Modern Graph Generators: A Survey

ANGELA BONIFATI, Lyon 1 University, France
IRENA HOLUBOVÁ, Charles University, Prague, Czech Republic
ARNAU PRAT-PÉREZ, DAMA-UPC, Universitat Politècnica de Catalunya, Spain
SHERIF SAKR, University of Tartu, Estonia

The abundance of interconnected data has fueled the design and implementation of graph generators reproducing real-world linking properties, or gauging the effectiveness of graph algorithms, techniques and applications manipulating these data. We consider graph generation across multiple subfields, such as graph databases, graph data mining, graph streaming and machine learning communities, alongside community detection, social networks and IoT communities. Despite the disparate requirements of modern graph generators throughout these communities, we analyze them under a common umbrella, reaching out the functionalities, the practical usage, and their supported operations. We argue that this classification is serving the need of providing scientists, researchers and practitioners with the right data generator at hand for their work.

This survey provides a comprehensive overview of the state-of-the-art modern graph generators by focusing on those that are pertinent and suitable for several data-intensive tasks. Finally, we discuss open challenges and missing requirements of current graph generators along with their future extensions to new emerging fields.

CCS Concepts: •**Mathematics of computing** → **Graph theory;** •**Theory of computation** → **Graph algorithms analysis;** •**Information systems** → *Graph-based database models;*

Additional Key Words and Phrases: Big Data management, graph data, generators, benchmarks, synthetic data, ...

ACM Reference Format:
A. Bonifati, I. Holubová, A. Prat-Pérez, S. Sakr, 2018. Modern Graph Generators: A Survey. *ACM CSUR* 0, 0, Article 0 (2016), 35 pages.
DOI: 0000001.0000001

1. INTRODUCTION

Graphs are ubiquitous data structures that span a wide array of scientific disciplines and are a subject of study in several subfields of computer science. Nowadays, due to the dawn of numerous tools and applications manipulating graphs, they are adopted as a rich data model in many data-intensive use cases involving disparate domain knowledge, mathematical foundations and computer science. Interconnected data is oftentimes used to encode domain-specific use cases, such as recommendation networks, social networks, protein-to-protein interactions, geolocation networks, and fraud detection analysis networks, to name a few.

In general, we can distinguish between two broad types of graph data sets: (1) a single large graph (possibly with several components), such as social networks or Linked

This work is supported by ...
Author's addresses: A. Bonifati, Université Claude Bernard Lyon 1, Liris CNRS Campus La Doua 23-25 Avenue Pierre de Coubertin, 69622 Villerubanne, France; I. Holubová, Department of Software Engineering, Faculty of Mathematics and Physics, Charles University, Malostranské nám. 25, 118 00 Praha 1, Czech Republic. A. Prat-Pérez, ...; S. Sakr, ...

DOI: 0000001.0000001

Data graphs, and (2) a large set of small graphs, such as chemical compounds[1] or linguistics syntax trees[2]. Naturally, the algorithms used in theses two classes differ a lot [Sakr and Pardede 2011]. In the former case we can search, e.g., for communities and their features or shortest paths, while in the latter case we usually query for supergraphs, subgraphs or graphs similar to a given graph pattern. In both the cases, as in the other fields, quite often the respective real-world graph data is not publicly available (or simply does not exist when a particular method for data manipulation is proposed). Even in the cases in which real data is abundant, many algorithms and techniques need to be tested on various order of magnitudes of the graph sizes thus leading to the inception of configurable graph generators that reproduce real-world graph properties and provide unique tuning opportunities for algorithms and tools handling such data.

In this survey, we provide a comprehensive overview of the state-of-the-art modern graph generators by focusing on those that are pertinent and suitable for data-intensive tasks. Our aim is to cover a wide range of currently popular areas of graph data processing. In particular, we consider graph generation in the areas of graph databases, graph data mining, graph streaming and machine learning communities, alongside community detection, social networks and IoT communities. Despite the disparate requirements of modern graph generators throughout these communities, we analyze them under a common umbrella, reaching out the functionalities, the practical usage, and the supported operations of graph generators. The reasons for this scope and classification are as follows:

(1) Despite the differences of the covered areas, the requirements for modern graph data generators can be similar in particular cases. Reusing or learning from tools in other fields can thus bring new opportunities for both researchers and practitioners.
(2) The selected classification is serving the need of providing scientists, researchers and practitioners with the right data generator at hand for their work.

To conclude the comparative study and provide a comprehensive view of the field, we also overview the most popular real-world data sets used in the respective covered areas and discuss open challenges and missing requirements of modern graph generators in view of identifying their future extensions to new emerging fields.

Contributions. Our survey revisits the representatives of modern graph data generators and summarizes their major features. The comprehensive review and analysis make this paper useful for motivating new graph data generation techniques as well as serving as a technical reference for selecting suitable solutions. In particular, the introductory categorization and comparative study enables the reader to quickly get his/her bearings in the field and identify a subset of generators of his/her interest. Since we do not limit the survey to a particular area of graph data management, the reader can get a broader scope in the field. Hence, while practitioners can find a solution in another previously unexpected area, researchers can identify new target areas or exploit successful results in new fields. Last but not least, we identify general open problems of graph data synthesis and indicate possible solutions to show that it still forms a challenging and promising research area.

Differences with prior surveys. To the best of our knowledge, our work is the first to survey the broad landscape of graph data generators spanning different data-intensive applications and targeting many computer science subfields. In particular, we cover

[1] https://pubchem.ncbi.nlm.nih.gov/
[2] https://catalog.ldc.upenn.edu/ldc99t42

graph database generators, graph processing generators, social networks and community detection generators, Semantic Web data generators, graph analytics generators, IoT, Telecommunication and graph streaming generators, and Machine Learning and graph mining generators. However, the literature is still lacking a comprehensive study of graph generators for many of the specific subfields mentioned above.

A limited subset of graph database generators, parallel and distributed graph processing generators, along with a few of the Semantic Web data generators presented in our survey have been discussed in a related book chapter [Bonifati et al. 2018] while cross-comparing them with respect to input, output, supported workload, data model and query language, along with the distinguished chokepoints. However, the provided classification is inherently database-oriented. In our work, we provide a more comprehensive and broader classification that serves the purpose of letting any researcher or practitioner interested in data generation to be able to make a better choice of the desired graph generator based on its functional and goal-driven features (such as the application domain, the supported operations and the key configuration options). Moreover, in contrast with [Bonifati et al. 2018], our work encompasses graph generators of several diverse communities, not limiting its scope to a few generators of the database and graph processing communities.

Graph generators matching graph patterns used in data mining have been studied in [Chakrabarti and Faloutsos 2006], focusing on mostly occurring patterns, such as power laws, size of graph diameters and community structure. The considered graph generators are compared in terms of graph type, degree distributions, exponentiality, diameter and community effects. We refer the reader to this survey for taxonomies involving these properties, whereas we provide here a functionality-driven taxonomy across all the categories of graph generators that we consider. We also point out that this survey is outdated as it does not consider the graph mining generators that fervently appeared in the last decade. We fill the gap of more recent social network and graph analytics generators respectively in Section 3.4 and 3.5.

Aggarwal and Subbian [Aggarwal and Subbian 2014] have surveyed the evolution of analysis in graphs, by primarily focusing on data mining maintenance methods and on analytical quantification and explanation of the changes of the underlying networks. A brief discussion on evolutionary network data generators is carried out in the paper. The data generation of evolutionary networks is based on Densification Power Law (DPL) and shrinking diameters [Leskovec et al. 2005d] and community-guided attachment properties [Leskovec et al. 2005d] is then considered, along with tackling Kronecker recursion with recursive tensor multiplication [Akoglu et al. 2008]. We refer the reader to this survey for evolutionary network generators and we further discuss the open challenges of evolving graph data in Section 5.

Outline. The rest of this paper is organized as follows: Section 2 provides the opening categorization and comparison of the generators. Section 3 provides an overview of the existing graph data generators and their main features in the frame of the proposed categories. To provide a complete overview, we list the most commonly used testing data sets for the particular categories in Section 4. In Section 5, we highlight some of the challenges and open problems of graph data synthesis before we conclude in Section 6.

2. CLASSIFICATION AND COMPARATIVE STUDY

In order to provide the reader with a quick preview of the generators and thus to enable finding the target solutions easily, we start the survey with a classification and comparative study of the existing tools. In general, there are various ways to classify them. We first provide an overview of the approaches used in related work after which we then introduce our approach. As mentioned in the Introduction, since this survey is unique in terms of scope and new tools related to Big Data processing, our classification and comparative strategy would differ as well.

The graph data generators can be classified according to various criteria. For example, [Chakrabarti et al. 2004] introduces two categories – degree-based and procedural generators. Given a degree distribution (typically following a power law), *degree-based generators* (e.g., Barabasi-Albert model [Barabasi and Albert 1999]) try to find a graph that matches it, but without giving any insights about the graph or trying to match other criteria (like, e.g., small diameter, eigenvalues etc.). On the other hand, *procedural generators* (e.g., R-MAT [Chakrabarti et al. 2004]) try to find simple mechanisms to generate graphs that match a property of the real graphs and, typically, the power law degree distribution.

Paper [Chakrabarti and Faloutsos 2006] introduces five categories of graph models that can be synthesized: (1) *random graph models* (e.g., Erdös-Rényi [Erdos and Renyi 1960]) generated by a random process, (2) *preferential attachment models* (e.g., Barabasi-Albert model) which try to model the power law from the preferential attachment viewpoint, (3) *optimization-based models* (e.g., HOT model [Carlson and Doyle 2000]) resulting from the idea that power laws can result from resource optimizations and tolerance to risks, (4) *tensor-based models* (e.g., R-MAT) targeting a trade-off between low number of model parameters and efficiency, and (5) *internet-specific models* corresponding to hybrids using ideas from the other categories in order to suit the specific features of the graphs.

The type of the generator can also be influenced by the benchmark involving it, whereas we can distinguish, e.g., *domain-specific* benchmarks, *application-specific* benchmarks, *workload-driven* benchmarks, *microbenchmarks* etc.

2.1. Classification

TODO: Add info to all three tables.

At first we classify the generators on the basis of the respective application domains or user communities. In particular we distinguish (1) general graphs, (2) Semantic Web, (3) graph databases, (4) social networks, (5) graph analytics, and (6) graph data steaming. The selected classes are not rigorously defined (e.g., they are not disjoint as we will show later), but they correspond to the currently most active research areas. Thus we believe that they form a natural first acquaintance for the reader.

In Tables I, II we overview the key characteristics of the data generators clustered according to the respective application domains.[3] In particular, we show:

- Characteristics of the domain – its *type* (fixed, specified using a schema, or extracted from input data) and the particular *target* domain, or, in case of a generic tool, the chosen sample domain,
- Characteristics of *read/update* operations (if provided), i.e., whether the set of operations is fixed/generated, if it involves operation mixes (i.e., sets/sequences of operations), or if templates of operations are supported.
- Key configuration options:
  - whether the generator deals only with structure, or also with *properties* of the graph,
  - supported types of *distributions* used for generating of the data,
  - *output format* of the produced graph, and

[3]"GDBs" stands for graph databases, "SNs" stands for social networks, "An." stands for graph analytics, "St." stands for graph streaming.

— whether the generator is *distributed* and thus enables more efficient data generation.

As we can see ... **TODO: Add a summary.**

2.2. Overlapping

As we have mentioned, the basic classification that we have used in this paper is based on a relatively vague division of the generators on the basis of the current application domains or research areas. In addition, some of the generators are either general or have features applicable in other domains. So the classes can overlap, as depicted in Table III.

TODO: Extend summary. For example, some social network graph generators such as the LDBC-SNB, S3G2 or LinkBench, can be used to test graph databases. In the case of the first two, even though they are designed not to be specific to any type of technology, they graph databases are their main target. Additionally, they also provide serializers for RDF, thus they can also be used to test RDF systems.

In the case of LinkBench, nothing prevents the user to load the generated graph in a graph database (Facebook uses a MySQL in that paper) to test a workload similar to Facebook and extend and complement it with more graphy queries like those in LDBC-SNB.

3. GRAPH DATA GENERATORS

In this section we explore in more detail the different graph data generators using the classification introduced before. For each category we briefly describe the key features of each of the representative examples. The aim is to provide the readers with a detailed look at each of the tools in the context of its competitors from the same domain.

3.1. General Graphs

We start by focusing on approaches that have been designed for dealing with the generation of general graph data that is not aimed at a particular application domain. Currently, there exists a number of tools which involve a kind of general graph data generator, such as gtools from projects nauty and Traces [McKay and Piperno 2014] or the Stanford GraphBase [Knuth 2008]. We, however, focus on primarily generating/benchmarking projects targeting the Big Data world.

Preferential Attachment. Barabasi and Albert [Barabasi and Albert 1999] introduced a graph generation model that relies on two main mechanisms. The first mechanism is continuously expanding the graphs by adding new vertices. The second mechanism is to preferentially attach the new vertices to the nodes/regions that are already well connected. So, in this approach, the generation of large graphs is governed by standard, robust self-organizing mechanisms that go beyond the characteristics of individual applications.

R-Mat. (*R*ecursive *Mat*rix) R-Mat is a procedural synthetic graph generator which is designed to generate power-law degree distributions [Chakrabarti et al. 2004]. The generator is recursive and uses only a small number of parameters. In principle, the strategy of this generator is to find simple mechanisms to generate graphs that match the properties of the real graphs. In particular, the design goals of R-Mat is to generate graphs that match the degree distributions, imitate a community structure and have a small diameter. R-Mat can generate weighted, directed and bipartite graphs.

GraphGen. For the purpose of testing the scalability of an indexing technique called FG-index [Cheng et al. 2007] on the size of the database of graphs, their average size and average density, the authors have also implemented a synthetic generator called

Table I. Key characteristics of the generators (part A)

		Domain		Operations		Configuration			
	Generator	**Type**	**Target / sample**	**Read**	**Update**	**Properties**	**Distributions**	**Output**	**Distributed**
General	[Barabási and Albert 1999]	fixed		Y	N	Y			N
	R-MAt	fixed		Y	N	Y	power-law		N
	GraphGen	fixed	N	Y	N	Y			N
	Graph 500	fixed	Various Domains	Y	N	Y	power-law		Y
	BTER	N	N	N		N	user-defined	?	Y
	Darwini	N	N	N		N	user-defined	?	Y
Semantic web	LUBM	fixed	university	fixed	N	Y	random (LCG)	RDF	N
	LBBM	extracted	Lehigh University BibTeX	N	N	Y	Monte Carlo	RDF	N
	UOBM	fixed	university	fixed	N	Y	random	RDF	N
	IIMB	fixed	movies	N	N	Y	random	RDF	N
	BSBM	fixed	e-commerce	fixed	N	Y	mostly normal	RDF, relational	N
	SP^{2}Bench	fixed	DBLP	fixed	N	Y	based on DBLP	RDF	N
	[Duan et al. 2011]	extracted	–	N	Y	N	–	RDF	N
	DBPSB	extracted	DBpedia	templates	N	Y	random	RDF	N
	LODIB	fixed	e-commerce	N	N	Y	44 types	RDF	N
	SIB	fixed	social network	mixes	mix	Y	from real-world data	RDF	N
	Geographica	fixed	OpenStreetMap	fixed + templates	N	Y	–	RDF	N
	WatDiv	schema-driven	user-defined	templates	N	Y	uniform, normal, Zipfian	RDF	N
	RBench	extracted	DBLP, Yago	templates	N	Y	from real-world data	RDF	N
	LDBC SPB	fixed	media	mixes	N	Y	power law, skewed values, value correlation	RDF	N
	LinkGen	schema-driven	user-defined	templates	Y	N	Gaussian, Zipfian	RDF	N

Table II. Key characteristics of the generators (part B)

		Domain		Operations		Configuration			
	Generator	**Type**	**Target / sample**	**Read**	**Update**	**Properties**	**Distributions**	**Output**	**Distributed**
GDBs	XGDBench	fixed	social network	fixed	Y	Y	power-law	MAG	Y
	gMark	schema-driven (internal schema)	user-defined	generated	N	Y	uniform, normal, zipfian	N-triples	N
	graphGen	pattern-driven (Cypher schema)	user-defined	–	–	Y	–	property graphs	N
SNs	[Barrett et al. 2009]	fixed	social network	N	N	Y	simulation-driven	–	–
	[Yao et al. 2011]	fixed	social network	N	N	N	power-law	–	–
	LinkBench	fixed	social network	Y	Y	Y	Facebook	–	–
	S3G2	fixed	social network	N	N	Y	Facebook	CSV, RDF	Y
	[Ali et al. 2014]	schema-driven	social-network	N	N	Y	power-law	CSV	N
	LDBC SNB	fixed	social network	generated	Y	Y	Facebook	CSV, RDF	Y
	[Nettleton 2016]	schema-driven	social network	N	N	Y	power-law	–	N
Ana.	HPC Scalable Graph Analysis	fixed		Y	N	N	uniform		N
	Graphalytics	extracted	social network	Y	N	power law			N
St.	S2Gen	schema-driven	Social Network	Y	N	N	user-defined	RDF	N
	RSPLab	schema-driven	Agnostic	Y	Y	N	user-defined	RDF	N

Table III. Overlapping of classes of generators

	Generator	**General**	**Semantic Web**	**Graph databases**	**Social networks**	**Analytics**	**Steaming**
General	[Barabási and Albert 1999]	x					
	R-MAt	x					
	GraphGen	x					
	Graph 500	x					
	BTER	x					
	Darwini	x					
Semantic web	LUBM		x				
	LBBM		x				
	UOBM		x				
	IIMB		x				
	BSBM		x				
	SP^2Bench		x				
	[Duan et al. 2011]		x				
	DBPSB		x				
	LODIB		x				
	SIB		x				
	Geographica		x				
	WatDiv		x				
	RBench		x				
	LDBC SPB		x				
	LinkGen		x				
GDBs	XGDBench			x	x		
	gMark		x	x	x		
	graphGen			x			
SNs	[Barrett et al. 2009]				x		
	[Yao et al. 2011]				x		
	LinkBench			x	x		
	S3G2		x	x	x		
	[Ali et al. 2014]				x		
	LDBC SNB		x	x	x		
	[Nettleton 2016]				x		
An.	HPC Scalable Graph Analysis					x	
	Graphalytics					x	
St.	S2Gen						x
	RSPLab						x

GraphGen [4]. It is based on the IBM synthetic data generation code for associations and sequential patterns[5]. GraphGen creates a collection of labeled, undirected and connected graphs which focuses on the performance evaluation of frequent subgraph mining algorithms and graph query processing algorithms. The result is represented as a list of graphs, each consisting of a list of nodes and a list of edges.

Graph 500 Benchmark. The Graph 500 Benchmark [Gra 2010] includes a scalable data generator which produces weighted, undirected graph as a list of edge tuples containing the label of start vertex and end vertex together with a weight that represents

[4]https://www.cse.ust.hk/graphgen/

[5]From 1996, no longer available at http://www.almaden.ibm.com/cs/projects/iis/hdb/Projects/datamining/mining.shtml

data assigned to the edge. The space of vertex labels is the set of integers beginning with 0. The input values required to describe the graph are (1) scale, i.e., the logarithm base two of the number of vertices, and (2) edge factor, i.e., the ratio of the graph's edge count to its vertex count (i.e., half the average degree of a vertex in the graph). The graph generator is a Kronecker generator similar to R-MAT. The data must not exhibit any locality, so in the final step the vertex labels and order of edges are randomly shuffled. The covered operations currently involve BFS; however, the authors intend to involve also two more types – optimization (single source shortest path) and edge-oriented (maximal independent set) – and five graph-related business areas: cybersecurity, medical informatics, data enrichment, social networks, and symbolic networks.

BTER. BTER (Block Two-Level Erdös-Rény) [Kolda et al. 2014] is a graph generator based on the creation of multiple Erdös-Rény graphs with different connection probabilities of which they are connected randomly between them. As the main feature, BTER is able to reproduce input degree distributions and average clustering coefficient per degree values. The generator starts by grouping the vertices by degree d, and forming groups of size $d + 1$ of nodes with degree d. Then, these groups are assigned an internal edge probability in order to match the observed average clustering coefficient of the nodes of such degree. Based on this probability, for each node, the excess degree (i.e, the degree that in expectation will not be realized internally in the group) is computed and used to connect nodes from different groups at random. The authors describe a highly scalable MapReduce based implementation that is capable of generating large graphs (with billions of nodes) in a reasonable amounts of time.

Darwini. Darwini [Edunov et al. 2016] is an extension of BTER designed to run on Vertex Centric computing frameworks like Pregel [Malewicz et al. 2010] or Apache Giraph [Ching et al. 2015], with the additional feature that it is more accurate when reproducing the clustering coefficient of the input graph. Instead of just focusing on the average clustering coefficient for each degree, Darwini is able to model the clustering coefficient distribution per degree. It achieves this by grouping the vertices of the graph into buckets by the expected number of closed triangles that they need to close in order to attain the expected clustering coefficient, which is sampled from the input distributions. Then, the vertices in each bucket are connected randomly with a probability that would produce the expected desired number of triangles for such bucket. Then, as in BTER, the excess degree is used to connect the different buckets. The authors report that Darwini is able to generate graphs with billions and even trillions of edges.

3.2. Semantic Web

With the dawn of the concept of Linked Data it is a natural development that there would emerge respective benchmarks involving both real-world data sets and synthetic data sets with real-world characteristics. The used data sets correspond to RDF representation of relational-like data [Guo et al. 2005; Bizer and Schultz 2009], social network-like data [Schmidt et al. 2010], or specific and significantly more complex data structures such as biological data [Wu et al. 2014b]. In this section, we provide an overview of benchmarking systems involving a kind of graph-based RDF data generator or data modifier.

LUBM. The use-case driven Lehigh University Benchmark (LUBM)[6] considers the university domain. The ontology defines 43 classes and 32 properties, including 25 object properties and 7 datatype properties [Guo et al. 2005]. The LUBM benchmark also

[6]http://swat.cse.lehigh.edu/projects/lubm/

provides 14 test queries. The authors focus on *extensional* queries, i.e., queries about the instance data over ontologies, as an opposite to *intentional* queries (i.e., queries about classes and properties). The Univ-Bench Artificial (UBA) data generator features random and repeatable data generation (exploiting classical linear congruential generator, LCG, of numbers). In particular, data generated by the tool are exactly repeatable with respect to universities (assigning them zero-based indexes, i.e., the first university is named University0 and so on). The generator allows a user to specify the seed for random number generation, the number of universities, and the starting index of the universities.

An extension of LUBM, the Lehigh BibTeX Benchmark (LBBM) [Wang et al. 2005], enables generating synthetic data for different ontologies. The generation is divided into two phases: (1) the property-discovering phase, and (2) the data generation phase. The authors present a probabilistic model that, given representative data of some domain, can capture the properties of the data and generate synthetic data that has similar properties. A Monte Carlo algorithm is used to generate synthetic data. The approach is demonstrated on the Lehigh University BibTeX ontology which consists of 28 classes and 80 properties. 12 test queries were designed for the benchmark data. Another extension of LUBM, the University Ontology Benchmark (UOBM)[7], focuses on two aspects: (1) usage of all constructs of OWL Lite and OWL DL [owl 2004] and (2) lack of necessary links between the generated data which thus form isolated graphs [Ma et al. 2006]. In the former case the original ontology is replaced by the two types of extended versions from which the user can choose. In the latter case cross-university and cross-department links are added to create a more complex graph.

IIMB. Contrary to the previous work, [Ferrara et al. 2008] proposed the ISLab Instance Matching Benchmark (IIMB)[8] for the problem of instance matching defined as follows: Given two instances i_1 and i_2, belonging to the same ontology or to different ontologies, instance matching is defined as a function $Im(i_1, i_2) \rightarrow \{0, 1\}$, where 1 denotes the fact that i_1 and i_2 are referred to the same real-world object and 0 denotes the fact that i_1 and i_2 are referred to different objects. It targets the domain of movie data which contains 15 named classes, 5 object properties and 13 datatype properties. The data are extracted from IMDb[9]. The data generator corresponds to a data modifier which simulates differences between the data. In particular it involves data value differences (such as typographical errors or usage of different standard formats, e.g., for names), structural heterogeneity (represented by different levels of depth for properties representation, different aggregation criteria for properties representation, or missing values specification) and logical heterogeneity (such as instantiation on different subclasses of the same superclass or instantiation on disjoint classes).

BSBM. The Berlin SPARQL Benchmark (BSBM)[10], models an e-commerce domain where types Product, Offer and Vendor are used to model the relationships between products and the vendors offering them, while types Person and Review are used to model the relationship between users and product reviews these users write [Bizer and Schultz 2009]. The benchmark comes with 12 queries and 2 query mixes (sequences of the 12 queries) emulating the search and navigation pattern of a consumer looking for a product. The data generator supports the creation of arbitrarily large datasets using the number of products n as scale factor which further influences also other characteristics of the data, such as, e.g., the depth of type hierarchy of products

[7]https://www.cs.ox.ac.uk/isg/tools/UOBMGenerator/
[8]http://www.ics.forth.gr/isl/BenchmarksTutorial/
[9]http://www.imdb.com/
[10]http://wifo5-03.informatik.uni-mannheim.de/bizer/berlinsparqlbenchmark/

(defined as $d = round(log_{10}(n))/2 + 1$), branching factor ($bfr = 2 \times round(log_{10}(n))$), the number of product features (having $lowerBound = 35 \times i/(d \times (d+1)/2 - 1)$ and $upperBound = 75 \times i/(d \times (d+1)/2 - 1)$) etc. BSBM can output two representations – an RDF representation and a purely relational representation. Thus, BSBM also defines an SQL [sql 2008] representation of the queries. This allows comparison of SPARQL [Prud'hommeaux and Seaborne 2008] results to be compared against the performance of traditional RDBMSs.

SP^{2}Bench. The SP^{2}Bench[11] is a language-specific benchmark [Schmidt et al. 2010] which uses the DBLP [dbl 2016] as a domain for the dataset. The generated documents mirror key characteristics and distributions found in the original DBLP database. The data mimics natural correlations between entities, such as power law distributions (found in the citation system or the distribution of publications among authors) and limited growth curves (e.g., the increasing number of venues and publications over time). All random functions of the generator base on a fixed seed which makes data generation deterministic. SP^{2}Bench is accompanied by 12 queries covering different operator constellations, RDF access paths, typical RDF constructs etc.

Data Coherence. A comparison of 4 RDF benchmarks (namely TPC-H [TPC 2016] data expressed in RDF, LUBM, BSBM, and SP^{2}Bench) and 6 real-wold data sets (such as, e.g., DBpedia [Bizer et al. 2009], the Barton Libraries Dataset [Abadi et al. 2007] or WordNet [Miller 1995]) has been reported by [Duan et al. 2011]. The authors focus mainly on the *structuredness* (*coherence*) of each benchmark dataset claiming that a primitive metric (e.g., the number of triples or the average in/outdegree) quantifies only some specific aspect of each dataset. The level of structuredness of a dataset D with respect to a type T is determined by how well the instance data in D conforms to type T. The type system is extracted from the data set by looking for triples whose property is `http://www.w3.org/1999/02/22-rdf-syntax-ns#type` and extracting type T from their object. Properties of T are determined as the union of all the properties that the instances of type T have. The structuredness is then expressed as a weighted sum of share of set properties of each type, whereas higher weights are assigned to types with more instances. The authors show that the structuredness of the chosen benchmarks is fixed, whereas real-world RDF datasets lie in currently untested parts of the spectrum. As a consequence, they introduce a new benchmark that accepts as input any dataset along with a desired level of structuredness and size (smaller than the size of the original data), and uses the input dataset as a seed to produce a subset of the original data with the indicated size and structuredness. In addition they show that structuredness and size mutually influence each other.

DBPSB. DBpedia SPARQL Benchmark (DBPSB)[12] proposed at the University of Leipzig is based on queries that were issued by humans and applications against existing RDF data [Morsey et al. 2011; Morsey et al. 2012]. In addition, the authors argue that benchmarks like LUBM, BSBM, or SP^{2}Bench resemble relational database benchmarks involving relational data structures with few and homogeneously structured classes, whereas RDF knowledge bases are increasingly heterogeneous (e.g., DBpedia version 3.6, for example, contains 289,016 classes of which 275 classes belong to the DBpedia ontology) and various datatypes and object references of different types are used in property values. Hence, they propose a generic SPARQL benchmark creation methodology based on a flexible data generation mimicking an input data source, query-log mining, clustering, and SPARQL feature analysis. The proposed dataset cre-

[11] http://dbis.informatik.uni-freiburg.de/forschung/projekte/SP2B/
[12] http://aksw.org/Projects/DBPSB.html

ation process starts with an input dataset. Datasets of multiple sizes of the original data are created by duplicating all triples and changing their namespaces. For generating smaller datasets, an appropriate fraction of all triples is selected randomly or by sampling across classes in the dataset. The goal of the query analysis and clustering is to detect prototypical queries on the basis of their frequent usage and similarity. The methodology is applied on the DBpedia SPARQL endpoint and a set of 25 SPARQL query templates is derived, which cover most commonly used SPARQL features and are used to generate the actual benchmark queries by parametrization.

LODIB. The Linked Open Data Integration Benchmark (LODIB)[13] has been designed with the aim of reflecting the real-world heterogeneities that exist on the Web of Data in order to enable testing of Linked Data translation systems [Rivero et al. 2012]. It provides a catalogue of 15 data translation patterns (e.g., rename class, remove language tag etc.), each of which is a common data translation problem in the context of Linked Data. The benchmark provides a data generator that produces three different synthetic data sets that need to be translated by the system under test into a single target vocabulary. They reflect the pattern distribution in analyzed 84 data translation examples from the LOD Cloud. The data sets reflect the same e-commerce scenario used for BSBM.

SIB. The developers of the Social Network Intelligence BenchMark (SIB)[14] based the design of their benchmark based on the claim that existing benchmarks are limited in representing the real RDF datasets and are mostly relational-like. Hence, they propose a benchmark for challenging query processing over real graphs [Boncz et al. 2013]. It simulates an RDF backend of a social network site, in which users and their interactions form a social graph of social activities such as writing posts, posting comments, creating/managing groups, etc. The distribution of generated data on each relation conforms to the data distribution analyzed over real-world social networks. Association rules are included in order to convey the real-world data correlation into synthetic data. The generated data is linked with the RDF datasets from DBpedia. The benchmark specification contains 3 query mixes – interactive, update, and analysis – expressed in SPARQL 1.1 Working Draft.

Geographica. The Geographica benchmark[15] has been designed to target the area of geospatial data [Garbis et al. 2013] and respective SPARQL extensions GeoSPARQL [Battle and Kolas 2012] and stSPARQL [Koubarakis and Kyzirakos 2010]. The benchmark involves a real-world workload based on publicly available linked data sets covering a range of geometry types (e.g., points, lines, polygons) and a synthetic workload. In the former case there is a (1) a micro benchmark that tests primitive spatial functions (involving 29 queries) and (2) macro benchmark that tests the performance of RDF stores in typical application scenarios like reverse geocoding or map search and browsing (consisting of 11 queries). In the latter case of a synthetic workload the generator produces synthetic datasets of various sizes that corresponds to an ontology based on OpenStreetMap (i.e., states in a country, land ownership, roads and points of interest) and instantiates query templates. The spatial extent of the land ownership dataset constitutes a uniform grid of $n \times n$ hexagons, whereas the size of each dataset is given relatively to n. The synthetic workload generator produces SPARQL queries corresponding to spatial selection and spatial joins by instantiating 2 query templates.

[13] http://lodib.wbsg.de/
[14] https://www.w3.org/wiki/Social_Network_Intelligence_BenchMark
[15] http://geographica.di.uoa.gr/

WatDiv. The Waterloo SPARQL Diversity Test Suite (WatDiv)[16] developed at the University of Waterloo provides stress testing tools to address the observation that existing SPARQL benchmarks are not suitable for testing systems for diverse queries and varied workloads [Aluç et al. 2014]. The benchmark introduces two classes of query features – structural and data-driven – and performs an in-depth analysis on existing SPARQL benchmarks (LUBM, BSBM, SP^{2}Bench, and DBPSB) using the two classes of query features. The structural features involve triple pattern count, join vertex count, join vertex degree, and join vertex count. The data-driven features involve result cardinality and several types of selectivity. The analysis of the four benchmarks reveals that they are not sufficiently diverse to test the strengths and weaknesses of different physical design choices employed by RDF systems. The proposed solution, WatDiv, involves (1) a data generator which generates scalable datasets according to the WatDiv schema, (2) a query template generator which traverses the WatDiv schema and generates a diverse set of query templates, (3) a query generator which instantiates the templates with actual RDF terms from the dataset, and (4) a feature extractor which computes the structural and data-driven features of the generated data and workload. For the study in the paper the authors generated 12,500 test queries from 125 query templates.

RBench. RBench [Qiao and Özsoyoğlu 2015] is an application-specific benchmark which takes any RDF dataset as a template and generates a set of synthetic datasets with similar characteristics, required size scaling factor s and (node) degree scaling factor d. A generated benchmark dataset is considered similar to the given dataset if their values for the dataset evaluation metrics and query evaluation times for different techniques are similar. Three evaluation metrics are utilized for this purpose: dataset coherence (i.e., a measure how uniformly predicates are distributed among the same type/class), relationship specialty (i.e., the number of occurrences of the same predicate associated with each resource), and literal diversity. A query generation process is proposed to generate 5 types of queries (node queries, edge queries, star queries, path queries, and subgraph queries) for any generated data. The benchmark project FEASIBLE [Saleem et al. 2015] is also an application-specific benchmark; however, contrary to RBench, it is able to generate benchmarks from a set of queries (in particular from query logs) by selecting prototypical queries of a user-defined size from the input set of queries.

LDBC. The Linked Data Benchmark Council[17] (LDBC) [Angles et al. 2014] is a result of a (closed) EU project that brought together a community of academic researchers and industry that had the main objective of developing an open source, yet industrial grade benchmarks for graph and RDF databases. In the Semantic Web domain, the project released the Semantic Publishing Benchmark (SPB) [spb 2015] that has been inspired by the Media/Publishing industry (namely BBC[18]). The application scenario of this benchmark considers a media or a publishing organization that deals with large volumes of streaming content, namely news, articles or "media assets". This content is enriched with metadata that describes it and links it to reference knowledge – taxonomies and databases that include relevant concepts, entities and factual information. The SPB data generator produces scalable in size synthetic large data. Synthetic data consists of a large number of annotations of media assets that refer entities found in reference datasets. The data generator models three types of relations in produced synthetic data: clustering of data, correlations of entities, and random tagging

[16]http://dsg.uwaterloo.ca/watdiv/
[17]http://ldbcouncil.org/industry/organization/origins
[18]http://www.bbc.com/

of entities. Two workloads are provided: (1) basic, involving an interactive query-mix querying the relations between entities in reference data, and (2) advanced, consisting of interactive and analytical query-mixes. The LDBC has designed two other benchmarks: the Social Network Benchmark (SNB) [Erling et al. 2015] for the social network domain (see Section 3.4) and Graphalytics [Iosup et al. 2016] for the analytics domain (see Section 3.5).

LinkGen. LinkGen is a synthetic linked data generator that has been designed to generate RDF datasets for a given vocabulary [Joshi et al. 2016]. The generator is designed to receive a vocabulary as an input and supports two statistical distribution techniques for generating entities: Gaussian distribution and Zipf's power-law distribution. LinkGen can augment the generated data with inconsistent and noisy data such as writing two conflicting values for a given datatype property, adding triples with syntactic errors, adding wrong statements by assigning them with invalid domain and creating instances with no type information. The generator is also designed with an option to inter-link the generated instances with real ones given that the user provides entities from real datasets. The datasets can be generated in any of of two modes: on-disk and streaming.

3.3. Graph Databases

Currently there exists a number of papers which compare the efficiency of graph databases with regards to distinct use cases, such as the community detection problem [Beis et al. 2015], social tagging systems [Giatsoglou et al. 2011], graph traversal [Ciglan et al. 2012], graph pattern matching [Pobiedina et al. 2014], data provenance [Vicknair et al. 2010], or even several distinct use cases [Grossniklaus et al. 2013]. However, the number of graph data generators and benchmarks that have been designed specifically for graph databases is relatively small. Either a general graph generator is used for benchmarking graph databases, such as, e.g., the HPC Scalable Graph Analysis Benchmark [Dominguez-Sal et al. 2010] or the graph DBMS benchmarking tools are designed in a more general scope. Hence it is questionable whether a benchmark that is targeted specifically for graph databases is necessary. [Dominguez-Sal et al. 2011] discussed this question and related topics. On the basis of a review of applications of graph databases (namely social network analysis, proteomics or genetic interactions, recommendation systems, and travel planning and routing), the authors analyzed and discussed the characteristics of the graphs that appear in such applications and how they could influence benchmarking, different types of operations used in these applications and the characteristics of the evaluation setup of the benchmark. In this section, we focus on graph data generators and benchmarks that have been primarily targeting graph DBMSs.

XGDBench. XGDBench [Dayarathna and Suzumura 2014] is an extensible graph database benchmarking platform for cloud computing systems. Its intent is to automate the process of graph database benchmarking in the cloud by focusing on a graph database application for social networking services. It extends the Yahoo! Cloud Multiplicative Attribute (MAG) Graph Serving Benchmark (YCSB) [Cooper et al. 2010] and is composed of a workload generator client and a package of standard workloads that cover interesting parts of the performance space. The workload generator of YCSB supports definition of new workload types. XGDBench involves basic operations such as read / insert / update / delete an attribute, loading of the list of neighbors and BFS traversal. Using the generators, 7 workloads are created, such as update heavy, read mostly, short range scan, traverse heavy etc. The data model of XGDBench is a simplified version of the Multiplicative Attribute Graph (MAG) [Kim and Leskovec 2010] model, a synthetic graph model for attribute graphs which models the interactions between the network structure and the node attributes. The generated graphs are thus in MAG format, with power-law degree distribution closely simulating real-world social networks. The simplified MAG algorithm accepts the number of vertices of the generated graph, the number of attributes per vertex, a threshold value for random initialization of attributes, an edge affinity threshold value that determines whether there is an edge between two vertices, and an affinity matrix. It has been proven that MAG generates graphs with both analytically tractable and statistically interesting properties. A multi-threaded version of the graph generator that generates large graphs on multi-core systems faster was implemented too.

gMark. gMark [Bagan et al. 2016] is a schema-driven, domain-independent and query language-independent graph instance and query workload generator. It leverages a schema definition, called a *graph configuration*, which includes the enumeration of predicates (i.e., edge labels) and node types (i.e., node labels) occurring in the data, along with their properties in generated instances (occurrence constraints, degree distribution, etc.). *Query workload configuration* describes parameters of the queries to be generated (e.g., number of queries, arity, shape, selectivity etc.). The authors prove that given a graph configuration G, deciding whether there exists a graph satisfying G is NP-complete. And, that given a query workload configuration Q, deciding whether there exists a query workload satisfying Q is NP-complete. Hence, gMark follows a heuristic strategy in the generation – it attempts to achieve the exact values of the parameters and it may decide to relax some of them in order to obtain linear running time. gMark generates graphs under the form of N-triples and query workloads in four concrete syntaxes, including Cypher[19], SPARQL, SQL and LogicQL.

GraphGen. GraphAware GraphGen[20] is a graph generation engine based on Neo4j's[21] query language Cypher [Gra 2015]. It creates nodes and relationships based on a schema definition expressed in Cypher, and it can also generate property values on both nodes and edges. As such, GraphGen is a precursor of property graphs generators. The resulting graph can be exported to several formats (namely GraphJson[22] and CypherQueries) or loaded directly to a DBMS. However, it is very likely that it is not maintained anymore due to the lack of available updates.

3.4. Social Networks

On-line social networks, like Facebook, Twitter, or LinkedIn, have become a phenomenon used by billions of people every day and thus providing extremely useful information for various domains. However, an analysis of such type of graph has to cope with two problems: (1) availability of the data and (2) privacy of the data. Hence, data generators which provide realistic synthetic social network graphs are in a great demand.

In general, any analysis of social networks identify their various specific features [Chakrabarti and Faloutsos 2006]. For example, a social graph has high *clustering coefficient*, i.e. the degree of transitivity of a graph. Or, its diameter, i.e. the minimum number of hops in which some fraction (e.g., 90%) of all connected pairs of nodes can reach each other, is usually low due to weak ties joining faraway cliques.

Another important aspect of social networks is the community effect. A detailed study of structure of communities in 70 real-world networks is provided, e.g., in [Leskovec et al. 2008]. [Prat-Pérez and Dominguez-Sal 2014] analyzed the structure

[19]https://neo4j.com/developer/cypher-query-language/
[20]http://graphgen.graphaware.com/
[21]https://neo4j.com/
[22]https://github.com/GraphAlchemist/GraphJSON/wiki/GraphJSON

of communities (clustering coefficient, triangle participation ratio, bridges, diameter, conductance and size) in both real-world graphs and outputs of existing graph generators LFR [Lancichinetti et al. 2008] and the LDBC-SNB [Erling et al. 2015]. They discover that communities found in different graphs follow quite similar distributions and that communities in a single graph have diverse nature and are difficult to fit with a single model.

The existing social network generators try to reproduce different aspects of the generated network. They can be categorized into statistical and agent-based. *Statistical approaches* [Lancichinetti et al. 2008; Yao et al. 2011; Armstrong et al. 2013; Pham et al. 2013; Ali et al. 2014; Erling et al. 2015; Nettleton 2016] focused on reproducing aspects of the network. In *agent-based approaches* [Barrett et al. 2009; Bernstein and O'Brien 2013] the networks are constructed by directly simulating the agents' social choices.

Realistic Social Network. [Barrett et al. 2009] focused on the construction of realistic social networks using a combination of public and private data sets and large-scale agent based techniques. The process works as follows: In the first step it creates a synthetic population by integrating databases from commercial and public sources. In the second step, a set of activity templates are determined. Each synthetic individual is assigned a 24-hour activity sequence including geolocations for each activity. To demonstrate the approach, the authors develop a synthetic population for the United States that models every individual in the population. The synthetic population is a set of geographically located people and households. Household structure and demographics are derived from U.S. Census data. The activity templates are based on several thousand responses to an activity or time-use survey. Demographic information for each person and location, a minute-by-minute schedule of each person's activities, and the locations where these activities take place is generated by a combination of simulation and data fusion techniques. This information is captured by a dynamic social contact network. Similar methods for agent-based strategies have been reported in [Bernstein and O'Brien 2013].

Linkage vs. Activity Graphs. [Yao et al. 2011] distinguished between two types of social network graphs – the *linkage graph*, where nodes stand for the people in the social network and edges are their friendship links, and the *activity graph*, where nodes also stand for the people but edges stand for their interactions. On the basis of the analysis of Flickr[23] social links and Epinions[24] network of user interactions, the authors discover that they both exhibit power-law degree distribution, high clustering coefficient (community structure), and small diameter; also regarding the dynamic properties they both follow the densification law and have relatively stable clustering coefficient over time. However, the authors do not observe diameter shrinking in opinions activity graph and there is a difference in degree correlation (how frequently nodes with similar degrees connect to each other). Namely linkage graphs have positive degree correlation whereas activity graphs show neutral degree correlation. With regards to the findings, the proposed generator focusses on linkage graphs with positive degree correlation. For this purpose it extends the forest fire spreading process algorithm [Leskovec et al. 2005c] with link symmetry. It has two parameters – the *burning probability* P_b which is in charge of the burning process, and the *symmetry probability* P_s which indicates backward linking from old nodes to new ones. P_b controls a BFS-based forward burning process. The fire burns increasingly fiercely with

[23]https://www.flickr.com/
[24]http://www.epinions.com/

P_b approaching 1. Meanwhile, P_s adds fuel to the fire as it brings more links. It gives chances for big nodes to connect back to big nodes.

LinkBench. The LinkBench benchmark [Armstrong et al. 2013] has been designed to predict the performance of a database when used for persistent storage of Facebook's production data. The benchmark considers true Big Data and related problems with sharding, replication etc. The social graph at Facebook comprises objects (nodes with IDs, version, timestamp and data) and associations (directed edges, pairs of node IDs, with visibility, timestamp and data). The size of the target graph is the number of nodes. Graph edges are generated concurrently with graph nodes during bulk loading. The node ID space is divided into chunks based on the ID of the source node which are processed in parallel. The edges of the graph are generated in accordance with the results of analysing real-world Facebook data (such as outdegree distribution). A workload corresponding to 10 graph operations (such as insert object, count the number of associations etc.) and their respective characteristics over the real-world data is generated for the synthetic data.

S3G2. The Scalable Structure-correlated Social Graph Generator (S3G2) [Pham et al. 2013] is a general framework which generates a directed labeled graph, where the nodes are objects with property values, and their structure is determined by the class a node belongs to. S3G2 does not aim at generating near real-world data, but at generating synthetic graphs with a correlated structure. It causes that the probability to choose a certain property value (from a pre-defined dictionary), or the probability to connect two nodes with an edge are influenced by existing data values. For example, it is possible to have a correlated degree distribution, from which the degree of each node is generated, correlated with the properties of node. Hence the generator can ensure that, e.g., people with many friends in a social network will typically post more pictures than people with few friends, i.e., the amount of friend nodes influences the amount of posted comment and picture nodes. The data generation process starts with generating a number of nodes with property values generated according to specified property value correlations and then adding respective edges according to specified correlation dimensions. It has multiple phases, each focusing on one correlation dimension. Each pass along one correlation dimension is a Map phase in which data is generated, followed by a Reduce phase that sorts the data along the correlation dimension in the next pass. A heuristic observation that "the probability that two nodes are connected is typically skewed with respect to some similarity between the nodes" enables to focus only on sliding window of most probable candidates. The core idea of the framework is demonstrated using an example of a social network (consisting of persons and social activities). The dictionaries for property values are inspired by DBpedia and provided with 20 property value correlations. The edges are generated according to 3 correlation dimensions.

Cloning of Social Networks. Paper [Ali et al. 2014] introduces a synthetic network generator designed for cloning social network statistics of an existing dataset. The network starts with a small number of nodes, and new nodes are added until the network reaches the required number. It has two basic parameters: homophily and link density. A high *homophily* value indicates that links are more likely to be formed between nodes with the same label; these labels can be viewed as being equivalent to community membership.

Attribute Synthetic Generator (ASG) is a network generator for reproducing the node feature distribution of standard networks and rewiring the network to preferentially connect nodes that exhibit a high feature similarity. The network is initialized with a group of three nodes, and new nodes and links are added to the network based on link density, homophily, and feature similarity. As new nodes are created, their labels are assigned based on the prior label distribution. After the network has reached the same number of nodes as the original social media dataset, each node initially receives a random attribute assignment. Then a stochastic optimization process is used to move the initial assignments closer to the target distribution extracted from social media dataset using the Particle Swarm Optimization algorithm. The tuned attributes are then used to add additional links to the network based on the feature similarity parameter – a source node is selected randomly and connected to the most similar node. Multi-Link Generator (MLG) further uses link co-occurrence statistics from the original dataset to create a multiplex network. MLG uses the same network growth process as ASG. Based on the link density parameter, either a new node is generated with a label based on the label distribution of the target dataset or a new link is created between two existing nodes.

LDBC SNB. The Social Network Benchmark (SNB) [Erling et al. 2015] provided by LDBC consists of three distinct benchmarks on a common dataset corresponding to three different workloads. SNB models a social network akin to Facebook. The dataset consists of persons and a friendship network that connects them; whereas the majority of the data is in the messages that these persons post in discussion trees on their forums. The three query workloads involve: (1) SNB-Interactive, i.e., complex read-only queries, that touch a significant amount of data, (2) SNB-BI which accesses a large percentage of all entities in the dataset and groups these in various dimensions, and (3) SNB-Algorithms, i.e., graph analysis algorithms, including PageRank, Community Detection, Clustering and Breadth First Search. The graph generator realizes power laws, uses skewed value distributions, and introduces plausible correlations between property values and graph structures. It is implemented on top of Hadoop to provide scalability.

Towards More Realistic Data. [Nettleton 2016] argued that the main body of existing work lies in topology generation which approximates the characteristics of a real social network (such as a small graph diameter, small average path length, skew degree distribution, and community structures), however, this is usually done without any data. Hence, they introduced a general stochastic modeling system which allows the users to populate a graph topology with data. The approach has three steps: (1) topology generation (using R-MAT) plus community identification using the Louvain method [Blondel et al. 2008] or usage of a real-world topology from SNAP[25], (2) data definition following distribution profiles, attribute value definitions, using a parameterizable set of data propagation rules and affinities, and (3) data population.

3.5. Graph Analytics

Graph analytics, especially in the context v Big Data, is a popular area of studying interesting structural specifics of graphs, usually various types of networks. Hence, the respective benchmarks and graph data generators developed for the purpose of testing graph analytics tools (such as, e.g., PowerGraph [Gonzalez et al. 2012] or Parallel Graph analytiX [Sevenich et al. 2016]) have to focus mainly on graphs with complex structures.

HPC Scalable Graph Analysis Benchmark. The HPC Scalable Graph Analysis Benchmark [HPC 2009; Bader and Madduri 2005] represents an application with multiple analysis techniques that access a single data structure representing a weighted, directed graph. The benchmark is composed of four separated operations (graph con-

[25] https://snap.stanford.edu/data/

struction, classification of large vertex sets, graph extraction with BFS, and graph analysis with betweenness centrality) on a graph that follows a power-law distribution. The graph generator constructs a list of edge tuples containing vertex identifiers (with the edge direction from the first one to the second one) and weights that represent data assigned to the edges of the multigraph in the form of positive integers with a uniform random distribution. The generator has the following parameters: number of vertices, number of edges, and maximum weight of an edge. The algorithm of the generator is based on R-MAT. Since the authors aim to avoid data locality, in the final step the vertex numbers are randomly permuted, and then edge tuples are randomly shuffled. A related project from the same authors developed for the 9th DIMACS Shortest Paths Challenge is GTgraph [Bader and Madduri 2006]. It involves three types of graphs: input graph instances used in the DARPA HPCS SSCA#2 graph theory benchmark (version 1.0), Erdös-Rényi random graphs, and small-world graphs based on the R-MAT model.

Graphalytics. Graphalytics[26] is an industrial-grade benchmark for graph analysis platforms [Iosup et al. 2016]. It involves 6 real-world datasets and 2 synthetic datasets generated so that they cover two commonly used types of graphs: social network graphs generated using LDBC SNB graph generator (see Section 3.4) and power-law graphs generated by Graph500 (see Section 3.1). The benchmark workload consists of 6 deterministic algorithms: breadth-first search, PageRank, weakly connected components, community detection using label propagation, local clustering coefficient, and single-source shortest paths. The benchmark uses various metrics to measure the performance and throughput for systems under test such as *upload time* that measures the required time to preprocess and convert the graph into a suitable format for a graph processing system, or *Makespan* that measures the total execution time of a benchmarking workload algorithm. The benchmark also describes experiments for measuring the scalability of the systems under test. To facilitate the end user job of running the designed experimental workload, the benchmark provides a performance evaluation framework, Granula[27].

3.6. Others

In the last subsection we focus on use cases where the amount of graph data generators is very small. Still we believe that these areas are important with regard to the research activity and, hence, we can assume that more generators of testing data will appear soon.

3.6.1. Community Detection. Community detection is one of the many graph analytics algorithms typically used on domains such as social networks or bioinformatics. Communities are groups of vertices that are highly connected among them, while being scarcely connected to the rest of the graph. Such communities emerge from the fact that real graphs are not random, but follow real-world dynamics that make similar entities to have a larger probability to be connected. As a consequence, detected communities are used to reveal vertices with similar characteristics, for instance to discover functionally equivalent proteins in protein-protein interaction networks, or persons with similar interests in social networks. Such applications have made community detection a hot topic during the last fifteen years with tens of developed algorithms and detection strategies [Zhao ; Kim and Lee 2015]. In order to compare the quality of the different proposed techniques, one needs graphs with *reference communities*, that is, communities known beforehand. Since it is very difficult to have large

[26]http://graphalytics.ewi.tudelft.nl/
[27]https://github.com/atlarge-research/granula

real graphs with reference communities (mainly because these would require a manual labeling), graphs for benchmarking community detection algorithms are typically generated synthetically.

The first attempts to compare community detection algorithms using synthetic graphs proposed the use of random graphs composed by several Erdös-Rényi subgraphs, connected more internally than externally [Danon et al. 2005]. Each of these subgraphs has the same size and the same internal/external density of edges. However, such graphs miss the realism observed in real graphs, where communities are of different sizes and densities. Thus, Lancichinetti, Fortunato and Radicchi (hence LFR) [Lancichinetti et al. 2008] propose a class of benchmark graphs for community detection where communities are of different sizes and densities. The generator generates communities of different sizes following a power-law distribution whose parameters can be configured. The degree of the nodes is also sampled from a power-law distribution. Additionally, the generator introduces the concept of the "mixing factor", which specifies the fraction of its links connecting nodes belonging to different communities. Such parameter allows the degree of modularity of the generated graph to be tuned, thus testing the robustness of the algorithms under different conditions. The generation process starts with an empty graph and incrementally fills in the adjacency matrix by obeying the described constraints. Lancichinetti, Fortunato and Radicchi [Lancichinetti and Fortunato 2009] extended LFR to support the notion of directed graphs and overlapping communities. Overlapping communities extend the notion of communities by allowing the sharing of vertices, thus a vertex can belong to more than one community. This extended generator allows controlling the same parameters of the previous version, but also the amount of overlap of the generated communities.

Besides synthetic graph generators, Yang and Leskovec [Yang and Leskovec 2015] proposed the use of real-world graphs with explicit group annotations (e.g., forums in a social network, categories of products, etc.) to infer what they call "meta-communities", and use them to evaluate overlapping community detection algorithms. However, a recent study from Hric, Darst and Fortunato [Hric et al. 2014] reveal a loose correspondence between communities (the authors refer to them as structural communities) and meta-communities. This result reveals that algorithms working for structural communities do not work well for finding meta-communities and vice versa, suggesting significantly different underlying characteristics between the two types of communities, which are yet to be identified.

In this regard and to the best of our knowledge, there does not exist a set of generators to generate graphs with meta-communities for community detection algorithm benchmarking. The closest one is the LDBC-SNB data generator [Erling et al. 2015] which has been provided by the generation of groups of users in the social network. Even though the generation process does not specifically enforce the generation of groups (meta-communities) for benchmarking community detection algorithms, the study from Prat-Peŕez and Dominguez-Sal reveals that these groups are more similar to the real meta-communities than those structural communities generated by the LFR benchmark [Prat-Pérez and Dominguez-Sal 2014].

The differences observed between structural and meta-communities reveal the need of more accurate community definitions tight more specifically to the domain or use case. Current community detection algorithms and graph generators for community detection are stuck to the traditional (and vague) definition of community, assuming that there exists a single algorithm that would fit all the use cases. Thus, future work requires the study of domain-specific community characteristics that can be used to generate graphs with a community structure that accurately resembles that of specific use cases, and thus revealing which are the best algorithms for each particular scenario.

3.6.2. Graph Data Streaming. One way for dealing with big graphs is to process them using the *streaming* mode where the data stream could consist of the edges of the graph. In this mode, the graph processing algorithms can process the input stream in the order it arrives while using only a limited amount of memory [McGregor 2014]. The streaming mode has mainly attracted the attention of the RDF and Semantic Web community. Thus, Phuoc et al. [Le-Phuoc et al. 2012] presented an evaluation framework for linked stream data processing engines. The framework uses a data generator for the Stream Social network data Generator (S2Gen) that simulates what users continuously generate on their social network activities (e.g., posts) in addition to the user metadata such as users' profile information, social network relationships, posts, photos and GPS information. The data generator of this framework provides the users the flexibility to control the characteristics of the generated data by tanning a range of parameters including the period in which the social activities are generated (generating period), the maximum number of posts/comments/photos for each user per week and the correlation probabilities between the different objects (e.g., users) in the social network.

Tommasini et al. [Tommasini et al. 2017] introduced another framework for benchmarking RDF Stream Processing systems, RSPLab. The Streamer component of this framework is designed to publish RDF streams from the various existing RDF benchmarks (e.g., BSBM, LUBM) (see Section 3.2). In particular, the Streamer component uses TripleWave[28], an open-source framework for publishing and sharing RDF streams on the Web [Mauri et al. 2016]. TripleWave acts as a mean for plugging-in diverse Web data sources and for consuming streams in both push and pull mode.

4. GRAPH DATASETS

Besides graph data generators, many graph algorithms and systems are benchmarked using real-world graph data sets, including new graph data generation techniques. In this section, we review the most widely used graph datasets in the literature and their purpose.

Large Scale Graph Analytics. Large scale graph analytics systems are usually benchmarked using a combination of real-world and synthetically generated graphs. Because of the rapid evolution of such systems and their increasing capability to process larger graphs, the datasets typically used in the literature quickly change overtime. The most commonly used datasets (which can be downloaded from several graph dataset repositories [Leskovec and Krevl 2014; degli studio di Milano 2018]) are:

— **DBLP** [Yang and Leskovec 2015]: Represents a co-authorship network where researchers from DBLP are represented as vertices and there exists an edge between them if they have published a paper together.
— **Amazon** [Yang and Leskovec 2015]: Represents a product co-purchasing network, such that each vertex is a product and an edge between two products exists if a person has bought the two products.
— **Road networks** [Leskovec et al. 2009]: Consist of a set of road networks from the US.
— **LiveJournal** [Yang and Leskovec 2015]: A social network where vertices represent the users and the edges represent the acquitances between them.
— **Orkut** [Yang and Leskovec 2015]: Like LiveJournal, this is a social network where vertices represent persons and the edges represent their friendships.
— **Twitter** [Kwak et al. 2010]: This is a directed network representing the follower-followee interaction of Twitter.

[28]http://streamreasoning.github.io/TripleWave/

— **Friendster** [Yang and Leskovec 2015]: A social network like LiveJournal and Orkut, where nodes represent persons and edges their relationships.
— **WebUK** [Boldi et al. 2008]: A web graph of the UK subdomain, where nodes represent websites and edges represent the hyperlinks between them.
— **ClueWeb2012** [Project 2012]. This is a crawl of the web from 2012, where vertices represent websites and the edges represent the hyperlinks connecting them.

Recommender Systems. Many recommender systems are based on graphs, either bipartite graphs or many-to-many graphs. Such systems are usually tested on several real-world datasets, many of them containing information about the rating of products or other items such as movies, books or songs. The following are the most widely used datasets for testing recommender systems:

— **MovieLens** [GroupLens 2018]: Consists of a bipartite graph between users and movies, where edges represent the ratings the different users have made to the movies they watched.
— **Book-Crossings** [Ziegler et al. 2005]: Is a dataset with book ratings from users.
— **Jester** [Goldberg et al. 2001]: This dataset contains jokes (with their text) and ratings from users.
— **Last.fm** [GroupLens 2011]: This dataset contains the list of the top most listened artists per user, including the number of times the songs from those artists were played. Additionally, it contains connections between users (the social network they form), and a set of tags attached to artists that can be used to create content vectors.
— **Netflix** [Zhou et al. 2008]: This dataset was used during the famous Netflix prize. It consists of a bipartite graph with movie ratings from users.

Reputation Algorithms. Another application using social network data is that of assessing the degree of reputation of a user based on their past interactions and rankings [Kamvar et al. 2003; Katz and Golbeck 2006; Kumar et al. 2016]. Such algorithms are typically evaluated on networks with explicit user rankings or votings from other users, being the most widely used datasets the following examples:

— **Bitcoin** [Moore and Christin 2013]: This dataset contains information from several Bitcoin exchanges, where users are able to rate other users after their transactions. Here, the vertices represent the users and the edges, which are labelled, the ratings between them.
— **Wikipedia RFA** [West et al. 2014]: This is a network extracted from Wikipedia, where the vertices represent users and the edges, which are labelled, the votes emitted by administrators for the user to become an administrator. Each vote is accompained with a text explaining the vote's sign.
— **WikiSigned** [Maniu et al. 2011]: This is a network of Wikipedia editors where the vertices are the editors and the edges, which are labelled, represent the trust level between two editors.
— **Extended Epinions** [Massa and Avesani 2007]: This is an extended version of the Epinions network which also contains the levels of distrust between the users, expressed by means of edges between them.
— **Twitter Indian Elections** [Kagan et al. 2015]: This network represents a Twitter network where vertices represent users and there is an edge between two users if a user mentions another one in a tweet. The edges are labelled with the average sentiment of one user towards another.

Graph partitioning, Clustering and Community Detection. Many of the already discussed datasets are also typically used to test and compare graph partitioning, clustering and community detection algorithms. One can find a comprehensive list of datasets for such algorithms in [Bader et al. 2012; Yang and Leskovec 2015], some of which we have already discussed for other applications. Here, we summarize the most widely used in the literature:

— **DBLP**, **Amazon**, **LiveJournal**, **Orkut** and **Friendster** [Yang and Leskovec 2015] are widely used for evaluating community detection since they provide information about the meta-communities they contain. For instance, in Livejournal, users can join groups of users talking about given topics. Such groups are exported as meta-communities, and similar approaches are used for other graphs. Community detection algorithms are then evaluated by trying to infer such meta-communities without prior knowledge of the user to group assignment.
— **Zachary Karate Club** [Zachary 1977]: This graph consists of a small network of members of a karate club that was dismissed and split into two new karate clubs. Vertices represent persons and edges friendship relationships. Information about what people joined each of the two new karate clubs is provided.
— **PolBlogs** [Adamic and Glance 2005]: This is a network consisting of blogs talking about the US 2005 political elections, where a vertex represents a blog and the edges the hyperlinks between the blogs. Each blog has an associated label, whether it is left or right oriented.
— **PolBooks** [Bader et al. 2012]: Is a network of books of Amazon that talk about politics, and edges between two books exist if they are co-purchased together. The books have labels of whether they are left or right oriented.
— **Football** [Girvan and Newman 2002]: A network of American university football teams. Each node represents a football team and the edges represent the matches between them during the season. Each team is associated to a division.

Information Diffusion. Information networks are studied using graph algorithms in order to understand how information propagates. As such, there several datasets used to understand such networks.

— **Higgs-Twitter** [De Domenico et al. 2013]: This dataset contains the Twitter network before, during and after the discovery of the Higgs boson. Specifically, it contains the tweets, the mentions, retweets, followers/followees, etc.
— **Memetracker** [Leskovec et al. 2009]: Memetracker is a dataset that contains the quotes and phrases that appear more frequently on the entire news spectrum. It consists of the links of the news, the time and memes, and is used to understand how information spawns, evolves and dies.

Semantic Web and Knowledge bases. Semantic web and RDF engines have a well accepted set of real-world datasets, used to test reasoning engines as well as to use as baselines for the creation of new synthetic generators for the semantic web.

— **DbPedia** [Bizer et al. 2009]: The DBPedia project aims at creating a structured version of Wikipedia. It allows users to semantically query the relationships between the different resources at Wikipedia.
— **Freebase** [Bollacker et al. 2008]: This is a collaborative knowledge base made of metadata that is mainly provided by community members. It contains structured data from multiple sources in an attempt to create a global resource of information accessible both from persons and machines. The project was discontinued in 2014 and replaced by Wikidata.
— **Yago** [Suchanek et al. 2007], **Yago2** [Hoffart et al. 2013] & **Yago3** [Mahdisoltani et al. 2013]: These are open source knowledge bases developed at the Max Planck Institute of Computer Science in Saarbrücken. They contain structured data that has been automatically extracted from Wikipedia and other sources.

— **Wikidata** [Vrandečić and Krötzsch 2014]: This is a knowledge base collaboratively edited by the community and hosted by the Wikimedia Foundation. It contains structured data from its sister projects Wikipedia, Wikivoyage, Wikisource and others.
— **Billion Triple Challenge** [Käfer and Harth 2014]: This is a semantic dataset crawled from different sources, including Freebase, DBPedia or the BBC. A detailed description of the crawling process can be found in [Käfer et al. 2012].

5. CHALLENGES AND OPEN PROBLEMS

To conclude the overview of the state-of-the-art of graph data generation, in this section we discuss several of the open challenges.

5.1. Simple Usage, Simple Parameters

The proposal of a data generator (not necessarily for graph data) has to face an important schism. On one hand, it must provide the user with as many parameters as possible in order to enable him/her to generate arbitrary data. This approach seems to be reasonable, but it entails a shortcoming due to the fact that ordinary users are unwilling to use complex benchmarking tools. This observation can be seen, for example, in the case of XML benchmarks – even though there exist robust and complex data generators (such as ToXGene [Barbosa et al. 2002], which supports the specification of structural aspects, value distributions, references etc.), the most popular benchmarking tool is XMark [Schmidt et al. 2002], which models a single use case and enables its users to specify just the size of the data. Hence, the other extreme is to provide a simple data generator which does not require any complex settings and thus guarantees a simple and fast benchmarking process.

Considering the complex structure of graph data and the variety of applications requiring highly specific types of graphs, the latter solution is difficult to implement. A reasonable compromise can be found in a data generator which is provided with sample graph data and is capable of automatic analysis of its structural and value features in order to learn the complex parameters.

5.2. Large Scale Graphs with Realistic Structure

Most of existing graph generators are focused on generating large graphs with realistic structural characteristics focus principally on reproducing the degree distribution and the clustering coefficient [Kolda et al. 2014; Edunov et al. 2016]. However, there are other structural characteristics that one might be interested in reproducing for a large graph, such as the diameter, the size of the largest connected component, or the hierarchical community structure. Graph practitioners are highly interested in knowing how other high-level structural characteristics affect the performance of graph queries and graph algorithms. Hence, a compelling open challenge consists of creating graph generators that allow one to reproduce diverse structural characteristics of the graphs along with large scale sizes.

5.3. Single- vs. Multi-domain

Most of existing graph generators also generate graphs that are either not labeled or are specific to a given domain (e.g. Social Networks). Graphs from different domains have different schemas, structural characteristics, property distributions, etc. which might have an impact on the performance of the application under test. Thus, graph processing engine developers are asking for generators or tools to flexibly and holistically generate multi-domain graphs in a flexible and holistic manner, allowing to configure aspects such as size, schemata, data distributions and other structural characteristics such as degree distributions, clustering coefficients, and so on.

5.4. Generating Noisy Graphs and Graphs with Anomalies

Injecting noise and/or anomalies and errors into graphs is crucial for testing both machine learning algorithms working on this complex data and data quality techniques aiming at detecting anomalies and repairing graph data.

Concerning the former, analyzing and labelling structural networks is deemed to be more difficult for graph datasets in the presence of noise. Since denoising graph data is difficult to achieve, several machine learning techniques have been adapted to work with noise (i.e. mislabeled samples) or outliers, such as imbalanced graph classification [Pan and Zhu 2013] and binary graph classification with positive and negative weights [Cheung et al. 2016]. Synthetic graph generators that take into account noisy and missing data have been studied in [Jr. and Getoor 2010], where graph identification is presented in order to model the inference of a cleaned output network from a noisy input graph. Concerning the latter, data quality techniques handling graph data are recently considering ad-hoc generation of graph data and graph quality rules in order to assess the effectiveness of error detection and data repairing algorithms [Fan et al. 2016; Arioua and Bonifati 2018]. The corresponding graph quality rules are typically handcrafted by domain experts, whereas an automatic generation of such rules along with the graph data generation in tandem would be an interesting future challenge for the community.

5.5. Streaming Graph Generators

Stream computing is a new paradigm that is necessitated by various modern data generation scenarios such as the ubiquity of mobile devices, location services, sensor pervasiveness and emerging IoT applications. These applications generate the data with high Velocity, one of the main 3V characteristics of big data applications [Sakr 2016]. In most of these high speed data generation scenarios, various objects are connected together with different relations and data exchanges in a graph-structured manner. The Semantic Web community has been considering the aspect of implementing streaming RDF generator and benchmarks, however, there is still a clear lack on considering this aspect in other important and timely domains such as IoT. In addition, graph streaming generators should consider some specific aspects for the stream processing domains such as the out-of-order handling (late arrivals) [Li et al. 2008] and the variety in the schemas and formats of the different data streaming sources. It is also recommended for the streaming graph generators to support the distributed environment as this is the most common scenario for such type of applications.

5.6. Evolving Graph Data

As user requirements as well as environments change, most of the existing applications naturally evolve over time, at least to some extent. This evolution usually influences the structure of the data and consequently all the related parts of the application (i.e., storage strategies, operations, indexes etc.). In the world of graph data such graphs that change with time are denoted as *evolving*, *temporal*, *dynamic*, or *time-varying*. They can be modelled as labeled graphs, where the labels capture some measure of time [Michail 2015].

The evolution of graphs can be considered from multiple perspectives. We can assume a static set of nodes and a varying set of edges. Or, there are applications where the graphs only “grow”, i.e., the set of nodes and/or edges is only extended with new items. In the most general case we can assume any changes in both set of nodes and set of edges. Anyway with the evolution aspect the complexity of classical graph problems increases significantly [Michail 2015; Wu et al. 2014a]. In some graph applications, such as, e.g., social networks, the evolution of the data is a significant aspect, especially in the activity graphs [Doreian and Stokman 1997; Kumar et al. 2006; Hellmann and Staudigl 2014; Wang et al. 2013; Kossinets and Watts 2006; Viswanath et al. 2009]. However, as shown in [Leskovec et al. 2005a; Leskovec et al. 2005c], evolving graphs have further specific features. For example, some graphs grow over time according to a *densification power law* which means that real graphs tend to sprout many more edges than nodes, and thus are densifying as they grow. Also the effective diameter of graphs tends to shrink or stabilize as the graph grows with time.

A related problem is *data versioning* and its respective ability to query across multiple versions of data or to carry out general analysis. This problem has been investigated for instance within the domain of Linked Open Data [Papakonstantinou et al. 2016; Meimaris and Papastefanatos 2016; Fernández et al. 2015; Fernández et al. 2015].

The respective data generator should hence be able to simulate a natural growth and/or changes in the structure of the graph with regards to the various features of distinct use cases. However, even though the area of dynamic graphs is intensively studied, surprisingly there seem to exist only very few proposals of a generator for dealing with this area. In [Goerke et al. 2012] the authors focus on *clustering dynamic graphs*, i.e. graphs where the clustering corresponds to the partition of nodes into natural groups based on the paradigm of intra-cluster density versus inter-cluster sparsity of edges. The generator generates a time series of random graphs $G_0, G_1, ..., G_n$, where G_t emerges from G_{t-1} via atomic updates, i.e., insertion or deletion of an edge or a vertex. The generator keeps track of a (dynamic) ground truth clustering. The probability of atomic events is chosen in a way that adheres to this clustering, without losing randomness.

Another recent proposal of a generator [Purohit et al. 2018] of temporal graphs results from an observation that small subgraph patterns in networks, called *network motifs* or *graphlets*, are crucial indicators of the structure and the evolution of the graphs [Paranjape et al. 2017]. For a given graph and a predefined ordered list of structural atomic motifs the generator first computes the distribution of the motifs in the graph. The distribution is then used to generate a synthetic graph with the same features.

5.7. Multi-model Data

With the dawn of Big Data and especially its variety aspect, new types of database management systems have emerged. One of the most interesting ones are the so-called *multi-model databases* that enable to store and thus query across structurally different data. There exist various types of multi-model systems combining distinct subsets of Big Data structures including graph data. For example, OrientDB[29] which was implemented on the basis of an object DBMS currently supports graph, document, key/value, and object models. Such type of DBMSs also needs a specific data generator that would enable to test new features and analyze efficiency of operations. However, since the multi-model systems are in the context of Big Data are rather new, there exist only a few benchmarks targeting multi-model DBMSs (such as Bigframe [Kunjir et al. 2014] or UniBench [Lu 2017]) with limited capabilities.

Another interesting approach to multi-model data is to adopt a unifying expressive graph data model, known as property graph data model [Bonifati et al. pear]. Such a model allows to specify multi-edges and list of properties for the nodes. Synthetic graph generators for property graphs and its companion standard graph query language [Angles 2018; Angles et al. 2018] are also needed in order to boost their availability and adoption for different communities.

[29] http://orientdb.com/orientdb/

5.8. Machine Learning Based Graph Generation

With the advent of neural networks and specially generative adversarial networks (GANs) [Goodfellow et al. 2014], several researchers have started to explore their application to generate graphs. This is the case of [Kipf and Welling 2016; Grover et al. 2018; Simonovsky and Komodakis 2018; Li et al. 2018; You et al. 2018], which present several generative models to generate realistic graphs. Such techniques still suffer from several problems. For instance, some of them are limited to learn from a single graph [Kipf and Welling 2016; Grover et al. 2018] or generate small graphs [Simonovsky and Komodakis 2018; Li et al. 2018; You et al. 2018]. The technique proposed in [You et al. 2018] is capable of generating graphs with complex edge dependencies (e.g. community structure) and is not restricted to graphs of a fixed size. However, there are still in general several open challenges, including the capability of learning from and generating large graphs comparable in size to those typically used for benchmarking, and robust generation techniques with structural guarantees (e.g. degree distribution, clustering coefficient, etc.).

5.9. Privacy-preserving Graph Generation

A lot of work has been conducted on techniques for publishing social network graphs with privacy guarantees [Wu et al. 2010]. However, the topic of generating social graphs with a realistic structure yet private has been barely explored.

Most of the existing work falls within the topic of graph generation with "differential privacy" [Dwork and Lei 2009] guarantees. More specifically, in [Wang and Wu 2013] the authors develop a differential privacy graph generation approach based on the dK-graph generation model [Mahadevan et al. 2006] that outperforms the Stochastic Kronecker Graph Model [Leskovec et al. 2005b] in terms of the produced structural properties, eventhough the results show that there is still room for improvement.

Following this line of research, recent work [Qin et al. 2017] extends the notion of differential privacy and propose an " edge local differential privacy" based graph generation method. The poposed method allows generating privacy preserving synthetic social graphs wihtout the need of a centralized data curator, while preserving structural properties more accurately than straw-hat methods such as Randomized Neighbor Lists (based on randomized response [Dwork et al. 2014]) and Degree-based Graph Generation (which perturbates the original graph degrees using the Laplace mechanism [Dwork and Lei 2009]). Again, eventhough the proposed technique outperforms the baselines, the results show that there is still room for improvement in terms of the structural properties of the generated graph.

6. CONCLUSION

Graph data occur in a vast amount of distinct applications, such as biology, chemistry, physics, computer science, or social sciences, to name just a few. Graphs form one of the most complex data structures requiring specific and usually sophisticated approaches for processing and analysis. The history of graph theory, that started from when these structures and their respective algorithms were studied can be traced back to the 18th century.

With the recent dawn of Big Data there have been more occurrences of large scale graphs where the efficiency of processing methods is critical. Approaches that work for smaller scale graphs often cannot be used, the data need to be processed in a distributed way and hence the efficiency is influenced by other aspects, such as limits of data transport. In addition, distribution of graphs, especially for highly connected cases, is a difficult task. Thus extensive testing of these methods for graphs of various sizes and structural complexity is extremely important.

The aim of this survey was to provide a thorough overview and comparison of graph data generators. We do not limit ourselves to a single application domain, but we cover the currently most popular areas of graph data processing. We believe that this wide scope provides a uniquely useful insight into state-of-the-art tools as well as open issues for both researchers and practitioners.

ACKNOWLEDGMENTS

The authors would like to thank Dr. Kamesh Madduri for consultations and suggestions on the covered areas.

REFERENCES

2004. *OWL Web Ontology Language Overview*. W3C. http://www.w3.org/TR/owl-features/

2008. *ISO/IEC 9075-1:2008 Information technology – Database languages – SQL – Part 1: Framework (SQL/Framework)*. ISO. http://www.iso.org/iso/catalogue_detail.htm?csnumber=45498

2009. *HPC Scalable Graph Analysis Benchmark*. GraphAnalysis.org. http://www.graphanalysis.org/benchmark/

2010. *Graph 500 Benchmark*. Cray User's Group (CUG). http://www.graph500.org/

2015. *GraphGen: Graph generator for Neo4j*. Graph Aware Ltd. http://graphgen.graphaware.com/

2015. *Semantic Publishing Benchmark v2.0*. LDBC. http://ldbcouncil.org/developer/spb

2016. *DBLP Computer Science Bibliography*. DBLP team. http://dblp.uni-trier.de/

2016. *The TPC Benchmark?H*. http://www.tpc.org/tpch/

Daniel J. Abadi, Adam Marcus, Samuel R. Madden, and Kate Hollenbach. 2007. *Using The Barton Libraries Dataset As An RDF Benchmark*. Technical Report MIT-CSAIL-TR-2007-036. MIT.

Lada A Adamic and Natalie Glance. 2005. The political blogosphere and the 2004 US election: divided they blog. In *Proceedings of the 3rd international workshop on Link discovery*. ACM, 36–43.

Charu C. Aggarwal and Karthik Subbian. 2014. Evolutionary Network Analysis: A Survey. *ACM Comput. Surv.* 47, 1 (2014), 10:1–10:36.

Leman Akoglu, Mary McGlohon, and Christos Faloutsos. 2008. RTM: Laws and a Recursive Generator for Weighted Time-Evolving Graphs. In *ICDM*. IEEE Computer Society, 701–706.

Awrad Mohammed Ali, Hamidreza Alvari, Alireza Hajibagheri, Kiran Lakkaraju, and Gita Sukthankar. 2014. Synthetic Generators for Cloning Social Network Data. In *Proceedings of the ASE International Conference on Social Informatics*. Cambridge, MA.

Güneş Aluç, Olaf Hartig, M. Tamer Özsu, and Khuzaima Daudjee. 2014. Diversified Stress Testing of RDF Data Management Systems. In *Proceedings of the 13th International Semantic Web Conference - Part I (ISWC '14)*. Springer-Verlag New York, Inc., New York, NY, USA, 197–212. DOI:http://dx.doi.org/10.1007/978-3-319-11964-9_13

Renzo Angles. 2018. The Property Graph Database Model. In *AMW (CEUR Workshop Proceedings)*, Vol. 2100. CEUR-WS.org.

Renzo Angles, Marcelo Arenas, Pablo Barceló, Peter A. Boncz, George H. L. Fletcher, Claudio Gutierrez, Tobias Lindaaker, Marcus Paradies, Stefan Plantikow, Juan F. Sequeda, Oskar van Rest, and Hannes Voigt. 2018. G-CORE: A Core for Future Graph Query Languages. In *SIGMOD Conference*. ACM, 1421–1432.

Renzo Angles, Peter Boncz, Josep Larriba-Pey, Irini Fundulaki, Thomas Neumann, Orri Erling, Peter Neubauer, Norbert Martinez-Bazan, Venelin Kotsev, and Ioan Toma. 2014. The Linked Data Benchmark Council: A Graph and RDF Industry Benchmarking Effort. *SIGMOD Rec.* 43, 1 (May 2014), 27–31. DOI:http://dx.doi.org/10.1145/2627692.2627697

Abdallah Arioua and Angela Bonifati. 2018. User-guided Repairing of Inconsistent Knowledge Bases. In *EDBT*. OpenProceedings.org, 133–144.

Timothy G. Armstrong, Vamsi Ponnekanti, Dhruba Borthakur, and Mark Callaghan. 2013. LinkBench: A Database Benchmark Based on the Facebook Social Graph. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data (SIGMOD '13)*. ACM, New York, NY, USA, 1185–1196. DOI:http://dx.doi.org/10.1145/2463676.2465296

David A. Bader and Kamesh Madduri. 2005. Design and Implementation of the HPCS Graph Analysis Benchmark on Symmetric Multiprocessors. In *Proceedings of the 12th International Conference on High Performance Computing (HiPC'05)*. Springer-Verlag, Berlin, Heidelberg, 465–476. DOI:http://dx.doi.org/10.1007/11602569_48

David A. Bader and Kamesh Madduri. 2006. *GTgraph – A Suite of Synthetic Random Graph Generators*. http://www.cse.psu.edu/~kxm85/software/GTgraph/index.html

David A. Bader, Henning Meyerhenke, Peter Sanders, and Dorothea Wagner. 2012. *10th DIMACS Implementation Challenge Workshop*. Georgia Institute of Technology, Atlanta, GA, USA. http://www.cc.gatech.edu/dimacs10/

Guillaume Bagan, Angela Bonifati, Radu Ciucanu, George Fletcher, Aurélien Lemay, and Nicky Advokaat. 2016. gMark: Schema-Driven Generation of Graphs and Queries. *IEEE Transactions on Knowledge and Data Engineering* (Nov. 2016). https://hal.inria.fr/hal-01402575

Albert-Laszlo Barabasi and Reka Albert. 1999. Emergence of Scaling in Random Networks. *Science* 286, 5439 (1999), 509–512. DOI:http://dx.doi.org/10.1126/science.286.5439.509

Albert-László Barabási and Réka Albert. 1999. Emergence of scaling in random networks. *science* 286, 5439 (1999), 509–512.

Denilson Barbosa, Alberto O. Mendelzon, John Keenleyside, and Kelly A. Lyons. 2002. ToXgene: An extensible template-based data generator for XML.. In *WebDB* (2003-03-06). 49–54. http://dblp.uni-trier.de/db/conf/webdb/webdb2002.html#BarbosaMKL02

Christopher L. Barrett, Richard J. Beckman, Maleq Khan, V. S. Anil Kumar, Madhav V. Marathe, Paula E. Stretz, Tridib Dutta, and Bryan Lewis. 2009. Generation and Analysis of Large Synthetic Social Contact Networks. In *Winter Simulation Conference (WSC '09)*. Winter Simulation Conference, 1003–1014. http://dl.acm.org/citation.cfm?id=1995456.1995598

Robert Battle and Dave Kolas. 2012. Enabling the geospatial semantic web with parliament and geosparql. *Semantic Web* 3, 4 (2012), 355–370.

Sotirios Beis, Symeon Papadopoulos, and Yiannis Kompatsiaris. 2015. *Benchmarking Graph Databases on the Problem of Community Detection*. Springer International Publishing, Cham, 3–14. DOI:http://dx.doi.org/10.1007/978-3-319-10518-5_1

Garrett Bernstein and Kyle O'Brien. 2013. Stochastic Agent-based Simulations of Social Networks. In *Proceedings of the 46th Annual Simulation Symposium (ANSS 13)*. Society for Computer Simulation International, San Diego, CA, USA, Article 5, 8 pages. http://dl.acm.org/citation.cfm?id=2499604.2499609

Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. 2009. DBpedia - A Crystallization Point for the Web of Data. *Web Semant.* 7, 3 (Sept. 2009), 154–165. DOI:http://dx.doi.org/10.1016/j.websem.2009.07.002

Christian Bizer and Andreas Schultz. 2009. The Berlin SPARQL Benchmark. *International Journal On Semantic Web and Information Systems* (2009).

Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment* 2008, 10 (2008), P10008. http://stacks.iop.org/1742-5468/2008/i=10/a=P10008

Paolo Boldi, Massimo Santini, and Sebastiano Vigna. 2008. A Large Time-Aware Graph. *SIGIR Forum* 42, 2 (2008), 33–38.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. AcM, 1247–1250.

Peter Boncz, Minh-Duc Pham, Orri Erling, Ivan Mikhailov, and Yrjana Rankka. 2013. *Social Network Intelligence BenchMark*. https://www.w3.org/wiki/Social_Network_Intelligence_BenchMark

Angela Bonifati, George Fletcher, Jan Hidders, and Alexandre Iosup. 2018. A Survey of Benchmarks for Graph-Processing Systems. In *Graph Data Management*. Springer.

Angela Bonifati, George Fletcher, Hannes Voigt, and Nikolay Yakovets. 2018 (to appear). *Querying Graphs*. Morgan & Claypool Publishers.

J. M. Carlson and John Doyle. 2000. Highly Optimized Tolerance: Robustness and Design in Complex Systems. *Phys. Rev. Lett.* 84 (Mar 2000), 2529–2532. Issue 11. DOI:http://dx.doi.org/10.1103/PhysRevLett.84.2529

Deepayan Chakrabarti and Christos Faloutsos. 2006. Graph Mining: Laws, Generators, and Algorithms. *ACM Comput. Surv.* 38, 1, Article 2 (June 2006). DOI:http://dx.doi.org/10.1145/1132952.1132954

Deepayan Chakrabarti, Yiping Zhan, and Christos Faloutsos. 2004. R-MAT: A Recursive Model for Graph Mining. In *Proceedings of the Fourth SIAM International Conference on Data Mining, Lake Buena Vista, Florida, USA, April 22-24, 2004*. 442–446. DOI:http://dx.doi.org/10.1137/1.9781611972740.43

James Cheng, Yiping Ke, Wilfred Ng, and An Lu. 2007. Fg-index: Towards Verification-free Query Processing on Graph Databases. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data (SIGMOD '07)*. ACM, New York, NY, USA, 857–872. DOI:http://dx.doi.org/10.1145/1247480.1247574

Gene Cheung, Weng-Tai Su, Yu Mao, and Chia-Wen Lin. 2016. Robust Semi-Supervised Graph Classifier Learning with Negative Edge Weights. *CoRR* abs/1611.04924 (2016).

Avery Ching, Sergey Edunov, Maja Kabiljo, Dionysios Logothetis, and Sambavi Muthukrishnan. 2015. One trillion edges: Graph processing at facebook-scale. *Proceedings of the VLDB Endowment* 8, 12 (2015), 1804–1815.

Marek Ciglan, Alex Averbuch, and Ladialav Hluchy. 2012. Benchmarking Traversal Operations over Graph Databases. In *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering Workshops (ICDEW '12)*. IEEE Computer Society, Washington, DC, USA, 186–189. DOI: http://dx.doi.org/10.1109/ICDEW.2012.47

Brian F. Cooper, Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, and Russell Sears. 2010. Benchmarking Cloud Serving Systems with YCSB. In *Proceedings of the 1st ACM Symposium on Cloud Computing (SoCC '10)*. ACM, New York, NY, USA, 143–154. DOI: http://dx.doi.org/10.1145/1807128.1807152

Leon Danon, Albert Diaz-Guilera, Jordi Duch, and Alex Arenas. 2005. Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment* 2005, 09 (2005), P09008.

Miyuru Dayarathna and Toyotaro Suzumura. 2014. Graph Database Benchmarking on Cloud Environments with XGDBench. *Automated Software Engg.* 21, 4 (Dec. 2014), 509–533. DOI: http://dx.doi.org/10.1007/s10515-013-0138-7

Manlio De Domenico, Antonio Lima, Paul Mougel, and Mirco Musolesi. 2013. The anatomy of a scientific rumor. *Scientific reports* 3 (2013), 2980.

Universita degli studio di Milano. 2018. Laboratory of Web Algorithmics. http://law.di.unimi.it/. (2018). [Online; accessed 03-July-2018].

David Dominguez-Sal, Norbert Martinez-Bazan, Victor Muntes-Mulero, Pere Baleta, and Josep Lluis Larriba-Pay. 2011. A Discussion on the Design of Graph Database Benchmarks. In *Proceedings of the Second TPC Technology Conference on Performance Evaluation, Measurement and Characterization of Complex Systems (TPCTC'10)*. Springer-Verlag, Berlin, Heidelberg, 25–40. http://dl.acm.org/citation.cfm?id=1946050.1946053

D. Dominguez-Sal, P. Urbón-Bayes, A. Giménez-Vañó, S. Gómez-Villamor, N. Martínez-Bazán, and J. L. Larriba-Pey. 2010. Survey of Graph Database Performance on the HPC Scalable Graph Analysis Benchmark. In *Proceedings of the 2010 International Conference on Web-age Information Management (WAIM'10)*. Springer-Verlag, Berlin, Heidelberg, 37–48. http://dl.acm.org/citation.cfm?id=1927585.1927590

P. Doreian and F.N. Stokman. 1997. *Evolution of Social Networks*. Number sv. 1 in The journal of mathematical sociology. Gordon and Breach Publishers. https://books.google.com/books?id=ZL4zCCgfmOkC

Songyun Duan, Anastasios Kementsietsidis, Kavitha Srinivas, and Octavian Udrea. 2011. Apples and Oranges: A Comparison of RDF Benchmarks and Real RDF Datasets. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data (SIGMOD '11)*. ACM, New York, NY, USA, 145–156. DOI: http://dx.doi.org/10.1145/1989323.1989340

Cynthia Dwork and Jing Lei. 2009. Differential privacy and robust statistics. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*. ACM, 371–380.

Cynthia Dwork, Aaron Roth, and others. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science* 9, 3–4 (2014), 211–407.

Sergey Edunov, Dionysios Logothetis, Cheng Wang, Avery Ching, and Maja Kabiljo. 2016. Darwini: Generating realistic large-scale social graphs. *arXiv preprint arXiv:1610.00664* (2016).

Paul Erdos and Alfred Renyi. 1960. On the evolution of random graphs. *Publ. Math. Inst. Hungary. Acad. Sci.* 5 (1960), 17–61.

Orri Erling, Alex Averbuch, Josep Larriba-Pey, Hassan Chafi, Andrey Gubichev, Arnau Prat, Minh-Duc Pham, and Peter Boncz. 2015. The LDBC Social Network Benchmark: Interactive Workload. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data (SIGMOD '15)*. ACM, New York, NY, USA, 619–630. DOI: http://dx.doi.org/10.1145/2723372.2742786

Wenfei Fan, Yinghui Wu, and Jingbo Xu. 2016. Functional Dependencies for Graphs. In *SIGMOD Conference*. ACM, 1843–1857.

Javier D Fernández, Axel Polleres, and Jürgen Umbrich. 2015. Towards Efficient Archiving of Dynamic Linked Open Data. *Proc. of DIACHRON* (2015), 34–49.

Javier D. Fernández, Jürgen Umbrich, and Axel Polleres. 2015. BEAR: Benchmarking the Efficiency of RDF Archiving. (2015).

Alfio Ferrara, Davide Lorusso, Stefano Montanelli, and Gaia Varese. 2008. Towards a Benchmark for Instance Matching. In *Ontology Matching (OM 2008) (CEUR Workshop Proceedings)*, Pavel Shvaiko, J?r?me Euzenat, Fausto Giunchiglia, and Heiner Stuckenschmidt (Eds.), Vol. 431. CEUR-WS.org.

George Garbis, Kostis Kyzirakos, and Manolis Koubarakis. 2013. Geographica: A Benchmark for Geospatial RDF Stores (Long Version). In *The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part II*. 343–359. DOI:http://dx.doi.org/10.1007/978-3-642-41338-4_22

Maria Giatsoglou, Symeon Papadopoulos, and Athena Vakali. 2011. *Massive Graph Management for the Web and Web 2.0*. Springer Berlin Heidelberg, Berlin, Heidelberg, 19–58. DOI:http://dx.doi.org/10.1007/978-3-642-17551-0_2

M Girvan and MEJ Newman. 2002. Network of american football games between division ia colleges during regular season fall 2000. *Proc. Natl. Acad. Sci. USA* 99 (2002), 7821–7826.

Robert Goerke, Roland Kluge, Andrea Schumm, Christian Staudt, and Dorothea Wagner. 2012. *An Efficient Generator for Clustered Dynamic Random Networks*. Technical Report 17. Karlsruhe.

Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. 2001. Eigentaste: A constant time collaborative filtering algorithm. *information retrieval* 4, 2 (2001), 133–151.

Joseph E. Gonzalez, Yucheng Low, Haijie Gu, Danny Bickson, and Carlos Guestrin. 2012. PowerGraph: Distributed Graph-parallel Computation on Natural Graphs. In *Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation (OSDI'12)*. USENIX Association, Berkeley, CA, USA, 17–30. http://dl.acm.org/citation.cfm?id=2387880.2387883

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.

Michael Grossniklaus, Stefania Leone, and Tilmann Zaschke. 2013. *Towards a benchmark for graph data management and processing*. Technical Report.

GroupLens. 2011. HetRec. https://grouplens.org/datasets/hetrec-2011/. (2011). [Online; accessed 03-July-2018].

GroupLens. 2018. Movielens. https://grouplens.org/datasets/movielens/. (2018). [Online; accessed 03-July-2018].

Aditya Grover, Aaron Zweig, and Stefano Ermon. 2018. Graphite: Iterative generative modeling of graphs. *arXiv preprint arXiv:1803.10459* (2018).

Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin. 2005. LUBM: A benchmark for {OWL} knowledge base systems. *Web Semantics: Science, Services and Agents on the World Wide Web* 3, 2???3 (2005), 158 – 182. DOI:http://dx.doi.org/10.1016/j.websem.2005.06.005 Selcted Papers from the International Semantic Web Conference, 2004 ISWC, 20043rd. International Semantic Web Conference, 2004.

Tim Hellmann and Mathias Staudigl. 2014. Evolution of social networks. *European Journal of Operational Research* 234, 3 (2014), 583 – 596. DOI:http://dx.doi.org/10.1016/j.ejor.2013.08.022

Johannes Hoffart, Fabian M Suchanek, Klaus Berberich, and Gerhard Weikum. 2013. YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia. *Artificial Intelligence* 194 (2013), 28–61.

Darko Hric, Richard K Darst, and Santo Fortunato. 2014. Community detection in networks: Structural communities versus ground truth. *Physical Review E* 90, 6 (2014), 062805.

Alexandru Iosup, Tim Hegeman, Wing Lung Ngai, Stijn Heldens, Arnau Prat-Pérez, Thomas Manhardto, Hassan Chafio, Mihai Capotă, Narayanan Sundaram, Michael Anderson, Ilie Gabriel Tănase, Yinglong Xia, Lifeng Nai, and Peter Boncz. 2016. LDBC Graphalytics: A Benchmark for Large-scale Graph Analysis on Parallel and Distributed Platforms. *Proc. VLDB Endow.* 9, 13 (Sept. 2016), 1317–1328. DOI:http://dx.doi.org/10.14778/3007263.3007270

Amit Krishna Joshi, Pascal Hitzler, and Guozhu Dong. 2016. LinkGen: Multipurpose Linked Data Generator. In *The Semantic Web – ISWC 2016*, Paul Groth, Elena Simperl, Alasdair Gray, Marta Sabou, Markus Krötzsch, Freddy Lecue, Fabian Flöck, and Yolanda Gil (Eds.). Springer International Publishing, Cham, 113–121.

Galileo Mark S. Namata Jr. and Lise Getoor. 2010. Identifying graphs from noisy and incomplete data. *SIGKDD Explorations* 12, 1 (2010), 33–39.

Tobias Käfer and Andreas Harth. 2014. Billion Triples Challenge data set. Downloaded from http://km.aifb.kit.edu/projects/btc-2014/. (2014).

Tobias Käfer, Jürgen Umbrich, Aidan Hogan, and Axel Polleres. 2012. Towards a dynamic linked data observatory. *LDOW at WWW* (2012).

Vadim Kagan, Andrew Stevens, and VS Subrahmanian. 2015. Using twitter sentiment to forecast the 2013 pakistani election and the 2014 indian election. *IEEE Intelligent Systems* 1 (2015), 2–5.

Sepandar D Kamvar, Mario T Schlosser, and Hector Garcia-Molina. 2003. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the 12th international conference on World Wide Web*. ACM, 640–651.

Yarden Katz and Jennifer Golbeck. 2006. Social network-based trust in prioritized default logic. In *AAAI*, Vol. 6. 1345–1350.

Jungeun Kim and Jae-Gil Lee. 2015. Community Detection in Multi-Layer Graphs: A Survey. *SIGMOD Rec.* 44, 3 (Dec. 2015), 37–48. DOI:http://dx.doi.org/10.1145/2854006.2854013

Myunghwan Kim and Jure Leskovec. 2010. *Multiplicative Attribute Graph Model of Real-World Networks*. Springer Berlin Heidelberg, Berlin, Heidelberg, 62–73. DOI:http://dx.doi.org/10.1007/978-3-642-18009-5_7

Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308* (2016).

Donald Knuth. 2008. *The Stanford GraphBase*. http://www3.cs.stonybrook.edu/~algorith/implement/graphbase/implement.shtml

Tamara G Kolda, Ali Pinar, Todd Plantenga, and Comandur Seshadhri. 2014. A scalable generative graph model with community structure. *SIAM Journal on Scientific Computing* 36, 5 (2014), C424–C452.

Gueorgi Kossinets and Duncan J. Watts. 2006. Empirical Analysis of an Evolving Social Network. *Science* 311, 5757 (2006), 88–90. DOI:http://dx.doi.org/10.1126/science.1116869

Manolis Koubarakis and Kostis Kyzirakos. 2010. Modeling and querying metadata in the semantic sensor web: The model stRDF and the query language stSPARQL. In *Extended Semantic Web Conference*. Springer, 425–439.

Ravi Kumar, Jasmine Novak, and Andrew Tomkins. 2006. Structure and Evolution of Online Social Networks. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '06)*. ACM, New York, NY, USA, 611–617. DOI:http://dx.doi.org/10.1145/1150402.1150476

Srijan Kumar, Francesca Spezzano, VS Subrahmanian, and Christos Faloutsos. 2016. Edge Weight Prediction in Weighted Signed Networks.. In *ICDM*. 221–230.

Mayuresh Kunjir, Prajakta Kalmegh, and Shivnath Babu. 2014. Thoth: Towards Managing a Multi-System Cluster. *PVLDB* 7, 13 (2014), 1689–1692.

Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. 2010. What is Twitter, a social network or a news media?. In *Proceedings of the 19th international conference on World wide web*. AcM, 591–600.

Andrea Lancichinetti and Santo Fortunato. 2009. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Phys. Rev. E* 80, 1 (July 2009), 016118. DOI:http://dx.doi.org/10.1103/PhysRevE.80.016118

Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. 2008. Benchmark graphs for testing community detection algorithms. *Phys. Rev. E* 78, 4 (Oct. 2008), 046110. DOI:http://dx.doi.org/10.1103/PhysRevE.78.046110

Danh Le-Phuoc, Minh Dao-Tran, Minh-Duc Pham, Peter Boncz, Thomas Eiter, and Michael Fink. 2012. Linked stream data processing engines: Facts and figures. In *International Semantic Web Conference*. Springer, 300–312.

Jure Leskovec, Lars Backstrom, and Jon Kleinberg. 2009. Meme-tracking and the dynamics of the news cycle. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 497–506.

Jurij Leskovec, Deepayan Chakrabarti, Jon Kleinberg, and Christos Faloutsos. 2005a. Realistic, Mathematically Tractable Graph Generation and Evolution, Using Kronecker Multiplication. In *Proceedings of the 9th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'05)*. Springer-Verlag, Berlin, Heidelberg, 133–145. DOI:http://dx.doi.org/10.1007/11564126_17

Jure Leskovec, Deepayan Chakrabarti, Jon M. Kleinberg, and Christos Faloutsos. 2005b. Realistic, Mathematically Tractable Graph Generation and Evolution, Using Kronecker Multiplication. In *PKDD (Lecture Notes in Computer Science)*, Vol. 3721. Springer, 133–145.

Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2005c. Graphs over Time: Densification Laws, Shrinking Diameters and Possible Explanations. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining (KDD '05)*. ACM, New York, NY, USA, 177–187. DOI:http://dx.doi.org/10.1145/1081870.1081893

Jure Leskovec, Jon M. Kleinberg, and Christos Faloutsos. 2005d. Graphs over time: densification laws, shrinking diameters and possible explanations. In *KDD*. ACM, 177–187.

Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data. (June 2014).

Jure Leskovec, Kevin J. Lang, Anirban Dasgupta, and Michael W. Mahoney. 2008. Statistical Properties of Community Structure in Large Social and Information Networks. In *Proceedings of the*

17th International Conference on World Wide Web (WWW '08). ACM, New York, NY, USA, 695–704. DOI:http://dx.doi.org/10.1145/1367497.1367591

Jure Leskovec, Kevin J Lang, Anirban Dasgupta, and Michael W Mahoney. 2009. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics* 6, 1 (2009), 29–123.

Jin Li, Kristin Tufte, Vladislav Shkapenyuk, Vassilis Papadimos, Theodore Johnson, and David Maier. 2008. Out-of-order processing: a new architecture for high-performance stream systems. *Proceedings of the VLDB Endowment* 1, 1 (2008), 274–288.

Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. 2018. Learning deep generative models of graphs. *arXiv preprint arXiv:1803.03324* (2018).

Jiaheng Lu. 2017. Towards Benchmarking Multi-Model Databases http://udbms.cs.helsinki.fi/?projects/ubench. In *CIDR*.

Li Ma, Yang Yang, Zhaoming Qiu, Guotong Xie, Yue Pan, and Shengping Liu. 2006. Towards a Complete OWL Ontology Benchmark. In *Proceedings of the 3rd European Conference on The Semantic Web: Research and Applications (ESWC'06)*. Springer-Verlag, Berlin, Heidelberg, 125–139. DOI:http://dx.doi.org/10.1007/11762256_12

Priya Mahadevan, Dmitri Krioukov, Kevin Fall, and Amin Vahdat. 2006. Systematic topology analysis and generation using degree correlations. In *ACM SIGCOMM Computer Communication Review*, Vol. 36. ACM, 135–146.

Farzaneh Mahdisoltani, Joanna Biega, and Fabian M Suchanek. 2013. Yago3: A knowledge base from multilingual wikipedias. In *CIDR*.

Grzegorz Malewicz, Matthew H Austern, Aart JC Bik, James C Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski. 2010. Pregel: a system for large-scale graph processing. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*. ACM, 135–146.

Silviu Maniu, Bogdan Cautis, and Talel Abdessalem. 2011. Building a signed network from interactions in Wikipedia. In *Databases and Social Networks*. ACM, 19–24.

Paolo Massa and Paolo Avesani. 2007. Trust-aware recommender systems. In *Proceedings of the 2007 ACM conference on Recommender systems*. ACM, 17–24.

Andrea Mauri, Jean-Paul Calbimonte, Daniele DellAglio, Marco Balduini, Marco Brambilla, Emanuele Della Valle, and Karl Aberer. 2016. Triplewave: Spreading RDF streams on the web. In *International Semantic Web Conference*. Springer, 140–149.

Andrew McGregor. 2014. Graph stream algorithms: a survey. *ACM SIGMOD Record* 43, 1 (2014), 9–20.

Brendan McKay and Adolfo Piperno. 2014. *nauty and Traces*. http://pallini.di.uniroma1.it/

Marios Meimaris and George Papastefanatos. 2016. The EvoGen Benchmark Suite for Evolving RDF Data. In *Joint Proceedings of the 2nd Workshop on Managing the Evolution and Preservation of the Data Web (MEPDaW 2016) and the 3rd Workshop on Linked Data Quality (LDQ 2016) co-located with 13th European Semantic Web Conference (ESWC 2016), Heraklion, Crete, Greece, May 30th, 2016.* 20–35. http://ceur-ws.org/Vol-1585/mepdaw2016_paper_03.pdf

Othon Michail. 2015. *An Introduction to Temporal Graphs: An Algorithmic Perspective*. Springer International Publishing, Cham, 308–343. DOI:http://dx.doi.org/10.1007/978-3-319-24024-4_18

George A. Miller. 1995. WordNet: A Lexical Database for English. *Commun. ACM* 38, 11 (Nov. 1995), 39–41. DOI:http://dx.doi.org/10.1145/219717.219748

Tyler Moore and Nicolas Christin. 2013. Beware the middleman: Empirical analysis of Bitcoin-exchange risk. In *International Conference on Financial Cryptography and Data Security*. Springer, 25–33.

Mohamed Morsey, Jens Lehmann, Sören Auer, and Axel-Cyrille Ngonga Ngomo. 2012. Usage-centric Benchmarking of RDF Triple Stores. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI'12)*. AAAI Press, 2134–2140. http://dl.acm.org/citation.cfm?id=2900929.2901031

Mohamed Morsey, Jens Lehmann, Sören Auer, and Axel-Cyrille Ngonga Ngomo. 2011. *DBpedia SPARQL Benchmark – Performance Assessment with Real Queries on Real Data*. Springer Berlin Heidelberg, Berlin, Heidelberg, 454–469. DOI:http://dx.doi.org/10.1007/978-3-642-25073-6_29

David F. Nettleton. 2016. A synthetic data generator for online social network graphs. *Social Network Analysis and Mining* 6, 1 (2016), 44. DOI:http://dx.doi.org/10.1007/s13278-016-0352-y

Shirui Pan and Xingquan Zhu. 2013. Graph Classification with Imbalanced Class Distributions and Noise. In *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*. 1586–1592.

Vassilis Papakonstantinou, Giorgos Flouris, Irini Fundulaki, Kostas Stefanidis, and Giannis Roussakis. 2016. Versioning for Linked Data: Archiving Systems and Benchmarks. In *Proceedings of the Work-*

shop on Benchmarking Linked Data (BLINK 2016) co-located with the 15th International Semantic Web Conference (ISWC), Kobe, Japan, October 18, 2016. http://ceur-ws.org/Vol-1700/paper-05.pdf

Ashwin Paranjape, Austin R. Benson, and Jure Leskovec. 2017. Motifs in Temporal Networks. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining (WSDM '17)*. ACM, New York, NY, USA, 601–610. DOI:http://dx.doi.org/10.1145/3018661.3018731

Minh-Duc Pham, Peter Boncz, and Orri Erling. 2013. *S3G2: A Scalable Structure-Correlated Social Graph Generator*. Springer Berlin Heidelberg, Berlin, Heidelberg, 156–172. DOI:http://dx.doi.org/10.1007/978-3-642-36727-4_11

Nataliia Pobiedina, Stefan Rümmele, Sebastian Skritek, and Hannes Werthner. 2014. *Benchmarking Database Systems for Graph Pattern Matching*. Springer International Publishing, Cham, 226–241. DOI:http://dx.doi.org/10.1007/978-3-319-10073-9_18

Arnau Prat-Pérez and David Dominguez-Sal. 2014. How Community-like is the Structure of Synthetically Generated Graphs?. In *Proceedings of Workshop on GRAph Data Management Experiences and Systems (GRADES'14)*. ACM, New York, NY, USA, Article 7, 9 pages. DOI:http://dx.doi.org/10.1145/2621934.2621942

The Lemur Project. 2012. ClueWeb2012. http://law.di.unimi.it/webdata/clueweb12/. (2012). [Online; accessed 02-July-2018].

Eric Prud'hommeaux and Andy Seaborne. 2008. *SPARQL Query Language for RDF*. http://www.w3.org/TR/rdf-sparql-query/

Sumit Purohit, Lawrence B Holder, and George Chin. 2018. Temporal Graph Generation Based on a Distribution of Temporal Motifs. In *Proceedings of the 14th International Workshop on Mining and Learning with Graphs (MLG)*.

Shi Qiao and Z. Meral Özsoyoğlu. 2015. RBench: Application-Specific RDF Benchmarking. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data (SIGMOD '15)*. ACM, New York, NY, USA, 1825–1838. DOI:http://dx.doi.org/10.1145/2723372.2746479

Zhan Qin, Ting Yu, Yin Yang, Issa Khalil, Xiaokui Xiao, and Kui Ren. 2017. Generating synthetic decentralized social graphs with local differential privacy. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 425–438.

Carlos R. Rivero, Andreas Schultz, Christian Bizer, and David Ruiz. 2012. Benchmarking the Performance of Linked Data Translation Systems. In *WWW2012 Workshop on Linked Data on the Web, Lyon, France, 16 April, 2012*. http://ceur-ws.org/Vol-937/ldow2012-paper-09.pdf

Sherif Sakr. 2016. *Big data 2.0 processing systems: a survey*. Springer.

Sherif Sakr and Eric Pardede (Eds.). 2011. *Graph Data Management: Techniques and Applications*. IGI Global. DOI:http://dx.doi.org/10.4018/978-1-61350-053-8

Muhammad Saleem, Qaiser Mehmood, and Axel-Cyrille Ngonga Ngomo. 2015. *FEASIBLE: A Feature-Based SPARQL Benchmark Generation Framework*. Springer International Publishing, Cham, 52–69. DOI:http://dx.doi.org/10.1007/978-3-319-25007-6_4

Albrecht Schmidt, Florian Waas, Martin Kersten, Michael J. Carey, Ioana Manolescu, and Ralph Busse. 2002. XMark: A Benchmark for XML Data Management. In *Proceedings of the 28th International Conference on Very Large Data Bases (VLDB '02)*. VLDB Endowment, 974–985. http://dl.acm.org/citation.cfm?id=1287369.1287455

Michael Schmidt, Thomas Hornung, Michael Meier, Christoph Pinkel, and Georg Lausen. 2010. *SP2Bench: A SPARQL Performance Benchmark*. Springer Berlin Heidelberg, Berlin, Heidelberg, 371–393. DOI:http://dx.doi.org/10.1007/978-3-642-04329-1_16

Martin Sevenich, Sungpack Hong, Oskar van Rest, Zhe Wu, Jayanta Banerjee, and Hassan Chafi. 2016. Using Domain-specific Languages for Analytic Graph Databases. *Proc. VLDB Endow.* 9, 13 (Sept. 2016), 1257–1268. DOI:http://dx.doi.org/10.14778/3007263.3007265

Martin Simonovsky and Nikos Komodakis. 2018. GraphVAE: Towards Generation of Small Graphs Using Variational Autoencoders. *arXiv preprint arXiv:1802.03480* (2018).

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*. ACM, 697–706.

Riccardo Tommasini, Emanuele Della Valle, Andrea Mauri, and Marco Brambilla. 2017. RSPLab: RDF stream processing benchmarking made easy. In *International Semantic Web Conference*. Springer, 202–209.

Chad Vicknair, Michael Macias, Zhendong Zhao, Xiaofei Nan, Yixin Chen, and Dawn Wilkins. 2010. A Comparison of a Graph Database and a Relational Database: A Data Provenance Perspective. In *Proceedings of the 48th Annual Southeast Regional Conference (ACM SE '10)*. ACM, New York, NY, USA, Article 42, 6 pages. DOI:http://dx.doi.org/10.1145/1900008.1900067

Bimal Viswanath, Alan Mislove, Meeyoung Cha, and Krishna P. Gummadi. 2009. On the Evolution of User Interaction in Facebook. In *Proceedings of the 2Nd ACM Workshop on Online Social Networks (WOSN '09)*. ACM, New York, NY, USA, 37–42. DOI:http://dx.doi.org/10.1145/1592665.1592675

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM* 57, 10 (2014), 78–85.

Chong Jun Wang, Gang Wang, Yu Li Lei, and Shao Jie Qiao. 2013. Community Evolution in Dynamic Social Networks. In *Information Technology Applications in Industry, Computer Engineering and Materials Science (Advanced Materials Research)*, Vol. 756. Trans Tech Publications, 2634–2638. DOI:http://dx.doi.org/10.4028/www.scientific.net/AMR.756-759.2634

Sui-Yu Wang, Yuanbo Guo, Abir Qasem, and Jeff Heflin. 2005. *Rapid Benchmarking for Semantic Web Knowledge Base Systems*. Springer Berlin Heidelberg, Berlin, Heidelberg, 758–772. DOI:http://dx.doi.org/10.1007/11574620_54

Yue Wang and Xintao Wu. 2013. Preserving differential privacy in degree-correlation based graph generation. *Transactions on data privacy* 6, 2 (2013), 127.

Robert West, Hristo S Paskov, Jure Leskovec, and Christopher Potts. 2014. Exploiting social network structure for person-to-person sentiment analysis. *arXiv preprint arXiv:1409.2450* (2014).

Huanhuan Wu, James Cheng, Silu Huang, Yiping Ke, Yi Lu, and Yanyan Xu. 2014a. Path Problems in Temporal Graphs. *Proc. VLDB Endow.* 7, 9 (May 2014), 721–732. DOI:http://dx.doi.org/10.14778/2732939.2732945

Hongyan Wu, Toyofumi Fujiwara, Yasunori Yamamoto, Jerven Bolleman, and Atsuko Yamaguchi. 2014b. BioBenchmark Toyama 2012: an evaluation of the performance of triple stores on biological data. *Journal of Biomedical Semantics* 5, 1 (2014), 32. DOI:http://dx.doi.org/10.1186/2041-1480-5-32

Xintao Wu, Xiaowei Ying, Kun Liu, and Lei Chen. 2010. A survey of privacy-preservation of graphs and social networks. In *Managing and mining graph data*. Springer, 421–453.

Jaewon Yang and Jure Leskovec. 2015. Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems* 42, 1 (2015), 181–213.

Yuan Yao, Jiufeng Zhou, Lixin Han, Feng Xu, and Jian Lü. 2011. *Comparing Linkage Graph and Activity Graph of Online Social Networks*. Springer Berlin Heidelberg, Berlin, Heidelberg, 84–97. DOI:http://dx.doi.org/10.1007/978-3-642-24704-0_14

Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. 2018. GraphRNN: Generating Realistic Graphs with Deep Auto-regressive Models. In *International Conference on Machine Learning*. 5694–5703.

Wayne W Zachary. 1977. An information flow model for conflict and fission in small groups. *Journal of anthropological research* 33, 4 (1977), 452–473.

Yunpeng Zhao. A survey on theoretical advances of community detection in networks. *Wiley Interdisciplinary Reviews: Computational Statistics* 9, 5 (????), e1403. DOI:http://dx.doi.org/10.1002/wics.1403

Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, and Rong Pan. 2008. Large-scale parallel collaborative filtering for the netflix prize. In *International Conference on Algorithmic Applications in Management*. Springer, 337–348.

Cai-Nicolas Ziegler, Sean M McNee, Joseph A Konstan, and Georg Lausen. 2005. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web*. ACM, 22–32.

Received August 2016; revised September 2016; accepted December 2016