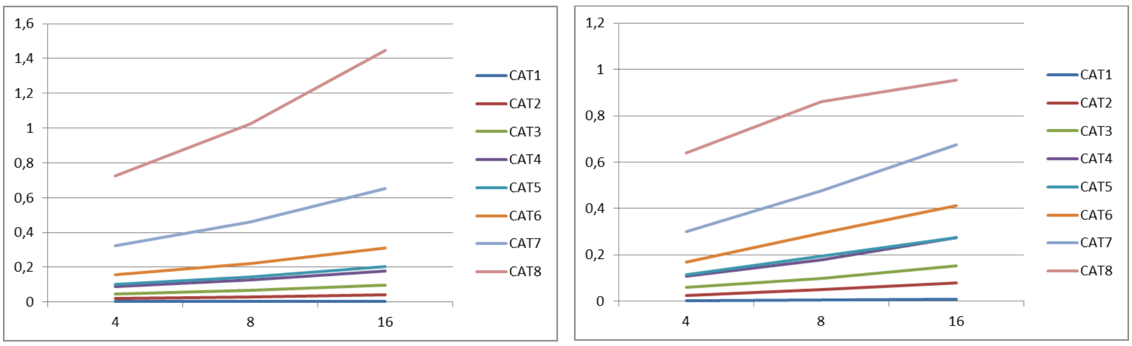

Figure 8. (Left) EDTPS, (Right) Edge-Cuts percentage of graph partitioned with ISP for each edge distance category. X-axis is the partition number, bounding box is FR, density information excluded as impact less on EDTPS and edge-cuts.

Contrary to our synthetics graphs, Thing'in the future graph possess less edges but its connectivity degree distribution has a better scaling, it could explain why FENNEL yields better results when applied on Thing'in the future. There remains a wide disparity between 4 and 8 partitions with FENNEL on Thing'in the future graph for which we have no appropriate explanation. Overall, we cannot be sure that the results similarities for FENNEL are really due to the connectivity. It is however safe to assume that edge distance has no direct impact on this solution.

7.2. Partition balance

We discuss now about how the different strategies handle balance across the partitions. FENNEL and LDG are pretty much perfect at preserving load balance across the partition, the same cannot be said for ISP. The partition imbalance ranges from 2.26% to 11.18% with 16 partitions on FR plane. Obviously, the results regarding imbalance are better for the EU plane. Its worst imbalance case has been recorded at 2.31% which is normal: there is the same number of nodes and density but for a larger plane. Looking at Table 7, we focus on the LONG+ edge distance category which has been tested with all 3 density distributions, as density gets smoother we see a reduced CDTPC and reduced load imbalance.

Table 7. CDTPC and load imbalance of ISP and FENNEL over the generated graphs for LONG+ and FR plane.

Category	K	CDTPC	FENNEL	ISP
LONG+	4	0.04916	0.0±0.0	0.22±0.002
	8	0.09831	0.0±0.0	2.13±0.013
	16	0.19662	0.0±0.0	7.09±0.022
LONG+_MD	4	0.00836	0.0±0.0	0.16±0.002
	8	0.01673	0.0±0.0	0.6±0.003
	16	0.03345	0.0±0.0	1.96±0.003
LONG+_LD	4	0.00325	0.0±0.0	0.1±0.001
	8	0.0065	0.0±0.0	0.19±0.001
	16	0.01299	0.0±0.0	0.61±0.004

The balance provided using the ISP strategy is fragile. It hugely depends on the graph, that is if it is "well behaved" or not, if the density does not peak too much relatively to the precision of the SFC. Even though a huge CDTPC ratio will not automatically trigger an imbalance problem, it is a good risk indicator as a graph with a very low CDTPC will never encounter balance trouble. A single cell might contain more than 500k of nodes all by itself. This does pose a problem because in some circumstances with a high number of partitions and extremely peaked density a single cell may have a load high enough to provoke single-cell partitions.

With the graphs generated based on the density HD given in Table 1 over a plane of a size similar to France and the Hilbert SFC of zoom 12, as demonstrated in Figure 9, 20% of the nodes in FR synthetic graphs with HD density resides in super-cells (cells with a weight over 100k nodes). In comparison, EU synthetics graphs even with HD density never reach the super-cell threshold. With LD density, the most loaded cells weights less than 10k nodes showing that a lower density will behave much better for load-balancing with ISP.

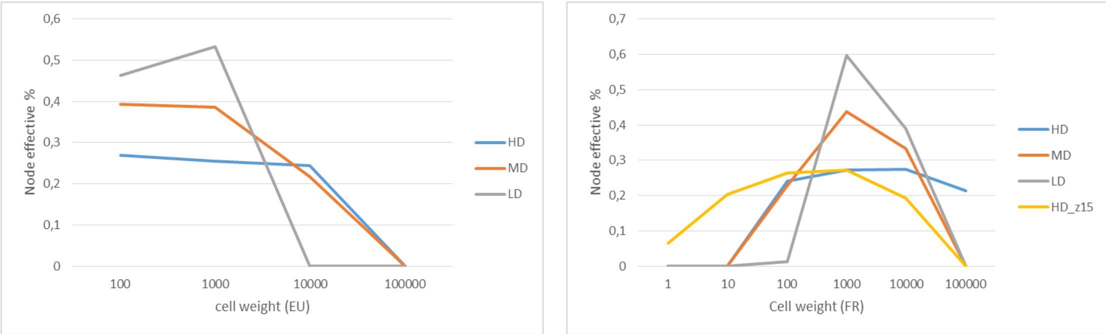

Figure 9. Node distribution by cell density in synthetic graphs (Left) Europe (Right) France. All curves used SFC zoom 12 except HD_z15 (Right) done with zoom 15.

Regardless, if this issue does happen it can be solved using SFC with higher zoom as it splits the load of a single cell into multiple cells. In theory, zoom is not guaranteed to splits the load. In practice it remains an effective approach, HD_z15 in Figure 9 precisely shows the same configuration with a higher zoom in which super-cells naturally disappear.

A cell which stores 500k nodes at zoom 12 may be reduced to 64 sub cells at zoom 15 with a load of at most 10k nodes by cell, dropping effectively the CDTPC from 17% to 1.5%. With such a small CDTPC, the risk of imbalance becomes almost zero, and in all fairness, zoom 15 is a rather standard precision level. Unfortunately, we could not afford zoom 15 as our metric system wouldn't have been able to hold the additional load memory wise. At last, if zooming in 2D is not efficient because of node overlapping, a third dimension (3D) can be used to split the load as both ISP and Hilbert can be translated in 3D. However, if any amount of dimensions is not able to split the load, ultimately, geometry itself cannot be used to distribute the graph which is not an issue. It only means this given graph is out of the scope of ISP.

8. Conclusion

In the present article we discussed and described the potential apparition of a new kind of graph named WAG. To tackle the partitioning of WAG, we proposed an extension of ISP a graph partitioning solution based on geometry originally designed to partition FEM issued graphs. Because choosing a graph partitioning strategy has become an issue due to the wide range of solutions, we also provided additional metrics to analyze why ISP performs well or not and identify potential use case. To evaluate ISP we compared it to state of the art graph streaming partitioning solutions: LDG and FENNEL. The lack of appropriate datasets inclined us to build a custom geometric graph generator to satisfy our needs. The results showed edge Euclidean distance and node density distribution have significant impact over ISP performances in both edge-cuts and load balance. ISP is highly polarized, unsuited for graphs using long range edges, typical of social graphs, producing very poor results and produces extremely good results for graphs with short edges, typical of spatial network or WAG. In a future with large scale WAG composed of tiny edges with a distance under one hundred meters as physical interaction between objects is characterized to such range, ISP could prove to be the best solution.

As for our next researches we think ISP has room for improvement. For instance, given ISP ring-like base structure, replication could be very beneficial and used for both resiliency and optimisation. Also, the analytical context is not the end goal of our research on ISP: database is. Unlike other streaming strategies, ISP adapted with the right life-cycle method can fit within the database context and remain efficient, thus we are excited to explore ISP in that direction. At last we would like to extend our WAG generator and mix in the preferential attachment model so that it's able to produce WAG which spatially logical connectivity.

REFERENCES

[1] K. Andreev et H. Räcke, « Balanced Graph Partitioning », in Proceedings of the Sixteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures, New York, NY, USA, 2004, p. 120–124, doi: 10.1145/1007912.1007931.

[2] « Recent Advances in Graph Partitioning | SpringerLink ». https://link.springer.com/chapter/10.1007/978-3-319-49487-6_4 (consulté le oct. 07, 2020).

[3] F. Payan, C. Roudet, et B. Sauvage, « Semi-Regular Triangle Remeshing: A Comprehensive Study », Comput. Graph. Forum, vol. 34, no 1, p. 86-102, 2015, doi: 10.1111/cgf.12461.

[4] J. R. Pilkington et S. B. Baden, « Partitioning with Spacefilling Curves », 1994.

[5] C. Tsourakakis, C. Gkantsidis, B. Radunovic, et M. Vojnovic, « FENNEL: streaming graph partitioning for massive scale graphs », in Proceedings of the 7th ACM international conference on Web search and data mining - WSDM '14, New York, New York, USA, 2014, p. 333-342, doi: 10.1145/2556195.2556213.

[6] G. Karypis et V. Kumar, « Multilevelk-way Partitioning Scheme for Irregular Graphs », J. Parallel Distrib. Comput., vol. 48, no 1, p. 96-129, janv. 1998, doi: 10.1006/jpdc.1997.1404.

[7] « Partitioning of unstructured problems for parallel processing - ScienceDirect ». https://www.sciencedirect.com/science/article/abs/pii/095605219190014V (consulté le janv. 20, 2020).

[8] J. R. Gilbert, G. L. Miller, et S.-Hua. Teng, « Geometric Mesh Partitioning: Implementation and Experiments », SIAM J. Sci. Comput., vol. 19, no 6, p. 2091-2110, nov. 1998, doi: 10.1137/S1064827594275339.

[9] K. Schloegel, G. Karypis, et V. Kumar, « Graph partitioning for high-performance scientific simulations », in Sourcebook of parallel computing, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003, p. 491–541.

[10] A. Akdogan, « Partitioning, Indexing and Querying Spatial Data on Cloud », p. 80.

[11] S. Agarwal, J. Dunagan, N. Jain, S. Saroiu, A. Wolman, et H. Bhogan, « Volley: Automated Data Placement for Geo-Distributed Cloud Services », p. 16.

[12] D. Delling, A. V. Goldberg, I. Razenshteyn, et R. F. Werneck, « Graph Partitioning with Natural Cuts », in 2011 IEEE International Parallel Distributed Processing Symposium, mai 2011, p. 1135-1146, doi: 10.1109/IPDPS.2011.108.

[13] I. Stanton et G. Kliot, « Streaming graph partitioning for large distributed graphs », in Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '12, Beijing, China, 2012, p. 1222, doi: 10.1145/2339530.2339722.

[14] D. Hilbert, « Über die stetige Abbildung einer Linie auf ein Flächenstück », in Dritter Band: Analysis • Grundlagen der Mathematik • Physik Verschiedenes: Nebst Einer Lebensgeschichte, Berlin, Heidelberg: Springer Berlin Heidelberg, 1935, p. 1–2.

[15] M. Knoll et T. Weis, « Optimizing Locality for Self-organizing Context-Based Systems », in Self-Organizing Systems, 2006, p. 62-73.

[16] B. M. Waxman, « Routing of multipoint connections », IEEE J. Sel. Areas Commun., vol. 6, no 9, p. 1617-1622, déc. 1988, doi: 10.1109/49.12889.

[17] M. D. Penrose, « Connectivity of soft random geometric graphs », Ann. Appl. Probab., vol. 26, no 2, p. 986-1028, avr. 2016, doi: 10.1214/15-AAP1110.

[18] R. Albert et A.-L. Barabasi, « Statistical mechanics of complex networks », Rev. Mod. Phys., vol. 74, no 1, p. 47-97, janv. 2002, doi: 10.1103/RevModPhys.74.47.

[19] A. Guttman, « R-trees: a dynamic index structure for spatial searching », in Proceedings of the 1984 ACM SIGMOD international conference on Management of data, New York, NY, USA, juin 1984, p. 47–57, doi: 10.1145/602259.602266.

AUTHORS

Cyprien Gottstein graduated from the University of Rennes 1 with a master degree in Software Engineering in 2016, Rennes, France. He worked from 2016 to 2018 as an engineer at Thales services and started his PhD in large scale graph partitioning for the Internet of Things (IoT)in 2018 at Orange Labs, Rennes, France.

Philippe Raipin Parvedy has received his PhD from the University of Rennes 1, Rennes, France, in 2004. He was a temporary lecturer and research assistant in the university of Rennes 1 from 2004 to 2006 and a post doc researcher in Orange from 2006 to 2007, since 2007, he has been a research engineer in Orange. He is currently the research program manager of the Web of Things platform in Orange. His research interests include dependability, large systems and graph databases.

Michel Hurfin holds a Ph.D. in Computer Science from the University of Rennes (1993). After one year spent at Kansas State University, he is currently conducting his scientific research activities at the INRIA Rennes Bretagne Atlantique research center. He defended his HDR (Habilitation à Diriger des Recherches) in 2004. Since 2012 he is a member of the CIDRE project that aims at addressing security problems. His research interests include distributed systems, software engineering and middleware for distributed operating systems. His recent works focus mainly on dependability and security issues in distributed systems (in particular, intrusion detection mechanisms).

Thomas Hassan is a researcher at Orange Labs, Rennes, France since 2019. He obtained a Thesis in Computer Science from the University of Burgundy, Dijon, France in 2017. His main research areas are Semantic Web, Machine Learning and Big Data. His research was applied to the domains of pervasive computing, Internet of Things, geographic information system and recommender system.

Thierry Coupaye is head of research on Internet of Things (IoT) inside Orange, Orange Expert on Future Network and Orange Open Source Referent. Prior to that, after he completed his PhD in Computer Science in 1996, he had several research and teaching positions at Grenoble University, European Bioinformatics Institute (Cambridge, U.K.) and Dassault Systems. He joined France Telecom in 2000 where he had several research expert, project manager, project and program director positions in the area of distributed systems architecture, autonomic computing, cloud/fog/edge computing and networking. He is the author of more than 75 refereed articles and has participated in multiple program and organisation committees of conferences in these areas. His current research interests include Digital Twins and Edge Intelligence (AI@Edge).