Giga Graph Cities: Their Buckets, Buildings, Waves, and Fragments

James Abello [ID], Haoyang Zhang [ID], Daniel Nakhimovich, Chengguizi Han, and Mridul Aanjaneya [ID], *Department of Computer Science, Rutgers University, Piscataway, NJ, 08854-8018, USA*

Graph Cities *are the 3-D visual representations of partitions of a graph edge set into maximal connected subgraphs, each of which is called a fixed point of degree peeling. Each such connected subgraph is visually represented as a* Building. *A polylog bucketization of the size distribution of the subgraphs represented by the buildings generates a 2-D position for each* bucket. *The Delaunay triangulation of the bucket building locations determines the* street network. *We illustrate Graph Cities for the* Friendster social network *(1.8 billion edges), a* co-occurrence keywords network *derived from the Internet Movie Database (115 million edges), and a* patent citation network *(16.5 million edges). Up to 2 billion edges, all the elements of their corresponding Graph Cities are built in a few minutes (excluding I/O time). Our ultimate goal is to provide tools to build humanly interpretable descriptions of any graph, without being constrained by the graph size.*

Techniques for analyzing massive datasets are becoming central to several communities, with the need for interactive and scalable visualizations being one of the most pressing issues.[1] Since a large variety of massive datasets can be abstracted as having an underlying graph topology, our interest is in computing graph decompositions that are useful for making sense of *massive* graphs, for which a direct layout is unavailable due to memory constraints or impractical due to screen size constraints. We focus on the identification of "global data patterns" that emerge due to the co-occurrence of the pairs of well-defined data entities.

Our goal is to find an "efficient" visual representation of any graph that is driven by the dismantling of its degree distribution. We introduce *Graph Cities* (see Figure 1) as a visual representation of such dismantling. It has the potential of bringing together streaming computations and visualization to offer "large scale" structural graph information *without* losing the ability to interactively extract finer scale connectivity. All this is possible by viewing city buildings as a representation of sequences of *graph waves*,[2] which are in turn the

sequences of *graph edge fragments* (see the "Problem Formulation" section). The corresponding visual representations can be rendered in a few minutes, with a rate of processing of 4 million edges per second. They offer humanly interpretable large-scale features of graphs with billions of edges at different levels of granularity.

The coarsest views are offered by *Graph City buildings* with floors and frustums representing waves and their interconnecting edge fragments. Each building's internal structure is represented by a Directed Acyclic Meta Graph (Meta-DAG) whose nodes can be expanded into their internal edge fragments. These edge fragments can be navigated by usual node-link diagrams at different levels of detail depending on their size. In summary, our approach follows a hierarchical edge decomposition with six levels: *edge graph decomposition, buckets, buildings, waves*, and *edge fragments*, with the bottom level consisting of "structured" node-link subgraphs of *reasonable* size that make them amenable to *human descriptions*. To the best of our knowledge, this is the first time that such a large-scale representation has been introduced for visualizing graph datasets, that is, humanly interpretable at "natural" topological levels of granularity.

Summary of Our Overall Approach

An overview of our proposed framework is illustrated in Figure 1. Our work builds upon the graph wave
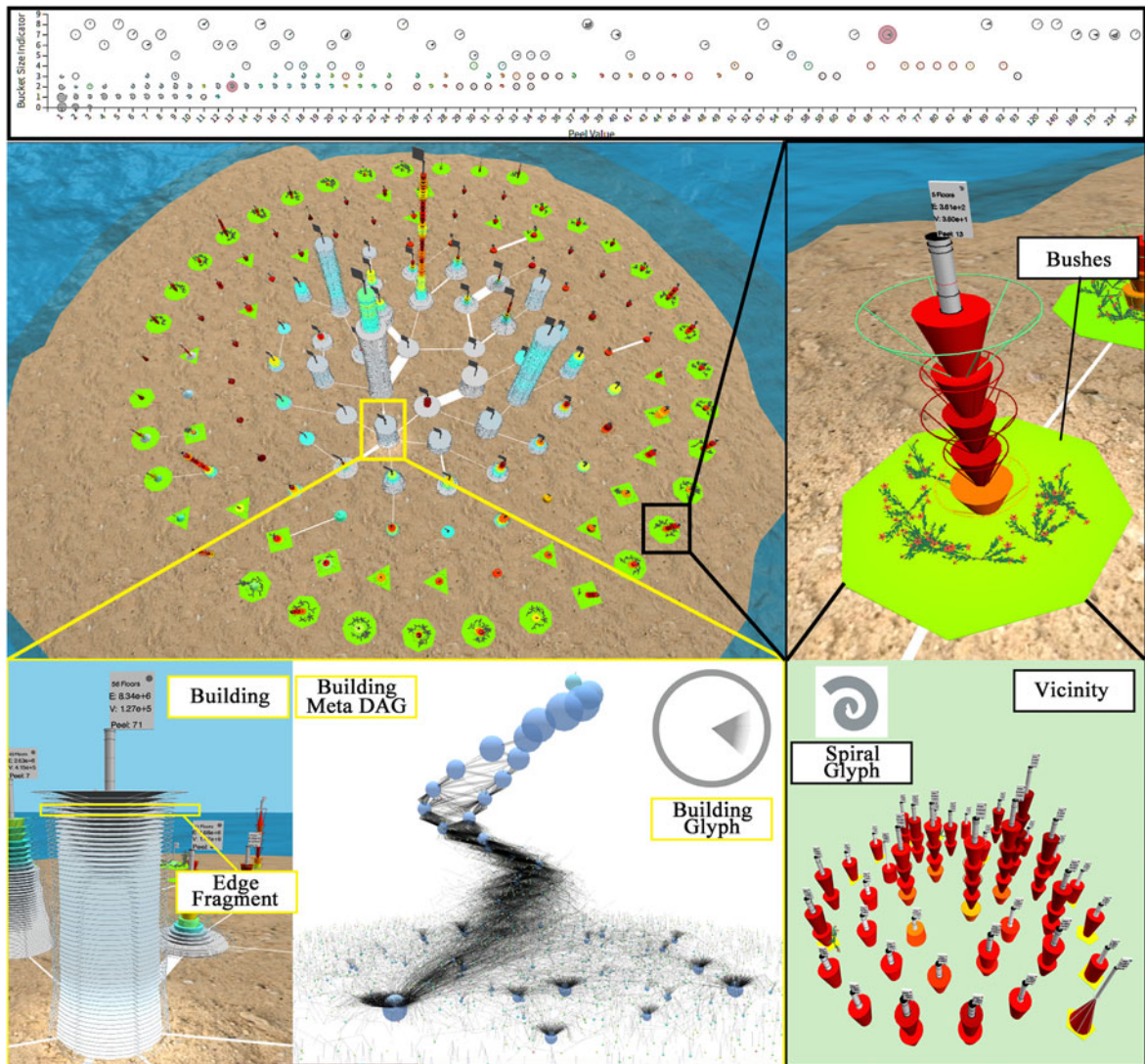
FIGURE 1. When a direct graph layout is unavailable or impractical due to memory or screen size constraints, our method can render humanly interpretable visualizations of massive graphs on the order of 10 seconds (after a few minutes of preprocessing) by leveraging the *Graph Waves*[2] decomposition. (Top) Glyph map for the Friendster social network with 1.8 billion edges. (Middle left) Graph City for the Friendster social network. (Bottom left) Individual buildings in the Graph City formed by stacking edge fragments. (Bottom center) Internal structure of a building's Meta-DAG with a peel value of 71 showing finer scale connectivity. By leveraging hierarchical decompositions, our framework allows for interactive visualization of massive graphs. (Middle right) Green bush patches representing buckets with more than one building. (Bottom right) Expansion of a green bush patch into a detailed view of its buildings' vicinity.

decomposition recently introduced in the work of Abello and Nakhimovich,[2] which is a refinement of the iterative degree edge partition.[3] Each connected subset of edges in the partition is visually represented as a building. The size distribution of the subgraphs represented by the buildings is bucketized via a polylog function of the total number of edges in the entire dataset. Buckets containing more than one building

are visually represented by *bushes* that are generated via special *L-systems*[4] [green patch areas in Figure 1 (middle right)]. A spiral embedding of this polylog bucketization provides an overall layout for all the buildings in a Graph City.

Each Graph City has a *street network* that is obtained by computing the intersection graph of the collection of vertex sets of the buildings. The 2-D

positions of the centers of the bottom floors of the buildings are used to indicate the location of the building vertex sets. The weight of the geometric edge representing the distance between two buildings is obtained as a function of the size of the intersection of the corresponding buildings' vertex sets divided by the size of their union (i.e., their *Jaccard* coefficient).

All the elements of a Graph City, including its buildings, bushes, and street network, are built in a few minutes (excluding I/O time) and storage proportional to the number of edges and vertices of the graph. We exemplify our results on a variety of large graph datasets ranging in size from a few million to up to 2 billion edges. They include the Friendster social network (1.8 billion edges), a co-occurrence keywords network derived from the Internet Movie Database (115 million edges), and a patent citation network (16.5 million edges). Our main contributions are given as follows.

› Graphs are represented as *buildings* collections. Colored building interfloor volumes encode subgraph sizes and density.
› The entire collection of buildings is accessible via a *city map*, where each map entry has associated two coordinates (*fixed point value, logarithmic bucket* ID) and a *glyph* that encodes the density and overall macrostructure of the corresponding set of fix points (see the "Touring a Graph City" section).
› A 2-D spiral layout is used to fix building locations, and also for visualizing the street network, whose edge widths are determined by the intersection graph of the buildings' vertex sets.
› Smaller buildings are bucketed into *green areas* of two types: bushes and vicinities. Bushes are generated by special natural-looking L-systems. Vicinities are specially filtered subcollections of buildings of "medium" size.
› Graph Cities can be navigated and queried at different levels of granularity using a multidirectional steering wheel and several city path exploration primitives.
› The largest buildings coming from graphs with about a billion edges are rendered in a few seconds. Generating all the corresponding buildings' rendering data from a given iterative edge decomposition takes a few minutes. This work assumes that the graph edge set has been partitioned into fixed points of degree peeling. All the algorithms used here are linear both in time and storage.

As far as we know, Graph Cities constitute the first abstract visualization of node-link representations of graphs that is linearly computable, and amenable to interactive topological exploration at different levels of granularity for massive graph datasets.

The rest of the article is organized as follows. The "Related Work" section summarizes relevant work. The "Problem Formulation" section introduces edge fragments and graph waves. The "What is a Graph City?" and "Graph Cities Street Network" sections describe Graph Cities and the corresponding street networks. The "Rendering Graph Cities" and "Touring a Graph City" sections discuss the rendering of Graph Cities and city tours. The "Description of Sample Datasets" section presents our results on three datasets. The "Machine Learning Connections" section points out some potential machine learning (ML) tasks that can be derived from graph edge decompositions. Finally, the "Conclusions and Closing Remarks" section concludes this article.

Some technical results, appendixes, and a video demonstrating our interface are available at https://rutgers.box.com/s/ms39u7z4a93lidv7av0zi8wadtwgdis9.

RELATED WORK

Efforts to deal with large graphs have mainly employed computational approaches to handle scale. Generally, computation and visualization appear to be treated as independent tasks. One approach is to develop scalable algorithmic tools that amplify users' understanding of the underlying data topologies at different levels of granularity. In general, macrograph views can in principle be obtained by some form of vertex or edge aggregation conceptually represented as hierarchy trees.[5,6] All previous techniques have algorithms with running time *substantially greater than linear* on the number of graph elements, making them not suitable for massive graph visualization. Graph Thumbnails, as a mechanism to identify and compare multiple graphs, are alone the subject of Yoghourdjian *et al.*'s[7] work. Their decomposition is based on a partition of the vertex set into vertex peel values. Ours is based on an iterative partition of the edge set into subgraphs, each of which is the subgraph induced by the highest core that appears on each iteration of the decomposition. Generation of graphs with a predefined core structure is the focus of Koevering *et al.*'s work.[8] Computational aspects of core-related graph decompositions and graph sparsification are studied in the works by Wang *et al.*,[9] Batagelj and Zaversnik,[10] and Arleo *et al.*[11] A new family of single parameter dense subgraph objectives is introduced in Veldt *et al.*'s work.[12] Some algorithmic principles for graph reachability in large graphs are proposed in Jin *et al.*'s work.[13] However, such approaches are not yet scalable to billion edge graphs. The ML approaches, such

as those described in the work of Hamilton *et al.*,[14] have been proposed to learn low-level embeddings of graphs.

Our approach differs from prior work in the sense that we look for efficient data traversal algorithms *via* exploration primitives that lend themselves to visual representations that amplify users' understanding of the internal graph structure for graphs that are too large to be visualized directly. Buckets, bushes, vicinities, buildings, waves, and edge fragments are examples of such subgraph primitives. Subgraph profiles that include properties, such as peel values, diversity, average weighted degree, size, volume, density, and height, are potentially useful to drive the search for "interesting" subgraphs in ML computational settings.

PROBLEM FORMULATION

Graph Edge Fragments and Waves

One approach to getting a sense of the topology of a graph begins with a choice of a starting set of vertices S_0 satisfying a property of interest P and marking these vertices as "explored." All the edges with at least one endpoint in S_0 are then traversed, deleting them from the graph, and adding them to the edge fragment associated with S_0. These edges then become the beginning part of the graph wave generated by S_0. If there are any edges with one endpoint not in S_0, we check whether those vertices not in S_0 still satisfy a stricter version of P in the remaining graph. If there are any such vertices, we continue exploring in parallel only from such vertices, adding incrementally the new-found edges into the current wave started by S_0, and deleting them from the existing graph. This process ends when all edges with exactly one endpoint in the current wave lead to vertices that do not satisfy P. This means that if there still remain "unvisited" edges, a new wave can be initiated by selecting another starting set of vertices satisfying a property of interest. We formalize the abovementioned process with the following definitions.

Definitions

We consider *undirected graphs* $G = (V, E)$ with vertex set $V(G)$ and edge set $E(G)$. We denote by n the number of vertices in V, m the number of edges in E, and the degree of a vertex $u \in V$ by deg(u). For any $U \subset V$, let $G(U)$ denote the subgraph of G induced by U. For the sake of completeness, we restate some definitions from Abello and Queyroi's work.[3]

Definition 1 (Peel Value): The peel value of a vertex $u \in V(G)$, denoted $\mathrm{peel}_G(u)$, is the largest $i \in$
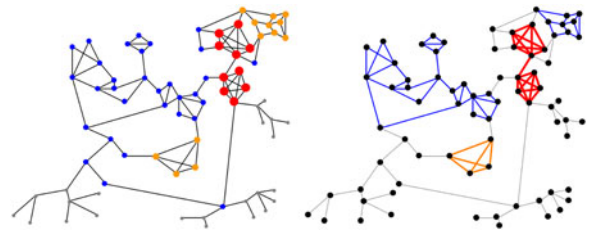

FIGURE 2. Left: Vertices are sized and colored according to their peel value: 1–small gray, 2–middle blue, 3–large orange, 4–extra-large red. Right: Edges are bolded and colored according to the iterative edge decomposition using the same size and color scale.

$[1, \deg(u)]$ such that u belongs to a subgraph of G of minimum degree i.

Definition 2 (Graph Core): The core of G, denoted $\mathrm{core}(G)$, sometimes also called the k-core of G, is the subgraph induced by the maximal subset of vertices of G whose peel value is maximum.

Definition 3: A graph F_k is a *fixed point* of degree peeling k, if $\mathrm{core}(F_k) = F_k$ and the peel value of each vertex in F_k is k.

The left of Figure 2 shows vertices in a small graph colored by their peel value. It also shows on the right edges colored according to the iterative edge decomposition from Abello and Queyroi's[3] work. This edge partition is obtained by removing the core edges (i.e., initially the red extra thick edges), updating the vertex peeling values, recoloring the affected edges, and identifying the next highest core edges and repeating until the whole graph is processed. When an edge gets removed from the graph during this iterative peeling process, it gets assigned its peel value. A Graph City (see the "What is a Graph City?" section) can be derived from this edge partition by mapping each connected edge color class into a building.

Next, we present some new definitions that generalize the notion of graph edge fragments and waves introduced in Abello and Nakhimovich's work.[2]

Definition 4: The *boundary* of a vertex set $S \subset V$ is defined as $\partial S = \{v \in V : \exists \, (u, v) \in E, \, u \in S, \, v \notin S\}$. The *proper boundary* of S, denoted $\partial_P S$, is the set of vertices in ∂S, which satisfy a desired property P restricted to the graph induced by $V \setminus S$. These definitions are extended, in a straightforward fashion, to a collection of disjoint sets by taking their union.

Definition 5: Given a vertex subset $S \subset V$, the edge fragment $\mathrm{frag}(S)$ generated by S is the set of edges (u, v), such that $u \in S$.

Definition 6: A *graph wave* $W(S_0, P)$ is the union of fragments in the sequence $(\text{frag}(S_j))_{j=0}^m$, with S_0 being the initial source seed set of vertices satisfying P and each subsequent $S_{j+1} = \partial_P(\bigcup_{i=0}^j S_i)$. We refer to S_0 as the initial source seed set for the wave.

Why are Graph Waves Useful Abstractions?

Graph waves generated by minimum degree seed sets were introduced in the work of Abello and Nakhimovich.[2] In this work, we use edge fragments to describe and visually represent waves' internal structure using a Meta-DAG.

Graph waves provide different "lenses" into the structure of a graph according to a particular property or substructure. Graph waves derived from a graph's degree distribution are essential for discovering the nonregular macrostructure of very large graphs, and at the same time, help isolate edge fragments with peculiar levels of regularity. For example, if k is the minimum degree of a graph G and if S_0 consists of all the vertices of degree k, and subsequent boundary sets are restricted to have degree strictly less than this minimum k, the corresponding wave is called a *minimum degree wave*.

Graph waves can be adapted to the particular properties of the boundary vertex sets and edge fragments being discovered during the algorithmic exploration of a large unknown graph topology. In this work, we only use waves generated based on vertex degree thresholds.

Meta-DAG Internal Structure of a Wave $W(S_0, P)$

Since the edges of a wave $W(S_0, P)$ are obtained by taking the union of edge fragments in the sequences $(\text{frag}(S_j))_{j=0}^m$, where $S_{j+1} = \partial_P(\bigcup_{i=0}^j S_i)$, the wave vertex set is an ordered partition of subsets $(S_0, S_1, \ldots S_m)$. We use the connected components of the subgraphs induced by each subset to define a Meta-DAG, where each macrovertex represents such connected components. Weighted directed metaedges $((C_{j,u}, C_{l,v}), W_{u,v})$ encode a nonempty set of edges $\{(x, y) : x \in C_{j,u}, y \in C_{l,v}\}$, where $C_{j,u}$ and $C_{l,v}$ are the connected components of S_j and S_l. The weight $W_{u,v}$ encodes the number of edges running between $C_{j,u}$ and $C_{l,v}$.

The spanning subgraph of this metagraph that consists of only those metaedges that connect components present in consecutive vertex sets, S_j, S_{j+1}, is called the *spanning Meta-DAG* of the wave [see Figure 1 (bottom center)]. It describes some of the most fundamental directed internal macroconnectivity of a wave and is expected to be substantially smaller than the wave itself. In the case of a tree, we have a simple description of its wave interpretation: its number of fragments is equal to the radius of the tree.

WHAT IS A GRAPH CITY?

We assume that the input is some ordered partition $(E_0, E_1, \ldots, E_z)$ of the edges of a graph $G = (V, E)$, where each E_i is edge maximal with respect to a predefined property. An efficient representation for such partitions is stored as a set of triples (source, target, and label $_i$). This assumption is justified since the edges of any graph G can be efficiently partitioned into edge-maximal subgraphs G_k, each of minimum degree at least k and average degree not more than $2\,k$.[3] We perform a wave decomposition as in the work of Abello and Nakhimovich.[2] This decomposition is logically stored in three parts: triples of (source, target, unique fragment ID), a mapping from unique fragment IDs to wave numbers, and a mapping from wave numbers to edge label$_i$ s. The vertex set V_k of each such subgraph G_k is partitioned into ordered sets $V_{k,j}$ that correspond precisely to minimum degree wave $W(S_0, P)$ (see the "Problem Formulation" section), where S_0 consists of the vertices of minimum degree, and the property P corresponds to boundary vertices of degree less than the degree of vertices in S_0.[2]

A *Graph City* is a 3-D representation of a given maximal edge graph partition $(E_0, E_1, \ldots, E_z)$, and such sample representations are shown in Figure 1 (middle left) and Figure 3(f)–(h). A Graph City consists of a floor plan of *buildings*, green patches of *bushes* that represent clusters of small buildings, and a weighted *street network*.

Graph City Buildings

For each edge-maximal subgraph G_k, we use its ordered sequence of vertex subsets $(V_{k,j})_{j=0}^h$, their "internal" edges, and those edges running between consecutive levels to create a visual representation for each such fixed point G_k, that resembles a building in a city with h floors (see Figure 1).

This building representation provides an alternative view of a fixed point $G_k = (V_k, E_k)$ that can be computed in time and space linearly dependent on the size of G_k, plus the complexity of finding and/or "describing" the initial subset V_{k0} of V_k. That is, the macrostructure of any fixed point can be described as a "building" whose internal structure is a Meta-DAG (see the "Problem Formulation" section), with macrovertices representing connected components of seed sets within each building floor [see Figure 1 (bottom center)].

A graph building with h floors representing a fixed point $G_k = (V_k, E_k)$ is completely determined by the *disjoint union of ordered edge fragments* specified by the partition of V_k into level sets $(V_{k,j})_{j=0}^h$. Our representation requires only $5\,h$ numbers. For each wave we specify two disk radii, the starting height, the color of a frustum, and a light intensity for a city night view.
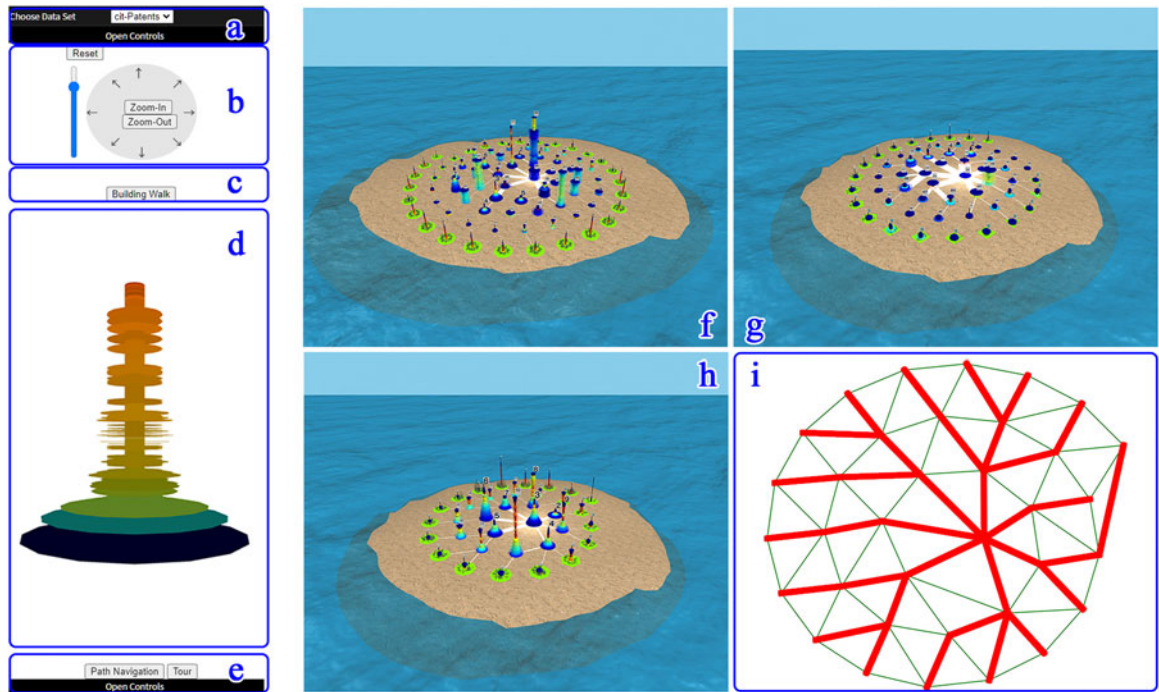
FIGURE 3. User controllers and sample city layouts. (a) Dataset selector. (b) Steering wheel. (c) Building walk controller. (d) Summary sculpture for the patent citation network. (e) Path navigation controller. (f) Friendster network city. (g) Movie phrase co-occurrence network city. (h) Patent citation network city. (i) Delaunay triangulation of the spiral building layout (thin green). The thick red street network corresponds to the white street network in the patent citation network city.

The "jth floor" of a building representing G_k corresponds to the subgraph induced by a subset of vertices $V_{k,j}$. Each floor is represented by two concentric disks, one above the other. The radius of the bottom disk encodes the number of vertices in the seed set of the starting fragment of the corresponding floor. The radius of the top disk encodes the total number of vertices besides the initial seed set vertices. The top disk is placed at the same height as the bottom disk of the next floor. A *frustum* between the bottom disks of adjacent floors is set to have volume encoding the total number of edges on the floor. Since the radii of the two disks are already determined, the height of the frustum is calculated from the desired volume.

A special fixed color map across the entire graph is used to encode the density of a variety of induced subgraphs. For example, the color of a frustum represents the density of the set of edges running between the corresponding two floors, and the same color map is used to highlight the density of the connected components within a floor, as shown in Figure 1.

A flag on top of a building displays summary information that includes the distribution of fixed point values of the corresponding bucket.

Handling buildings with lots of fragments does not present an issue because an intermediate structural level of granularity between fixed point buildings and fragments is provided by waves (see the "Problem Formulation" section). The floors of a building represent contiguous segments of fragments that satisfy some initial condition. They are characterized by their source layer of vertices and an ending layer of vertices whose unexplored neighbors violate a prespecified expansion condition. The beginning and end fragments of minimum degree waves specifically satisfy a bounded-degree condition.

Summary Graph City Sculpture

To provide an overview of the size distribution of fixed points in the set of buildings in a graph city, we use a *summary sculpture* [see Figure 3(d)]. This sculpture is obtained by considering each building as its set of connected components. All these edge maximal connected subgraphs with the same peel value and the same size are represented by cylinders encoding their frequency. Each cylinder of a particular peel value appears at a unique height in the sculpture. Larger cylinder radii correspond to a higher frequency of a

particular size. Our interface provides access to the location of all buildings in the graph city that correspond to a fixed point selected in the sculpture.

Vertex Diversity and Light Intensities

Graph Cities can be seen as 3-D representations of a coloring of the graph edges, where colors encode the edge peel values. This coloring partitions the edges adjacent to any particular vertex by their assigned color. The frequencies of these local colors for a vertex define a profile vector for that vertex. Following the work of Abello and Queyroi,[3] we compute this profile vector's *Shannon entropy* and use it as a measure of the *diversity* of the color pattern of the local edge coloring around each vertex. A higher diversity of a vertex is an indicator of a higher weighted level of participation of that vertex in a given edge partition. It is worth noting that diversity is a more expressive measure than degree. Specifically, very high-degree vertices can have very low diversity. We add diversity light intensities to the disks in the city sculpture to encode the average diversity of the vertices in the corresponding connected fixed point.

Graph City Interpretation

Since Graph City buildings represent connected fixed points, the following are natural questions.

1) *What do floors tell us about a building in a Graph City?* The number h of floors in a building (i.e., the number of waves) indicates a fixed point whose full exploration requires the sequential activation of h disjoint seed sets.

2) *What does a building volume represent?* It encodes the number of edges of the represented fixed point of degree peeling. A building with no enclosure represents a more localized topology (i.e., is a "tree-like" fixed point with only consecutive floor edges).

3) *How is the internal detailed structure of a building made accessible for user exploration?* It is represented by a Meta-DAG obtained by contracting the connected components of each fragment seed set. This Meta-DAG represents the connectivity between the connected components of all seed sets appearing in the waves.

GRAPH CITIES STREET NETWORK

Graph City Layout and Street Network

The size distribution and the peel value of the connected fixed point are used to generate a 2-D position for each building. This is done by bucketing the fixed point of a graph by size and peel value, and then mapping each such bucket to a 2-D location by following an Archimedean spiral. These connected fixed points are grouped together into buckets, according to the number of edges and peel value. Bucket i has fixed points of size s, such that $\log^{i-1}(n) < s \le \log^i(n)$, and a sub-bucket i_k is created for connected fixed points with peel value k in bucket i. We create a building for the largest fixed point in each sub-bucket B, and if $|B| > 1$, we create bushes (explained in the next section) for a representative selection of $\log(8 \times (|B| - 1))$ fixed points from B. In addition, for such buckets we also draw a *grass patch* as a green polygon with $\log(8 \times (|B| - 1)) + 2$ sides [see Figure 1 (middle right)]. From each sub-bucket, we display the largest building and a flag with the information of the shown building, with a glyph on the back for opening the local vicinity.

The Graph City *street network* is determined by the Delaunay triangulation of the building locations in the spiral layout [see Figure 3(i)]. The weight of a connection between two buildings is proportional to the intersection of the subgraph vertex sets represented by the two buildings. Graph-theoretically, the street network is determined by the intersection graph of the collection of vertex sets of the subgraphs $(G_k = (V_k, E_k))_{k=0}^z$, which in turn are determined by the given edge partition $(E_0, E_1, \ldots, E_z)$. When a particular building is selected by the user, a Euclidean spanning tree rooted at that building displays the corresponding street network, which is obtained by a Breadth First Search. If the building street network is disconnected, then we show a spanning forest instead.

Bushes and L-Systems

To provide a visual indication for the properties of fixed points in a sub-bucket (besides the largest one represented by the building), we sort the fixed points in a sub-bucket by size and uniformly select a few fixed points to draw as bushes [see Figure 1(middle right)].

We extended an implementation of a turtle graphics-based L-system interpreter to draw the underlying bush skeleton structure and then applied 2–3 iterations of a natural looking L-system, starting at evenly spaced lengths along the stem and branches.[a] Some sample bushes are accessible in an appendix gallery.[b]

---

[a][Online]. Available: https://github.com/andonutts/donatello
[b][Online]. Available: https://rutgers.app.box.com/s/ms39u7z4a93lidv7av0zi8wadtwgdis9

TABLE 1. Statistics for all datasets.

Dataset	CC	FP	CV	MW	MF	WT	RT
Friendster	1	29,692	304	212	3,279	501.45	11.82
Movies	38	2,044	3114	37	282	15.42	3
Patents	3,627	6,469	64	47	996	1.93	3.3

The last column shows the rendering time (RT) for the graph city in seconds.

RENDERING GRAPH CITIES

We use *Three.js*,[15] a 3-D Javascript library to create and display Graph Cities interactively in the web browser using WebGL. Each building in the Graph City consists of several floors (i.e., edge fragments). For each floor, we instantiate a *cylinder* shape geometry, where the top and bottom face radii, height, and color are chosen appropriately from the data (see the "What is a Graph City?" section), the number of balcony segments defaults to 6, with 3 windows per floor in the night view. All floors are generated in the material space, centered at the origin, and subsequently translated in the *Y* (up) direction to form a building in the world space. Each building is also translated in the *X*- and *Z*-directions to form the spiral layout (see the "Graph Cities Street Network" section). *Bushes* are generated with an L-system.[4]

On top of each building, a *flag* displaying summary information for that building (i.e., edge-maximal subgraph) is added. A *box* shape geometry is used for the flag, and a *cylinder* shape geometry is used for the mast. The length of the mast and the size of the flag are all determined from the dataset.

To highlight the *street network* representing data flow from one building to the remaining Graph City, we precompute the Delaunay triangulation for all the buildings in the Graph City. When a user left-clicks a particular building, we perform a Breadth First Search (BFS) from the root building to the rest, and display only those edges that are encountered in the search. The width of the edges is determined by the data. Rendering times for the Graph City for each dataset are summarized in the last column in Table 1.

TOURING A GRAPH CITY

The interactivity provided by *Three.js*[15] allows the user to explore different parts of a Graph City conveniently from a browser. Our interface offers users several entry points to different "regions" of the city (see Figure 4). A "region" of a city is characterized by a peel value and the bucket ID of all the buildings with that peel value and whose sizes are within a logarithmic factor. All these pairs (peel value, bucket ID) are displayed in a 2-D grid (i.e., a city map), where each point has associated either a circular glyph representing a big building in the central part of the city (i.e., downtown) or a spiral glyph representing multiple buildings of similar size that are located next to each other in green patches of bushes (i.e., vicinities).

Steering Wheel

A steering wheel and a path navigation controller assist users during Graph City explorations.

The steering wheel consists of three subcomponents [see Figure 3(b)]: a reset camera button, height slider, and directional wheel controller. Under the reset camera button, there is a height slider that the user can use to change the camera's vertical position. On the right-hand side of the slider, a wheel controller gives the user the ability to move freely in eight horizontal directions to reach any position on a horizontal plane. The combination of the steering wheel and the height slider allows a user to reach any geometric position in the city's free space.

Glyph Map

Each point in this grid map has an associated *glyph* that summarizes the subset of the corresponding connected fixed points (i.e., buildings) in the graph city layout [see Figure 1 (top)]. A circular glyph represents a unique building, and a spiral glyph summarizes a set of buildings whose represented subgraphs have sizes within a logarithmic factor.

The area of a circular glyph represents its corresponding building size in edges, and the color encodes its density. A clockwise sequence of spikes inside the circle corresponds to its floors (i.e., waves). For each spike, the lengths of two line segments starting from the center encode the vertex size of the seed set and the vertex size of the whole wave, respectively. The edge size and density are represented by its area and color. The starting angle, from the left to the first spike, encodes the ratio
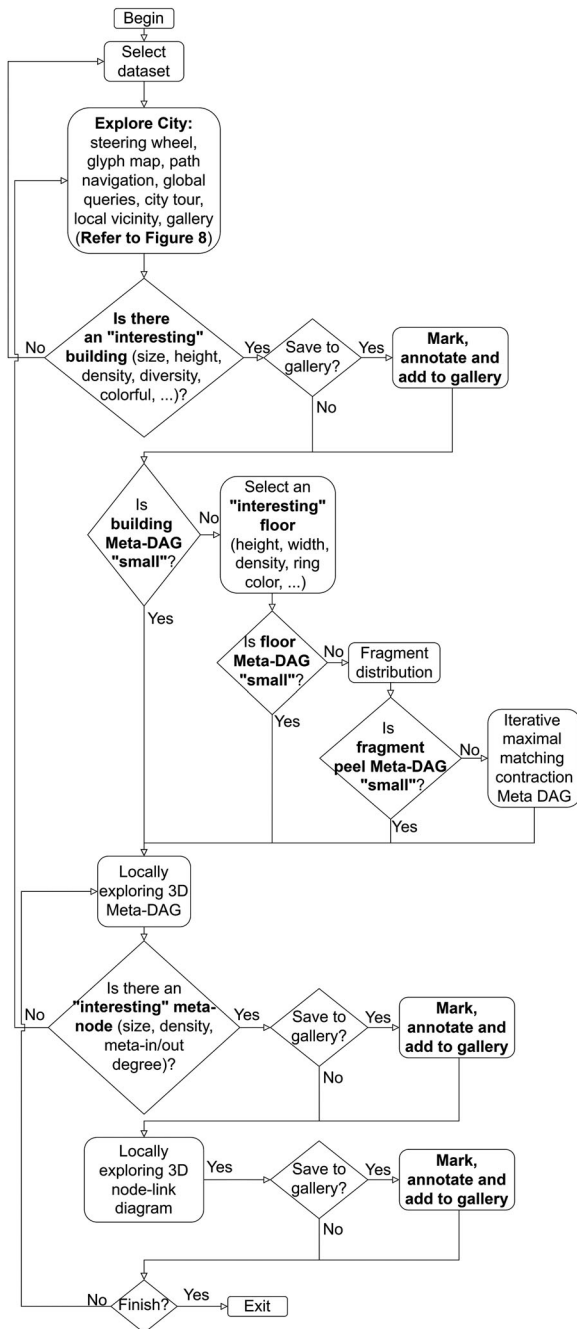
Path Navigation

The path navigation controller [see Figure 3(e)] opens up when users click the "Path Navigation" button. Users can left- and right-click a building in the city to set a source and a target, respectively. A cumulative exploration mode is provided, where the user can save the results of previous queries for further exploration.

Global Queries and City Tours

We use a navigation primitive to implement building walks for inspecting a building exterior and also for constructing a full city tour.

Graph-based notions, such as the subgraph's size, diameter, density, and diversity, have an "intuitive" interpretation in a Graph City. On the other hand, terms, such as *city tours*, *city commutes*, *building heights*, *landmarks*, *local vicinities*, and *city galleries*, can be given a graph theoretical interpretation that may be potentially useful to describe data properties.

Besides queries, the spiral layout of buildings provides a local geometry that can be used to explore the city. As an example, we compute a two-factor approximation of a minimum geometric TSP[16] on the Euclidean Delaunay triangulation graph.

Building Exploration

Hovering over one point on the city map highlights the corresponding buildings in the city and the corresponding frustum in the summary sculpture. This brings an infobox summarizing information about the corresponding subgraphs, which guides the user for further exploration. A left-click on a chosen fixed point address allows users to view more detailed information, such as floor size distribution, and perform an exterior building inspection to detect "interesting" floors.

Zooming Inside a Building

Left-clicking the glyph on the top-right corner of the selected building flag activates the next level of exploration based on the size of the corresponding Meta-DAG. Namely, if the Meta-DAG has a size less than a prespecified threshold, it is displayed via a 3-D force-directed layout, and its metanodes can be individually explored in a 2-D standard node-link diagram representation or in a 3-D level representation of its own local edge set decomposition. In the case that the Meta-DAG of the corresponding fixed point is too large, the user is offered a floor/fragment selection menu. In the extreme case that fragments themselves are fixed points too large to be displayed, we apply an

FIGURE 4. Graph City exploration scenarios.

between the peel value and the average degree of the corresponding building. For a spiral glyph, its area and color represent the overall size and density of the corresponding set of connected fixed points. The number of rounds it spins out encodes the number of connected fixed points contained. The starting angle, from the left to the outer end, encodes the overall ratio between the peel value and average degree.

iterative maximal matching contraction process to get a macroview of it that can then be explored further interactively. It is worth mentioning that in our extensive exploration of a variety of datasets, we have found only a couple of such cases. Theoretically speaking, this is possible if we have "very large" regular bipartite subgraphs embedded in our input graph. Needless to say that the detection of such very large "regular" special subgraphs is an interesting avenue for further theoretical investigations.

Upon entering a particular building, the view changes to highlight the finer scale internal connectivity for each edge-maximal subgraph (or fixed point) that constitutes a building. Our interactive visualization ultimately owes its speed to the hierarchical Graph Wave decomposition.[2] At the macroscopic level, our abstraction has a much smaller spatial complexity than the actual dataset, which makes rendering cheap. Rendering is never the bottleneck, as the user is only visualizing a subset of the data.

Local Vicinity
When a map glyph corresponds to more than one connected fixed point, a natural step is to expand the union of all the edges in these subgraphs as a mini Graph City that we call a local vicinity [see Figure 1 (bottom right)]. A local bucketization of edge size distribution is applied to generate the layout of the local vicinity. Users can use the same navigation tools, including the steering wheel, building walk, and path navigation, to explore the local vicinity, and click the glyph on the top-right corner of a flag to go inside the corresponding building in the local vicinity.

User Summarization Tools
We are designing summarization tools to aid users annotating those subgraph patterns that they find interesting, and place them in a city gallery with annotations. Currently, users can annotate subgraph patterns indicating their findings and add the corresponding subgraph patterns to a pattern gallery accessible from the top of the user interface. This gallery could potentially be used as labeled data for "making sense" of the overall large graph structure at different levels of granularity. Some sample "interesting" subgraphs are listed in an appendix.[c]

DESCRIPTION OF SAMPLE DATASETS
Our first dataset is the Friendster social network consisting of 65,608,366 nodes each representing a user and 1,806,067,135 edges representing "friendships" between them. This dataset was retrieved from the Stanford Large Dataset Collection (SNAP).[17] Our next dataset is a graph of phrases used in movie reviews. There are 218,052 nodes each representing a phrase and 115,050,370 edges, where each edge connects two phrases both used to describe the same movie in a review. This dataset was derived from the Internet Movie Database.[d] Our third dataset is a patent citation network, also from the work of Leskovec and Krevl.[17] There are 3,774,768 nodes, each representing a patent, and 16,518,947 edges, each linked to a cited patent.

Table 1 tabulates the number of connected components (CC), connected fixed points (FP), peel value of the core (CV), maximum number of waves among its fixed points (MW), maximum number of fragments among its fixed points (MF) for each dataset, and the corresponding wave decomposition time (WT) and rendering time (RT) in seconds. The spiral length is related to the total number of graph edges. The number of fragments in a building, i.e., the building's height, encodes the longest path length in the building's Meta-DAG.

MACHINE LEARNING CONNECTIONS
Catalogue of subgraph patterns: A useful outcome of user or computer explorations of any graph city will be a summary and an extensive catalogue of the subgraph patterns found. For this to be feasible, users must be provided with annotation and summarization tools that can keep track of their exploration trails. These patterns should be classified at least by size, density, frequency of occurrence, rarity, interest, and usefulness. To assess the efficacy of these tools, we are currently identifying a list of basic tasks that a user can perform in order to conduct a substantial number of user experiments. For example, can graph cities be used effectively on the shortest path approximations queries?

One of the ultimate goals of this work will be to have a descriptive semantic summary of the expected patterns that can be fed into a deep learning engine to learn to discriminate and find new patterns according to certain specified criteria. In our current

---

[c][Online]. Available: https://rutgers.box.com/s/ms39u7z4a93 lidv7av0zi8wadtwgdis9link

[d][Online]. Available: https://www.imdb.com/interfaces/

experimentation with a variety of datasets, the most naturally detected patterns include: tree forests, cliques, bicliques, and hierarchical compositions of these basic patterns.

Graph Cities and ML

In general, large graphs' processing techniques help for improving existing ML algorithms. Namely, for filtering, cleaning, enriching, and merging data before or during the training phase, large graphs are becoming at the forefront in ML due to the efficacy of novel representation learning methods, boosting prediction performance in a variety of tasks. Representation learning methods for graphs embed the nodes in a low-dimensional real-valued space in order to apply traditional ML methods. This work suggests Peel Value Similarity as a "learning" task that deserves further study.

Peel value similarity: Edge partitions, such as the ones proposed here, associate with each node a vector profile of peel values encoding its level of participation in the edge decomposition. It is "natural" to ask when pairs of nodes whose peel value vectors match in a proportion greater than expected play "similar roles" in the entire network. This approach offers peel value vector profiles as "novel" embeddings that can help predict "fake news" or "pandemic" spreading using the decomposition of an underlying social interaction network.

Graph-powered ML processes offer transparent management of data sources, novel application of algorithms to relational training datasets, storing and accessing of mixed predictive models, identification of the most reliable access patterns for providing predictions, and, of course, graph visualization as a useful tool to support analysis of multirelational data. There is a need to perform studies that compare the state-of-the-art representation learning methods with Big Graph Visualization algorithms using "readability-" based measures. It is not clear, what is the right coupling between a representation learning method and a "useful" visual abstraction.

CONCLUSIONS AND CLOSING REMARKS

The iterative edge decomposition partitions the edges of a graph into fixed points of degree peeling; they are in turn decomposed into graph waves and edge fragments. They provide mechanisms that may help assess the topological and statistical reasons that explain the emergence of a large class of bipartite graph-like patterns in very large graph datasets. These

include traffic analysis derived from transaction data, population migration, fraud detection, and viruses spreading.

We introduced the 3-D representations of fixed points based on their wave decomposition that resemble buildings in a city, hence Graph Cities. A spiral arrangement of the buildings is obtained from the size distribution of the fixed points. The Delaunay triangulation of the building locations determines the graph city street network. The size distribution of all the fixed points is summarized by a graph city sculpture (see the "What is a Graph City?" section). The city map provides an overall access mechanism to any bucket, building, wave, and fragment in our five-level edge decomposition.

Dense bipartite graph-like patterns have been proposed as an abstract formalization of "concepts" in the work of Wille.[18] Their identification in very large datasets has defied computation. Nevertheless, Graph Cities offer a promising approach to the efficient detection of a large subclass of these patterns in attributed graphs.

Streaming Graph Cities, composing global solutions from local ones, and generalizing our approach to Hypergraph Cities are tantalizing directions for future work.

REFERENCES

1. S. Sahu, A. Mhedhbi, S. Salihoglu, J. Lin, and M. T. Özsu, "The ubiquity of large graphs and surprising challenges of graph processing," in *Proc. VLDB Endowment*, vol. 11, no. 4, pp. 420–431, 2017.
2. J. Abello and D. Nakhimovich, "Graph waves," in *Proc. 3rd Int. Workshop Big Data Vis. Exploration Anal. EDBT/ICDT*, 2020.
3. J. Abello and F. Queyroi, "Fixed points of graph peeling," in *Proc. IEEE/ACM Int. Conf. Adv. in Soc. Netw. Anal. Mining*, 2013, pp. 256–263.

4. P. Prusinkiewicz and A. Lindenmayer, *The Algorithmic Beauty of Plants*. New York, NY, USA: Springer, 2012.

5. J. Abello, F. Van Ham, and N. Krishnan, "ASK-graphview: A large scale graph vis. system," *IEEE Trans. Vis. Comput. Graphics*, vol. 12, no. 5, pp. 669–676, Sep./Oct. 2006.

6. Y. Zhang, Y. Wang, and S. Parthasarathy, "Visualizing attributed graphs via terrain metaphor," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2017, pp. 1325–1334.

7. V. Yoghourdjian, T. Dwyer, K. Klein, K. Marriott, and M. Wybrow, "Graph thumbnails: Identifying and comparing multiple graphs at a glance," *IEEE Trans. Vis. Comput. Graphics*, vol. 24, no. 12, pp. 3081–3095, Dec. 2018.

8. K. Van Koevering, A. Benson, and J. Kleinberg, "Random graphs with prescribed k-core sequences: A new null model for network analysis," in *Proc. Web Conf.*, 2021, pp. 367–378.

9. N. Wang, D. Yu, H. Jin, C. Qian, X. Xie, and Q.-S. Hua, "Parallel algorithm for core maintenance in dynamic graphs," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst.*, 2017, pp. 2366–2371.

10. V. Batagelj and M. Zaveršnik, "Fast algorithms for determining core groups in social networks," *Adv. Data Anal. Class.*, vol. 5, no. 2, pp. 129–145, 2011.

11. A. Arleo, O.-H. Kwon, and K.-L. Ma, "GraphRay: Distributed pathfinder network scaling," in *Proc. IEEE 7th Symp. Large Data Anal. Vis.*, 2017, pp. 74–83.

12. N. Veldt, A. R. Benson, and J. Kleinberg, "The generalized mean densest subgraph problem," 2021, *arXiv:2106.00909*.

13. R. Jin, N. Ruan, S. Dey, and J. Y. Xu, "SCARAB: Scaling reachability computation on large graphs," in *Proc. Int. Conf. Manage. Data*, 2012, pp. 169–180.

14. W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications," *IEEE Data Eng. Bull.*, vol. 40, no. 3, pp. 52–74, 2017.

15. E. Angel and E. Haines, "An interactive introduction to WEBGL and three.JS," in *Proc. ACM SIGGRAPH Courses*, 2017, pp. 1–95.

16. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. Cambridge, MA, USA: MIT Press, 2001.

17. J. Leskovec and A. Krevl, "SNAP datasets: Stanford large network dataset collection," Jun. 2014. [Online]. Available: http://snap.stanford.edu/data

18. R. Wille, "Concept lattices and conceptual knowledge systems," *Comput. Math. Appl.*, vol. 23, no. 6–9, pp. 493–515, 1992.

19. J. Abello, D. Nakhimovich, C. Han, and M. Aanjaneya, "Graph cities: Their buildings, waves, and fragments," in *Proc. EDBT/ICDT Workshops*, 2021.

JAMES ABELLO is the director of the Rutgers Computer Science master program, Rutgers University, Piscataway, NJ, 08854, USA. He is a permanent member of DIMACS. Contact him at abello@dimacs.rutgers.edu.

HAOYANG ZHANG is currently working toward the Ph.D. degree with Rutgers Computer Science Department, Rutgers University, Piscataway, NJ, 08854, USA. He is the corresponding author of this article. Contact him at hz333@scarletmail.rutgers.edu.

DANIEL NAKHIMOVICH is currently working toward the Ph.D. degree with Rutgers Computer Science Department, Rutgers University, Piscataway, NJ, 08854, USA. Contact him at d.nak@cs.rutgers.edu.

CHENGGUIZI HAN is currently working toward the Ph.D. degree with Rutgers Computer Science Department, Rutgers University, Piscataway, NJ, 08854, USA. Contact her at chengguizi.han@rutgers.edu.

MRIDUL AANJANEYA is an assistant professor at Rutgers Computer Science Department, Rutgers University, Piscataway, NJ, 08854, USA. Contact him at mridul.aanjaneya@rutgers.edu.