# Graph Generators: State of the Art and Open Challenges

ANGELA BONIFATI, Lyon 1 University, France
IRENA HOLUBOVÁ, Charles University, Czech Republic
ARNAU PRAT-PÉREZ, Sparsity-Technologies, 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 Semantic Web, graph databases, social networks, and community detection, along with general graphs. 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 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:**
Angela Bonifati, Irena Holubová, Arnau Prat-Pérez, and Sherif Sakr. 2020. Graph Generators: State of the Art and Open Challenges. *ACM Comput. Surv.* 53, 2, Article 36 (April 2020), 30 pages.
https://doi.org/10.1145/3379445

## 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



The work of Sherif Sakr is funded by the European Regional Development Funds via the Mobilitas Plus programme (Grant No. MOBTT75). The work of Irena Holubová was partially funded by the GAČR Project No. 19-01641S.
Authors' addresses: A. Bonifati, Lyon 1 University, Liris CNRS Campus La Doua 23-25 Avenue Pierre de Coubertin, Villerubanne, 69622, France; email: angela.bonifati@univ-lyon1.fr; I. Holubová, Charles University, Faculty of Mathematics and Physics, Department of Software Engineering, Malostranské nám. 25, Prague, 118 00, Czech Republic; email: holubova@ksi.mff.cuni.cz; A. Prat-Pérez, Sparsity-Technologies, Compte Guell 13, Barcelona, Spain; email: aprat@ac.upc.edu; S. Sakr, University of Tartu, J Liivi 2, Tartu, 50409, Estonia; email: sherif.sakr@ut.ee.

0360-0300/2020/04-ART36 $15.00
https://doi.org/10.1145/3379445

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 Data graphs, and (2) a large set of small graphs (e.g., chemical compounds[1] or linguistics syntax trees[2]). Naturally, the algorithms used in these two classes differ a lot [115]. 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 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 orders 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 [112, 114].

In this survey, we provide a detailed overview of the state-of-the-art modern graph generators by focusing on those that are pertinent and suitable for data-intensive tasks and benchmarking context. 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 Semantic Web, graph databases, social networks, and community detection, along with general graphs. 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 main characteristics. The detailed review and analysis make this article useful for stimulating new graph data generation mechanisms as well as serving as a main 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.

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

*Differences with prior surveys.* To the best of our knowledge, this article 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 Semantic Web, graph databases, social networks, and community detection, along with general graphs. 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 [26], 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 Reference [26], 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 Reference [30], 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 generators that appeared in the last decade. We fill the gap of more recent social network generators in Section 3.4, as well as more recent representatives of the other categories.

Aggarwal and Subbian [2] 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 article. The data generation of evolutionary networks is based on the shrinking diameters and Densification Power Law (DPL), i.e., community-guided attachment properties [80]. Generation of graphs tackling Kronecker recursion with recursive tensor multiplication [5] is then considered in the above survey. We refer the readers to the aforementioned survey for evolutionary network generators and further discuss the open challenges of evolving graph data in Section 4.

*Outline.* The rest of the text is structured 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. In Section 4, we highlight some of the challenges and open problems of graph data synthesis before we conclude in Section 5.

## 2 CLASSIFICATION AND COMPARATIVE STUDY

To provide a general preview of the generators and 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 offer an overview of the approaches used in state-of-the-art work, after which we introduce our approach. As mentioned in the Introduction, since this survey is unique especially in terms of scope, our classification and comparative strategy differs as well.

The graph data generators can be classified using various distinct criteria. For example, Reference [31] introduces two categories—degree-based and procedural generators. In general, *degree-based generators* (e.g., Barabasi-Albert model [16]) are commonly attempting to find a graph that matches it, but without providing any information about the graph or attempting to match other graph features (e.g., eigenvalues, small diameter, etc). However, *procedural generators* (e.g., R-MAT [31]) are commonly targeting simple techniques to produce graphs that are matching the characteristics of the real-world graphs (such as, e.g., the power-law degree distribution).

Reference [30] introduces five categories of graph models that can be synthesized: (1) *random graph models* (e.g., Erdös-Rényi [46]) 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 [28]), resulting from the idea that power laws can result from tolerance to risks and resource optimizations; (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 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*, and so on.

### 2.1 Classification

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, and (5) community detection. 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 Table 1, 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* (column "Type"), i.e., 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 (column "Target/sample").
- Characteristics of *read*/*update* **operations** (columns "Read" and "Update"), 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* (column "Pro.") of the graph (Y/N feature),
  - supported types of *distributions* (column "Distributions") used for generating of the data,
  - *output format* (column "Output") of the produced graph, and
  - whether the generator is *distributed* (column "Dis.") and thus enables more efficient data generation (Y/N feature).

*Number of Generators, Size, and Output Format of the Data.* As we can see, the biggest set of available generators can be found in the Semantic Web application domain, probably due to its

[3] "GDBs" stands for graph databases, "SNs" stands for social networks, and "Co" stands for community detection. Value "–" means that the information is not available or relevant.

Table 1. Key Characteristics of the Generators

| | | Domain | | Operations | | Configuration | | | |
|---|---|---|---|---|---|---|---|---|---|
| | **Generator** | **Type** | **Target/sample** | **Read** | **Update** | **Pro.** | **Distributions** | **Output** | **Dis.** |
| **General** | Preferential attachment | – | – | N | N | Y | power-law | edge-list | N |
| | R-MAT | – | – | N | N | Y | power-law | edge-list | N |
| | HPC Scal. Graph Anal. | fixed | – | fixed | N | N | uniform | edge-list | N |
| | GraphGen | – | – | N | N | Y | user-defined | node/edge-list | N |
| | BTER | – | – | N | N | N | user-defined | edge-list | Y |
| | Darwini | – | – | N | N | N | user-defined | edge-list | Y |
| | RTG | – | – | N | N | N | power-law | edge-list | N |
| **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 |
| | [42] | extracted | – | N | N | Y | – | RDF | N |
| | DBPSB | extracted | DBpedia | templates | N | Y | random | RDF | N |
| | LODIB | fixed | e-commerce | N | N | Y | 44 types | 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 |
| | S2Gen | schema-driven | social network | Y | N | N | user-defined | RDF | N |
| | RSPLab | schema-driven | agnostic | Y | Y | N | user-defined | RDF | N |
| | LDBC SPB | fixed | media | mixes | N | Y | power-law, skewed values, value correlation | RDF | N |
| | LinkGen | schema-driven | user-defined | templates | N | N | Gaussian, Zipfian | RDF | N |
| **GDBs** | XGDBench | fixed | social network | generated | generated | Y | power-law | MAG | Y |
| | gMark | schema-driven | user-defined | generated | N | Y | uniform, normal, Zipfian | N-triples | N |
| | graphGen | pattern-driven | user-defined | – | – | Y | – | GraphJson, CypherQueries | N |
| **SNs** | [18] | fixed | social network | N | N | Y | simulation-driven | impl. NA | – |
| | [134] | fixed | social network | N | N | N | power-law | impl. NA | – |
| | LinkBench | fixed | social network | generated | generated | Y | Facebook | impl. NA | – |
| | S3G2 | fixed | social network | N | N | Y | Facebook | CSV, RDF | Y |
| | SIB | fixed | social network | mixes | mix | Y | from real-world data | RDF | N |
| | [6] | schema-driven | social-network | N | N | Y | power-law | CSV | N |
| | LDBC SNB | fixed | social network | generated | generated | Y | Facebook | CSV, RDF | Y |
| | [99] | schema-driven | social network | N | N | Y | power-law | impl. NA | – |

(Continued)

Table 1. Continued

| | | Domain | | Operations | | Configuration | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Generator | Type | Target/sample | Read | Update | Pro. | Distributions | Output | Dis. |
| Co | [37] | – | – | Y | N | N | uniform | edge-list | N |
| | LFR | – | – | Y | N | N | power-law | edge-list | N |
| | LFR-Overlapping | – | – | Y | N | N | power-law | edge-list | N |
| | Stochastic Block Models | – | – | Y | N | N | user-defined | edge-list | Y |

recent popularity and a number of research groups dealing with this topic. None of these generators is natively implemented in a distributed manner and thus primarily generating output at the Big Data scale. However, the *Linked Open Data* (LOD) is expected to be large in general; however, this is the case of the whole *LOD Cloud*,[4] but not necessarily of the particular data sets forming it.

The second large group of generators also corresponds to a popular application domain—social networks. In this case, the size of the output graph is important if we require realistic features of the result. However, surprisingly many of the proposals do not provide an implementation at all and amongst the others there is a high percentage of those that are not highly scalable.

In case of the other application-specific domains the amount of generators is relatively small. As we will show in Section 2.2, the situation is not so critical. Some of the application-specific generators can be re-used also in other application domains or the general graph generators can be used.

Considering the output format of the data, as expected, the generators produce data in a standard format (e.g., RDF) or in a reasonable graph-related form (e.g., edge list).

*Domain.* If we consider the features of the particular domain of the generators, then in most cases it is expectably *fixed*, i.e., it is pre-defined (describing, e.g., a social network) and cannot be modified. It is the simplest option, but it does mean that the generator is simple too—it can still focus on other complex aspects of the output. While the *schema-driven* approaches enable to influence the target domain using a user-supplied schema, there also exist approaches where the domain is *extracted* from sample data. The particular target (in case of fixed) or sample (in case of schema-driven or extracted) domains do not provide very rich areas. They either correspond to the respective application domains (like in the case of social networks), or they are based on well-known and commonly used data sets (such as, e.g., DBLP [121] or DBpedia [22]). Except for general graphs without any domain, in all other cases, we can find a flexible representative with schema-driven/pattern-driven domain or a domain extracted from sample data.

*Operations.* In the case of operations, most commonly the respective generators are accompanied with a set of fixed read operations or sequences of operations (query mixes) representing typical behavior of a user. In some cases this aspect is more flexible—either query templates are used or the operations are generated, e.g., to access most of the data in the generated graph. However, update operations are provided only in a small amount of cases. As we will discuss in Section 4.6, the more general problem of evolving graphs is still an open issue.

*Configuration.* A natural feature of the generators is to provide as realistic graphs as possible. Hence, most of them focus not only on the structure of the output graph but also the properties. For the purpose of generating graphs with near real-world characteristics various distributions

[4] https://lod-cloud.net/.

are used, such as power law, Zipfian, and so on. Especially interesting are distributions extracted from real-wold data (such as, e.g., Facebook, in the case of the social network domain).

### 2.2 Overlapping

As we have mentioned, the basic classification of the generators that we have used in this article is relatively vaguely based on the current application domains or research areas. In addition, some of the generators are either general, and thus can be used universally, or have features applicable in more than one domain/research area. So the classes we use can overlap, as depicted in Table 2.

For example, many general or domain-agnostic graph generators, such as the preferential attachment [16] or R-MAT, are typically used to test graph analytics frameworks when large real graphs are not available.

Similarly, some social network graph generators, such as 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, the 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 (e.g., Facebook uses MySQL in Reference [13]) to test a workload similar to Facebook and extend and complement it with more graph queries like those in LDBC SNB.

Generators for community detection aim, in general, at creating graphs with a more realistic structure (graphs with communities of nodes where the density of edges is larger internally than externally). Even though these generators are, in general, used to test community detection algorithms (they generate also the expected communities in the graph), some studies also use them for general graph purposes or to test graph analytics algorithms besides community detection.

## 3 GRAPH DATA GENERATORS

In this section, we discuss the various graph data generators based on the classification introduced before in more detail. For each category, we first describe the key features of each of the representative examples and summarize their strengths and weaknesses. The goal is to offer a detailed information about 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. In general, such generators focus on reproducing properties observed in real graphs regardless of their domain such as the degree distribution, the diameter, the presence of a large connected component, a community structure or a significantly large clustering coefficient.

*Preferential Attachment.* Barabasi and Albert [16] 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.* Recursive Matrix (R-MAT) is a procedural synthetic graph generator that is designed to generate power-law degree distributions [31]. The generator is recursive and employs a fairly small number of parameters. In principle, the strategy of this generator is to achieve simple means to produce graphs whose properties correspond to properties of the real-world graphs. In particular, the design goal of R-MAT is to produce graphs that mimic the degree distributions, imitate

Table 2. Overlapping of Classes of Generators

| | Generator | General | Semantic Web | Graph databases | Social networks | Community detection |
|---|---|---|---|---|---|---|
| **General** | Preferential attachment | **x** | | | | |
| | R-MAT | **x** | | | | |
| | HPC Scal. Graph Anal. | **x** | | | | |
| | GraphGen | **x** | | x | | |
| | BTER | **x** | | | | |
| | Darwini | **x** | | | | |
| | RTG | **x** | | | | x |
| **Semantic web** | LUBM | | **x** | | | |
| | LBBM | | **x** | | | |
| | UOBM | | **x** | | | |
| | IIMB | | **x** | | | |
| | BSBM | | **x** | | | |
| | $SP^2$Bench | | **x** | | | |
| | [42] | | **x** | | | |
| | DBPSB | | **x** | | | |
| | LODIB | | **x** | | | |
| | Geographica | | **x** | | | |
| | WatDiv | | **x** | | | |
| | RBench | | **x** | | | |
| | S2Gen | | **x** | | | |
| | RSPLab | | **x** | | | |
| | LDBC SPB | | **x** | | | |
| | LinkGen | | **x** | | | |
| **GDBs** | XGDBench | | | **x** | x | |
| | gMark | | x | **x** | x | |
| | graphGen | | | **x** | | |
| **SNs** | [18] | | | | **x** | |
| | [134] | | | | **x** | |
| | LinkBench | | | x | **x** | |
| | S3G2 | | x | x | **x** | |
| | SIB | | x | | **x** | |
| | [6] | | | | **x** | |
| | LDBC SNB | | x | x | **x** | |
| | [99] | | | | **x** | |
| **Co** | [37] | x | | | | **x** |
| | LFR | x | | | | **x** |
| | LFR-Overlapping | x | | | | **x** |
| | Stochastic Block Models | x | | | | **x** |

a community structure and have a small diameter. R-MAT can generate weighted, directed, and bipartite graphs.

*HPC Scalable Graph Analysis Benchmark.* The HPC Scalable Graph Analysis Benchmark [14, 57] consists of a weighted, directed graph that has a power-law distribution and four related analysis techniques (namely, graph construction, graph extraction with BFS, classification of large vertex sets, and graph analysis with betweenness centrality). The generator has the following parameters: the number of nodes, the number of edges, and maximum weight of an edge. It outputs a list of tuples containing identifiers of vertices of an edge (with the direction from the first one to the second one) and weights (positive integers with a uniform random distribution) assigned to the edges of the multigraph. The algorithm of the generator is based on R-MAT.

*GraphGen.* For the purpose of testing the scalability of an indexing technique called FG-index [32] on the size of the database of graphs, their average size and average density, the authors have also implemented a synthetic generator called GraphGen.[5] It relies on data generation code for associations and sequential patterns provided by IBM.[6] GraphGen yields a collection of undirected, labeled and connected graphs. It addresses 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 along with a list of edges.

*BTER.* Block Two-Level Erdös-Rény (BTER) [70] 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 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 report that BTER is able to generate graphs with billions of edges.

*Darwini.* Darwini [45] is an extension of BTER designed to run on Vertex Centric computing frameworks like Pregel [89] or Apache Giraph [34], 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 gathering the nodes of the graph into buckets based on the expected number of closed triangles that they need to close to attain the expected clustering coefficient. The latter 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 each 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.

*RTG.* The Random Typing Generator (RTG) [4] aims at generating realistic graphs. In particular, it outputs (un)weighted, (un)directed, as well as uni/bipartite graphs, whereas the realism of the output is ensured by 11 laws (e.g., densification power law, weight power law, small and shrinking diameter, community structure, etc.) known to be typically exhibited by real-world graphs. On input it requires 4 parameters ($k$, $q$, $W$, and $\beta$) that correspond to the core Miller's observation [94] that a random process (namely, having a keyboard with $k$ characters and a space, the process of

[5] https://www.cse.ust.hk/graphgen/.
[6] From 1996, no longer available at http://www.almaden.ibm.com/cs/projects/iis/hdb/Projects/data mining/mining.shtml.

random typing of $W$ words, where the probability of hitting a space is $q$ and the probability of hitting any other characters is $(1 - q)/k$) leads to Zipf-like power laws (of the resulting words) plus in addition (using imbalance factor $\beta$) ensure homophily and community structure. RTG is based on the idea creating an edge between pairs of consecutive words.

Reference [8] further extends this idea mainly in the direction of simplification of specifying of the parameters. Instead of Miller's parameters that are not much associated with graphs, the authors prove and exploit their relationship with size and density of the target graph.

*Strengths and Weaknesses of General Graph Generators.* In general, existing general graph generators produce graphs with the following properties: skewed degree distribution (e.g., power law), small diameter, a large largest connected component, large clustering coefficient, and some degree of community structure. Degree distribution can be typically configured, while other properties are just a result of the generation process and cannot be controlled by any means. This is not the case in the work of BTER and Darwini, which besides the degree distribution, they also allow tuning of the clustering coefficient. However, some use cases demand for more control on the characteristics of the generated graphs. This is the case, for example, of benchmarking, where the underlying graph structure has direct implications to the performance of the graph algorithms to run. For this reason, the design of graph generators with the capability of fine tuning the characteristics of the generated graphs still remains as an open challenge. The questions that remain are what characteristics to tune and which are the algorithms that depend on such characteristics.

### 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 synthetic data and data sets with real-world characteristics. The used data sets correspond to RDF representation of relational-like data [23, 60], social network-like data [118], or specific and significantly more complex data structures such as biological data [131]. 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)[7] considers the university domain. The ontology defines 43 classes and 32 properties [60]. In addition, 14 test queries are provided in the LUBM benchmark. In particular, the benchmark focuses on *extensional* queries, i.e., queries that target the particular data instances of the ontology, as an opposite to *intentional* queries, i.e., queries that target properties and classes of the ontology. The Univ-Bench Artificial (UBA) data generator features repeatable and random data generation (exploiting classical linear congruential generator, LCG, of numbers). In particular, the data that is produced by the generator are assigned zero-based indexes (i.e., *University0*, *University1*, etc.), thus they are reproducible at any time with the same indexes. The generator naturally allows to specify a seed for random number generation, along with the starting index and the desired number of universities.

An extension of LUBM, the Lehigh BibTeX Benchmark (LBBM) [127], enables generating synthetic data for different ontologies. The data generation process is managed through two main phases: (1) the property-discovery phase and (2) the data generation phase. LBBM provides a probabilistic model that can emulate the discovered properties of the data of a particular domain and generate synthetic data exhibiting similar properties. Synthetic data are generated using a Monte Carlo algorithm. The approach is demonstrated on the Lehigh University BibTeX ontology, which consists of 28 classes along with 80 properties. The LUBM benchmark includes 12 test queries that were designed for the benchmark data. Another extension of LUBM, the

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

University Ontology Benchmark (UOBM),[8] focuses on two aspects: (1) usage of all constructs of OWL Lite and OWL DL [125] and (2) lack of necessary links between the generated data that thus form isolated graphs [87]. 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.* Ferrara et al. [51] proposed the ISLab Instance Matching Benchmark (IIMB)[9] for the problem of instance matching. For any two objects $o_1$ and $o_2$ adhering to different ontologies or to the same ontology, instance matching is specified in the form of a function $Om(o_1, o_2) \rightarrow \{0, 1\}$, where $o_1$ and $o_2$ are linked to the same real-world object (in which case the function maps to 1) or $o_1$ and $o_2$ are representing different objects (in which case the function maps to 0). It targets the domain of movie data, which contains 15 named classes, along with 5 objects and 13 datatypes. The data are extracted from IMDb.[10] The data generator corresponds to a data modifier that 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, diverse aggregation criteria for properties, or missing values specification), and logical heterogeneity (such as, e.g., instantiation on disjoint classes or various subclasses of the same superclass).

*BSBM.* The Berlin SPARQL Benchmark (BSBM),[11] is centered around an e-commerce application domain with object types such as *Customer*, *Vendor*, *Product* and *Offer* in addition to the relationship among them [23]. The benchmark provides a workload that has 12 queries with 2 types of query workloads (i.e., 2 sequences of the 12 queries) emulating the navigation pattern and search of a consumer seeking a product. The data generator is capable of producing arbitrarily scalable datasets by controlling the number of products ($n$) as a scale factor. The scale factor also impacts other data characteristics, such as, e.g., the depth of type hierarchy of products, branching factor, the number of product features, and so on. BSBM can output two representations, i.e., an RDF representation along with a relational representation. Thus, BSBM also defines an SQL [65] representation of the queries. This allows comparison of SPARQL [107] results to be compared against the performance of traditional RDBMSs.

*SP²Bench.* The SP²Bench[12] is a language-specific benchmark [118] that is based on the DBLP dataset. The generated datasets follow the key characteristics of the original DBLP dataset. In particular, the data mimics the correlations between entities. All random functions of the generator use a fixed seed that ensures that the data generation process is deterministic. SP²Bench is accompanied by 12 queries covering the various types of operators such as RDF access paths in addition to typical RDF constructs.

*DBPSB.* DBpedia SPARQL Benchmark (DBPSB)[13] proposed at the University of Leipzig has been designed using workloads that have been generated by applications and humans [96, 97]. In addition, the authors argue that benchmarks like LUBM, BSBM, or SP²Bench resemble relational database benchmarks involving relational-like data, which is structured using a small amount of homogeneous classes, whereas, in reality, RDF datasets are tending to be more heterogeneous. For example, DBpedia 3.6 consists of 289,016 classes, whereas 275 of them are defined based on

[8] https://www.cs.ox.ac.uk/isg/tools/UOBMGenerator/.
[9] http://www.ics.forth.gr/isl/BenchmarksTutorial/.
[10] http://www.imdb.com/.
[11] http://wifo5-03.informatik.uni-mannheim.de/bizer/berlinsparqlbenchmark/.
[12] http://dbis.informatik.uni-freiburg.de/forschung/projekte/SP2B/.
[13] http://aksw.org/Projects/DBPSB.html.

the DBpedia ontology. In addition, in property values different data types as well as references to objects of the various types are used. Hence, they presented a universal SPARQL benchmark generation approach, which uses a flexible data production mechanism that mimics the input data source. This dataset generation process begins using an input dataset; then multiple datasets with different sizes are generated by duplicating all the RDF triples with changing their namespaces. For generating smaller datasets, an adequate selection of all triples is selected randomly or using a sampling mechanism over the various classes in the dataset. The methodology is applied on the DBpedia SPARQL endpoint and a set of 25 templates of SPARQL queries is derived to cover frequent SPARQL features.

*LODIB.* The Linked Open Data Integration Benchmark (LODIB)[14] has been designed with the aim of reflecting the real-world heterogeneities that exist on the Web of Data to enable testing of Linked Data translation systems [111]. 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.

*Geographica.* The Geographica benchmark[15] has been designed to target the area of geospatial data [53] and respective SPARQL extensions GeoSPARQL [19] and stSPARQL [72]. The benchmark involves a real-world workload that uses openly available datasets that cover various geometry elements (such as, e.g., lines, points, polygons, etc.) and a synthetic workload. In the former case there is (1) a micro benchmark that evaluates primitive spatial functions (involving 29 queries) and (2) a macro benchmark that tests the performance of RDF engines in various application scenarios such as map exploring and search (consisting of 11 queries). In the latter case of a synthetic workload the generator produces synthetic datasets of different sizes that corresponds to an ontology based on OpenStreetMap and instantiates query templates. The generated SPARQL query workload is corresponding to spatial joins and selection using two query templates.

*WatDiv.* The Waterloo SPARQL Diversity Test Suite (WatDiv)[16] has been designed at the University of Waterloo. It implements stress testing tools that focus on addressing the observation that the state-of-the-art SPARQL benchmarks do not fully cover the variety of queries and workloads [7]. The benchmark focuses on two types of query aspects—structural and data-driven – and performs a detailed analysis on existing SPARQL benchmarks (LUBM, BSBM, DBPSB, and SP$^2$Bench) using these two properties of queries. The structural features involve triple pattern 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 their diversity is insufficient for evaluation of the weaknesses/strengths of the distinct design alternatives implemented by the different RDF systems.

In particular, WatDiv, provides (1) a data generator that generates scalable datasets according to the WatDiv schema, (2) a query template generator that produces a set of query templates according to the WatDiv schema, and (3) a query generator that uses the generated templates and instantiates them with real RDF values from the dataset, and (4) a feature extractor that extracts the structural features of the generated data and workload.

[14] http://lodib.wbsg.de/.
[15] http://geographica.di.uoa.gr/.
[16] http://dsg.uwaterloo.ca/watdiv/.

*RBench*. RBench [109] is an application-specific benchmark that receives any RDF dataset as an input and produces a set of datasets, that have similar characteristics of the input dataset, using size scaling factor $s$ and (node) degree scaling factor $d$. These factors ensure that the original RDF graph $G$ and the synthetic graph $G'$ are similar and the average node degree and the number of edges of $G'$ are changed by $s$ and $d$, respectively. A query generation process has been implemented to produce five different types of queries (edge-based queries, node-based queries, path queries, star queries, subgraph queries) for any generated data. The benchmark project FEASIBLE [116] is also an application-specific benchmark; however, contrary to RBench, it is designed to produce benchmarks from the set of sample input queries of a user-defined size.

In practice, one way for handling big RDF 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 RDF processing algorithms can process the input stream in the order it arrives while using only a limited amount of memory [91]. The streaming mode has mainly attracted the attention of the RDF and Semantic Web community.

*S2Gen*. Phuoc et al. [78] presented an evaluation framework for linked stream data processing engines. The framework uses a dataset generated with the Stream Social network data Generator (S2Gen), which simulates streams of user interactions (or events) in social networks (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 stream by tuning a range of parameters, which includes the frequency at which interactions are generated, limits such as the maximum number of messages per user and week, and the correlation probabilities between the different objects (e.g., users) in the social network.

*RSPLab*. Tommasini et al. [122] 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). In particular, the Streamer component uses TripleWave,[17] an open-source framework that enables sharing RDF streams on the Web [90]. TripleWave acts as a means for plugging-in and combining streams from multiple Web data sources using either pull or push mode.

*LDBC*. The Linked Data Benchmark Council[18] (LDBC) [11] had the goal of developing open-source yet industrial-grade benchmarks for RDF and graph databases. In the Semantic Web domain, it released the Semantic Publishing Benchmark (SPB) [77] that has been inspired by the Media/Publishing industry (namely, BBC[19]). The application scenario of this benchmark simulates a media or a publishing organization that handles large amount of streaming content (e.g., news, articles). The data generator mimics three types of relations in the generated synthetic data: correlations of entities, data clustering, and random tagging of entities. Two workloads are provided: (1) basic, involving an interactive query-mix querying the relations between entities in reference data, and (2) advanced, focusing on interactive and analytical query-mixes. The LDBC has designed two other benchmarks: the Social Network Benchmark (SNB) [47] for the social network domain (see Section 3.4) and Graphalytics [64] for the analytics domain.

*LinkGen*. LinkGen is a synthetic linked data generator that has been designed to generate RDF datasets for a given vocabulary [66]. The generator is designed to receive a vocabulary as an input

[17] http://streamreasoning.github.io/TripleWave/.
[18] http://ldbcouncil.org/industry/organization/origins.
[19] http://www.bbc.com/.

and supports two statistical distributions for generating entities: Zipf's power-law distribution and Gaussian distribution. LinkGen can augment the generated data with inconsistent and noisy data such as updating a given datatype property with two conflicting values or adding triples with syntactic errors. The generator also provides a feature to inter-link the generated objects with real-world ones from user-provided real-world datasets. The datasets can be generated in any of of two modes: on-disk and streaming.

*Strengths and Weaknesses of Semantic Web Graph Generators.* Graphs are intuitive and standard representation for the RDF model that form the basis for the Semantic Web community, which has been very active on building several benchmarks, associated with graph generators that had various design principles.

A comparison of four RDF benchmarks (namely, TPC-H [120] data expressed in RDF, LUBM, BSBM, and SP$^2$Bench) and six real-wold data sets (such as, e.g., DBpedia, the Barton Libraries Dataset [1], or WordNet [95]) has been reported by Reference [42]. 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 target characteristics of each dataset. With respect to a type $T$ the degree of structuredness of a dataset $D$ is based on the regularity of instance data in $D$ in conforming to type $T$. The type system is extracted from the data set by finding the RDF triples that have property[20] and extract type $T$ from their object. Properties of $T$ are determined as the union of all properties of type $T$. 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 are belonging to the non-tested area of the spectrum. As a consequence, they introduce a new benchmark that receives as input any dataset associated with a required level of structuredness and size (smaller than the size of the original data), and exploits the input documents as a seed to produce a subset of the original data with the target structuredness and size. In addition, they show that structuredness and size mutually influence each other.

With the recent increasing momentum of streaming data, the Semantic Web community started to consider the issues and challenges of RDF streaming data. However, there are still a lot of open challenges that need to tackled in this direction such as covering different real-world application scenarios.

### 3.3 Graph Databases

Currently there exists a number of papers that compare the efficiency of graph databases with regard to distinct use cases, such as the community detection problem [20], social tagging systems [54], graph traversal [35], graph pattern matching [104], data provenance [123], or even several distinct use cases [58]. However, the number of graph data generators and benchmarks that have been designed specifically for graph data management systems (Graph DBMS) is relatively small. Either a general graph generator is used for benchmarking graph databases, such as, e.g., the HPC Scalable Graph Analysis Benchmark [40] or the graph DBMS benchmarking tools are designed while having in mind a more general scope. Hence, it is questionable whether a benchmark that is targeted specifically for graph databases is necessary. Reference [39] discussed this question and related topics. On the basis of a review of applications of graph databases (namely, social network analysis, genetic interactions and recommendation systems), the authors analyzed and discussed the features of the graphs for these types of applications and how such features can

[20] http://www.w3.org/1999/02/22-rdf-syntax-ns#type.

affect the benchmarking process, various 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 [38] is an extensible benchmarking platform for graph databases used in cloud-based systems. Its intent is to automate benchmarking of graph databases in the cloud by focusing on the domain social networking services. It extends the Yahoo! Cloud Multiplicative Attribute (MAG) Graph Serving Benchmark (YCSB) [36] and provides a set of standard workloads representing various performance issues. In particular, the workload of XGDBench involves basic operations such as read/insert/update/delete an attribute, loading of the list of neighbours, and BFS traversal. Using the generators, seven workloads are created, such as update heavy, read mostly, short-range scan, traverse heavy, and so on. The data model of XGDBench is a simplified version of the Multiplicative Attribute Graph (MAG) [68] model, a synthetic graph model that models the interactions between node attributes and graph structure. 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 required number of nodes and for each node the number of attributes, a threshold for random initialization of attributes, a threshold for edge affinity that determines the existence of an edge between two nodes, and an affinity matrix. Large graphs can be generated on multi-core systems by XGDBench multi-threaded version.

*gMark.* gMark [15] is a schema-driven and domain-agnostic generator of both graph instances and graph query workloads. It can generate instances under the form of N-triples and queries in various concrete query languages, including OpenCypher,[21] recursive SQL, SPARQL, and LogicQL. In gMark, it is possible to specify a *graph configuration* involving the admitted edge predicates and node labels occurring in the graph instance along with additional parameters such as degree distribution, occurrence constraints, and so on. The *Query workload configuration* describes parameters of the query workload to be generated, by including the number of queries, arity, shape, and selectivity of the queries. The problem of deciding whether there exists a graph that satisfies a defined graph specification $G$ is NP-complete. The same applies to the problem of deciding whether there exists a query workload compliant with a given query workload configuration $Q$. In view of this, gMark adopts a best effort approach in which the parameters specified in the configuration files are attained in a relaxed fashion to achieve linear running time whenever possible.

*GraphGen.* GraphAware GraphGen[22] is a graph generation engine based on Neo4j's[23] query language OpenCypher [84]. 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[24] 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 recent commits.

*Strengths and Weaknesses of Graph Database Generators.* The graph DBMS generators discussed in this section have in common the fact that they can generate semantically rich labeled graphs with properties (ranging from properties values in GraphGen to MAG structures in XGDBench). They are also capable of generating graph instances and query workloads in concrete syntaxes (among which OpenCypher in GraphGen and gMark) and one of them (XGDBench) can also handle update operations on both graph structure and content. However, more comprehensive

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

graph DBMS generators that also produce data manipulation operations (such as updates for graph databases) are urgently needed. Additionally, none of these generators is enabled to work on corresponding query languages for property graphs, such as the newly emerging standard GQL [29] and G-Core [10]. Hence, a full-fledged graph DBMS generator for property graphs and property graph query workloads [27] is still missing and there exists an interesting opportunity to build such a generator in the near future.

Another apparent inconvenience is represented by the fact that explicit correlations among graph elements cannot be encoded for instance in gMark or GraphGen, whereas they could be fruitful to reproduce the behavior of real-world graphs in which attribute values are correlated one with another. However, social network and Linked Data generators that support correlations (as highlighted in Sections 3.4 and 3.2) typically exhibit a fixed schema and are not not necessarily multi-domain as are some of the graph DBMS generators discussed in this section (namely, GraphGen and gMark).

### 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 graphs has to cope with two problems: (1) availability of the data and (2) privacy of the data. Hence, data generators that provide realistic synthetic social network graphs are in a great demand.

In general, any analysis of social networks identifies their various specific features [30]. For example, a social networks graph often has a high degree of transitivity of the graph (so-called *clustering coefficient*). Or, its diameter, i.e., the longest shortest path amongst some fraction (e.g., 90%) of all connected nodes, is usually low due to weak ties joining faraway cliques.

Another key 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 Reference [81]. Reference [105] analyzed the structure of communities (clustering coefficient, triangle participation ratio, diameter, bridges, conductance, and size) in both real-world graphs and outputs of existing graph generators such as LFR [76] and the LDBC SNB [47]. They found out that discovered communities in different graphs have common distributions and that communities of a single graph have different characteristics and are challenging to be represented using 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* [6, 13, 47, 76, 99, 103, 134] focused on reproducing aspects of the network. In *agent-based approaches* [18, 21] the networks are constructed by directly simulating the agents' social choices.

*Realistic Social Network.* Reference [18] focused on the construction of realistic social networks. For this purpose the authors combine both private and public data sets with large-scale agent-based techniques. The process works as follows: In the first step it generates synthetic data by combining public and commercial databases. In the second step, it determines a set of activity templates. A 24-hour activity sequence including geolocations is assigned to each synthetic individual. To demonstrate the approach, the authors create a synthetic US population consisting of people and households together with respective geolocations. For this purpose the authors combine simulation and data fusion techniques utilizing various real-world data sources such as U.S. Census data, responses to a time-use survey or an activity survey. The result is captured by a dynamic network of social contacts. Similar methods for agent-based strategies have been reported in Reference [21].

*Linkage vs. Activity Graphs.* Reference [134] distinguished between two types of social network graphs—the *linkage graph*, where nodes correspond to people and edges correspond to their friendships, and the *activity graph*, where nodes also represent people but edges correspond to their interactions. On the basis of the analysis of Flickr[25] social links and Epinions[26] network of user interactions, the authors discover that they both exhibit high clustering coefficient (community structure), power-law degree distribution and small diameter. Considering the dynamic properties they both have relatively stable clustering coefficient over time and follow the densification law. However, diameter shrinking is not observed in Epinions activity graph and there is a difference in degree correlation (i.e., frequency of mutual connections of similar nodes)—activity graphs have neutral, whereas linkage graphs have positive degree correlation. With regard to the findings, the proposed generator focuses on linkage graphs with positive degree correlation. For this purpose, it extends the forest fire spreading process algorithm [80] with link symmetry. It has two parameters: the *symmetry probability* $P_s$ and the *burning probability* $P_b$. $P_b$ ensures a forward burning process based on BFS in which fire burns strongly with $P_b$ approaching 1. $P_s$ ensures backward linking from old nodes to new nodes and "adds fuel to the fire as it brings more links."

*LinkBench.* The LinkBench benchmark [13] has been designed for the purpose of analysis of efficiency of a database storing Facebook's production data. The benchmark considers true Big Data and related problems with sharding, replication, and so on. 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 and nodes are generated concurrently during bulk loading. The space of node IDs is divided into chunks that enable parallel processing. 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) [103] is a general framework that produces a directed labeled graph whose vertices represent objects having property values. The respective classes determine the structure of the properties. S3G2 does not aim at generating near real-world data, but at generating synthetic graphs with a correlated structure. Hence, the existing data values influence the probability of choosing a particular property value from a predefined dictionary or connecting two nodes. For example, the degree distribution can be correlated with the properties of a node and thus, e.g., people who have many friend relationships typically post more comments and pictures. 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. Data is generated in a Map phase corresponding to a pass along one correlation dimension. Then the data are sorted along the correlation dimension in the following Reduce phase. 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 three correlation dimensions.

[25] https://www.flickr.com/.
[26] http://www.epinions.com/.

*SIB.* The developers of the Social Network Intelligence BenchMark (SIB)[27] based the design of their benchmark on the claim that the state-of-the-art benchmarks are limited in reflecting the characteristics of the real RDF databases and are mostly focusing on the relational style aspects. Hence, they proposed a benchmark for query evaluation using real graphs [25]. The proposed benchmark mimics using an RDF store for a social network. The distribution of the generated data for each type follows the distribution of the associated type inferred from real-world social networks. Additionally, association rules are exploited for representing the real-world data correlation in the generated synthetic data. The generated data is linked with the RDF datasets from DBpedia. The benchmark specification contains three query mixes—interactive, update, and analysis—expressed in SPARQL 1.1 Working Draft.

*Cloning of Social Networks.* Reference [6] introduces two synthetic generators to reproduce two characteristics typically observed in social networks: node features and multiple link types. Both generators extend the generator proposed by Reference [128], which 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 reflects that links have higher chances to be established among the nodes belonging to the same community, whereas the community membership is represented by the same labels.

The first proposed generator is Attribute Synthetic Generator (ASG), used 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. 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. The second proposed generator, so-called 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.* Despite having a common Facebook-like dataset, thanks to three distinct workloads the Social Network Benchmark (SNB) [47] provided by LDBC represents three distinct benchmarks. The network nodes correspond to people and the edges represent their friendship and messages they post in discussion trees on their forums. The three query workloads involve: (1) SNB-Interactive, i.e., complex read-only queries accessing a high portion of data, (2) SNB-BI, i.e., queries accessing a high percentage of entities and grouping them in various dimensions, and (3) SNB-Algorithms, involving graph analysis algorithms, such as community detection, PageRank, BFS, and clustering. The graph generator, called Datagen, is a fork of `S3G2` [103] and realizes power laws, uses skewed value distributions, and ensures reasonable correlations between graph structures and property values. Additionally, it extends `S3G2` with "spiky" patterns in the distribution of social network activity along the timeline, also provides the ability of generating update streams to the social network. Datagen is also based on Hadoop to provide scalability, but compared to `S3G2`, it contains numerous performance improvements and the ability to be

[27] https://www.w3.org/wiki/Social_Network_Intelligence_BenchMark.

deterministic regardless of the number of computer nodes used for the generation of the graphs and for a given set of configuration parameters.

*Towards More Realistic Data.* Reference [99] argued that the majority of existing works focuses on topology generation, which approximates the features of a real-world social network (e.g., community structures, skew degree distribution, a small average path length, or a small graph diameter); however, this is usually done without any data. Hence, they introduced a general stochastic modeling approach that enables the users to fill a graph topology with data. The approach has three steps: (1) topology generation (using R-MAT) plus community identification using the Louvain method [24] or usage of a real-world topology from SNAP,[28] (2) data definition that describes definitions of attribute values (distribution profiles) using a parameterizable set of affinities and data propagation rules, and (3) data population.

*Strengths and Weaknesses of Social Network Generators.* Compared to more general graph generators, social network generators focus mainly on reproducing intra- and inter-node feature correlations. Among existing generators, LDBC SNB and S3G2 look like the most advanced ones in terms of the complexity of the generated graph and the amount of features and correlations they can generate, while providing a large degree of scalability. Their generation process is based on input dictionaries and have configuration files that allow tweaking many parameters of the generated graphs, but their schema is mainly static and cannot be easily configured to meet the needs of other use cases besides the benchmarks they have been designed for. In this regard, the approaches like those proposed in References [99] and [6] offer a more flexible and understandable configuration process to tweak the types, values, and correlations between different features.

Regarding the correlation between the underlying graph structure and the node features, approaches such as LDBC SNB, S3G2, or Reference [99] take into account this aspect and the generated graphs have realistic structural properties while similar nodes have a larger probability of being connected. However, their approach seems to be more based on intuition and common sense than to be backed up by any study of how the relation between structure and attributes showcase in real social networks. In this regard, this remains as a clear open challenge for social network generators.

Finally, scalability is another aspect to be considered in social network graph generators. LDBC SNB and S3G2 are engineered with this in mind, thus they provide a way to scale to billions of nodes and edges. This is not the case for the other generators, which can make them impractical if our goal is to generate real sized social network graphs.

### 3.5 Testing Community Detection

Community detection is one of the many graph analytics algorithms typically used in domains such as social networks or bioinformatics. *Communities* are usually defined as sets of nodes that are highly mutually connected, while being scarcely connected to the other nodes of the graph. Such communities emerge from the fact that real-world 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-to-protein interaction networks, or persons with similar interests in social networks. Such applications have made community detection a hot topic during the last 15 years with tens of developed algorithms and detection strategies [67, 136]. For comparing 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

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

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

*Danon et al.* 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 [37]. 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-world graphs, where communities are of different sizes and densities, thus several proposals exist to overcome such an issue.

*LFR.* Lancichinetti, Fortunato and Radicchi (hence, LFR) [76] propose a class of benchmark graphs for community detection where communities are of diverse sizes and densities. The generated communities follow 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 consists of the percentage of edges in the graph connecting nodes that belong to distinct 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 is implemented as an optimization process starting with an empty graph and progressively filling it with nodes and edges guided by the specified constraints.

*LFR-Overlapping.* Lancichinetti, Fortunato, and Radicchi [75] 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 LFR, as well as the amount of overlap of the generated communities.

*Stochastic Block Models.* Another popular family of generators widely used in the community detection field are the stochastic block models [62]. In such models, the community structure of the graph is typically defined as an array of $n$ community or cluster sizes and a density square matrix of size $n \times n$ containing the density of intra-cluster edges (in the diagonal of the matrix) and the density of inter-cluster edges. Then, a stochastic procedure is run to sample graphs from such array and matrix, using the sizes to compute the possible edges and the densities as probabilities of such edges to exist. The popularity of these methods stem from its simplicity and scalability, which makes them suitable for generating large graphs fast and in distributed environments, provided that the density matrix is sparse (as it happens in most of real-world graphs). Moreover, given that the generation process of such models is mathematically tractable, they are typically used to analyze the limitations of algorithms for community detection such as those based on modularity optimization [52] or based on triads [106]. Extensions of such models exist, such as the Mixed Membership Stochastic Block Model [3], for overlapping communities.

*Strengths and Weaknesses of Community Detection Generators.* Besides synthetic graph generators, Yang and Leskovec [133] 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 [63] 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 are no available generators that can generate graphs with meta-communities for community detection algorithm benchmarking. The closest one is the LDBC SNB data generator, 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 [105] reveals that these groups are more similar to the real meta-communities than those structural communities generated by the LFR benchmark.

The differences observed between structural and meta-communities reveal the need of more accurate community definitions that are more tight and more specific to the domain or the 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.

## 4 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.

### 4.1 Simple Usage, Simple Parameters

The proposal of a data generator (not necessarily for graph data) has to face an important schism. It must provide the user with as many parameters as possible 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 [17], which supports the specification of structural aspects, value distributions, references, etc.), the most popular benchmarking tool is XMark [117], 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 that 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 to learn the complex parameters. We could see this type approach in some cases, such as Semantic Web generators LBBM or DBPSB.

### 4.2 Large-scale Graphs with Realistic Structure

Most of existing graph generators are focused on generating large graphs with realistic structural characteristics and focus principally on reproducing the degree distribution and the clustering coefficient [45, 70]. 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.

### 4.3 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, and so on, 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 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.

### 4.4 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 labeling structural networks is deemed to be more difficult for graph datasets in the presence of noise. Since de-noising graph data is difficult to achieve, various machine learning-based approaches have been adapted to work with noise (i.e., mislabeled samples) or outliers, such as imbalanced graph classification [100] and binary graph classification with positive and negative weights [33]. Synthetic graph generators that take into account noisy and missing data have been studied in Reference [98], where graph identification is presented 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 to evaluate the effectiveness of error detection and data repairing algorithms [12, 48]. 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.

### 4.5 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 [113]. 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) [82] 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 types of applications.

### 4.6 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 modeled as labeled graphs, where the labels capture some measure of time [93].

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 [93, 130]. In some graph applications, such as, e.g., social networks, the evolution of the data is a significant aspect, especially in the activity graphs [41, 61, 71, 73, 124, 126]. However, as shown in References [79, 80], evolving graphs have further specific features. For example, some graphs grow over time according to a *densification power law*, which means that in real graphs, edges tend to appear at a higher pace than vertices, meaning that these graphs densify as they grow. Also, the way the new edges are distributed has the effect of a shrinking diameter that ends up stabilizing 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 [49, 50, 92, 101].

The respective data generator should hence be able to simulate a natural growth and/or changes in the structure of the graph with regard 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 Reference [55], 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 concept of density of edges within and between the clusters. The generator generates a time series of random graphs $G_0, G_1, \ldots, G_n$, where $G_t$ emerges from $G_{t-1}$ via successive atomic updates like for instance, the insertion of a vertex or the removal of an edge. The generator dynamically monitors the ground truth clustering, and the probability of the updates is chosen in such a way that the ground truth is maintained while the randomness of the generated graph is kept.

Another recent proposal of a generator [108] 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 [102]. 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.

### 4.7 Multi-Model Data

With the dawn of Big Data and especially its Variety, another key 3V characteristic, new types of database management systems have emerged. One of the most interesting ones is the so-called *multi-model database* [86], which enable storing and thus querying across structurally different data, including unstructured, semi-structured, and structured. There exist various types of multi-model systems combining distinct subsets of Big Data structures including graph data. For example, OrientDB,[29] which has been mainly designed as an object DBMS, currently supports graph, document, key/value, and object models. Such types of DBMSs also need 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 rather new, there exist only a few benchmarks targeting multi-model DBMSs (such as Bigframe [74] or UniBench [85]) with limited capabilities.

Another interesting approach to multi-model data is to adopt a unifying expressive graph data model, so-called *property graph data model* [27]. Such a model allows to specify multi-edges and list of properties for the nodes. Synthetic graph generators for property graphs and its companion

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

standard graph query language [9, 10] are also needed to boost their availability and adoption for different communities.

### 4.8 Machine Learning-based Graph Generation

With the advent of neural networks and specially generative adversarial networks (GANs) [56], several researchers have started to explore their application to generate graphs. This is the case of References [59, 69, 83, 119, 135], 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 [59, 69] or generate small graphs [83, 119, 135]. The technique proposed in Reference [135] 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.).

### 4.9 Privacy-preserving Graph Generation

A lot of work has been conducted on techniques for publishing social network graphs with privacy guarantees [132]. 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" [43] guarantees. More specifically, in Reference [129] the authors develop a differential privacy graph generation approach based on the dK-graph generation model [88] that outperforms the Stochastic Kronecker Graph Model [79] in terms of the produced structural properties, even though the results show that there is still room for improvement.

Following this line of research, recent work [110] extends the notion of differential privacy and propose an "edge local differential privacy"-based graph generation method. The proposed method allows generating privacy preserving synthetic social graphs without 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 [44]) and Degree-based Graph Generation (which perturbates the original graph degrees using the Laplace mechanism [43]). Again, even though the proposed technique outperforms the baselines, the results show that there is still room for improving the structural properties of the generated graph.

## 5 CONCLUSION

Graph data occur in a vast amount of distinct applications, such as biology, chemistry, physics, computer science, or social sciences, to name 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, which 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 thank Dr. Kamesh Madduri for consultations and suggestions on the covered areas.

## REFERENCES

[1] 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.

[2] Charu C. Aggarwal and Karthik Subbian. 2014. Evolutionary network analysis: A survey. *ACM Comput. Surv.* 47, 1 (2014), 10:1–10:36.

[3] Edoardo M. Airoldi, David M. Blei, Stephen E. Fienberg, and Eric P. Xing. 2008. Mixed membership stochastic blockmodels. *J. Mach. Learn. Res.* 9(Sep. 2008), 1981–2014.

[4] Leman Akoglu and Christos Faloutsos. 2009. RTG: A recursive realistic graph generator using random typing. *Data Min. Knowl. Discov.* 19, 2 (2009), 194–209. DOI: https://doi.org/10.1007/s10618-009-0140-7

[5] Leman Akoglu, Mary McGlohon, and Christos Faloutsos. 2008. RTM: Laws and a recursive generator for weighted time-evolving graphs. In *Proceedings of the IEEE International Conference on Data Mining (ICDM'08)*. IEEE Computer Society, 701–706.

[6] 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*.

[7] 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 (ISWC'14)*. Springer-Verlag, New York, NY, 197–212. DOI: https://doi.org/10.1007/978-3-319-11964-9_13

[8] Khaled Ammar and M. Tamer Özsu. 2013. WGB: Toward a universal graph benchmark. In *Proceedings of the Workshop Series on Big Data Benchmarking (WBDB'13)*. 58–72. DOI: https://doi.org/10.1007/978-3-319-10596-3_6

[9] Renzo Angles. 2018. The property graph database model. In *Proceedings of the Alberto Mendelzon Central Europe Workshop* (CEUR'18), Vol. 2100. Retrieved from CEUR-WS.org.

[10] 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 *Proceedings of the ACM Special Interest Group on Management of Data (SIGMOD'18)*. ACM, 1421–1432.

[11] 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: https://doi.org/10.1145/2627692.2627697

[12] Abdallah Arioua and Angela Bonifati. 2018. User-guided repairing of inconsistent knowledge bases. In *Proceedings of the International Conference on Extending Database Technology (EDBT'18)*. OpenProceedings.org, 133–144.

[13] 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 ACM International Conference on Management of Data (SIGMOD'13)*. ACM, New York, NY, 1185–1196. DOI: https://doi.org/10.1145/2463676.2465296

[14] 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, 465–476. DOI: https://doi.org/10.1007/11602569_48

[15] Guillaume Bagan, Angela Bonifati, Radu Ciucanu, George Fletcher, Aurélien Lemay, and Nicky Advokaat. 2016. gMark: Schema-driven generation of graphs and queries. *IEEE Trans. Knowl. Data Eng.* (Nov. 2016). Retrieved from https://hal.inria.fr/hal-01402575.

[16] Albert-Laszlo Barabasi and Reka Albert. 1999. Emergence of scaling in random networks. *Science* 286, 5439 (1999), 509–512. Retrieved from arXiv:http://www.sciencemag.org/cgi/reprint/286/5439/509.pdf. DOI: https://doi.org/10.1126/science.286.5439.509

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

[18] 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 *Proceedings of the Winter Simulation Conference (WSC'09)*. 1003–1014. Retrieved from http://dl.acm.org/citation.cfm?id=1995456.1995598.

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

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

[21] 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. Retrieved from http://dl.acm.org/citation.cfm?id=2499604.2499609.

[22] 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: https://doi.org/10.1016/j.websem.2009.07.002

[23] Christian Bizer and Andreas Schultz. 2009. The Berlin SPARQL benchmark. *Int. J. Semant. Web Info. Syst.* 5, 2 (2009), 1–24.

[24] Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *J. Stat. Mech.: Theory Exper.* 2008, 10 (2008), P10008. Retrieved from http://stacks.iop.org/1742-5468/2008/i=10/a=P10008.

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

[26] Angela Bonifati, George Fletcher, Jan Hidders, and Alexandre Iosup. 2018. A survey of benchmarks for graph-processing systems. In *Graph Data Management*. Springer.

[27] Angela Bonifati, George Fletcher, Hannes Voigt, and Nikolay Yakovets. 2018. *Querying Graphs*. Morgan & Claypool Publishers.

[28] 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: https://doi.org/10.1103/PhysRevLett.84.2529

[29] Hassan Chafi, Jason Crawford, Alastair Green, and Keith Hare. 2018. Graph Query Language GQL. Retrieved from https://www.gqlstandards.org//.

[30] Deepayan Chakrabarti and Christos Faloutsos. 2006. Graph mining: Laws, generators, and algorithms. *ACM Comput. Surv.* 38, 1 (June 2006). DOI: https://doi.org/10.1145/1132952.1132954

[31] Deepayan Chakrabarti, Yiping Zhan, and Christos Faloutsos. 2004. R-MAT: A recursive model for graph mining. In *Proceedings of the 4th SIAM International Conference on Data Mining*. 442–446. DOI: https://doi.org/10.1137/1.9781611972740.43

[32] James Cheng, Yiping Ke, Wilfred Ng, and An Lu. 2007. Fg-index: Towards verification-free query processing on graph databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'07)*. ACM, New York, NY, 857–872. DOI: https://doi.org/10.1145/1247480.1247574

[33] 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).

[34] Avery Ching, Sergey Edunov, Maja Kabiljo, Dionysios Logothetis, and Sambavi Muthukrishnan. 2015. One trillion edges: Graph processing at facebook-scale. *Proc. VLDB Endow.* 8, 12 (2015), 1804–1815.

[35] Marek Ciglan, Alex Averbuch, and Ladialav Hluchy. 2012. Benchmarking traversal operations over graph databases. In *Proceedings of the IEEE 28th International Conference on Data Engineering Workshops (ICDEW'12)*. IEEE Computer Society, Washington, DC, USA, 186–189. DOI: https://doi.org/10.1109/ICDEW.2012.47

[36] 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, 143–154. DOI: https://doi.org/10.1145/1807128.1807152

[37] Leon Danon, Albert Diaz-Guilera, Jordi Duch, and Alex Arenas. 2005. Comparing community structure identification. *J. Stat. Mech.: Theory Exper.* 2005, 9 (2005), P09008.

[38] Miyuru Dayarathna and Toyotaro Suzumura. 2014. Graph database benchmarking on cloud environments with XGDBench. *Autom. Softw. Eng.* 21, 4 (Dec. 2014), 509–533. DOI: https://doi.org/10.1007/s10515-013-0138-7

[39] 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 2nd TPC Technology Conference on Performance Evaluation, Measurement and Characterization of Complex Systems (TPCTC'10)*. Springer-Verlag, Berlin, 25–40. Retrieved from http://dl.acm.org/citation.cfm?id=1946050.1946053.

[40] 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 International Conference on Web-age Information Management (WAIM'10)*. Springer-Verlag, Berlin, 37–48. Retrieved from http://dl.acm.org/citation.cfm?id=1927585.1927590.

[41] P. Doreian and F. N. Stokman. 1997. *Evolution of Social Networks*. Number 1 in *The Journal of Mathematical Sociology*. Gordon and Breach Publishers. Retrieved from https://books.google.com/books?id=ZL4zCCgfmOkC.

[42] 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 ACM SIGMOD International Conference on Management of Data (SIGMOD'11)*. ACM, New York, NY, 145–156. DOI: https://doi.org/10.1145/1989323.1989340

[43] Cynthia Dwork and Jing Lei. 2009. Differential privacy and robust statistics. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*. ACM, 371–380.

[44] Cynthia Dwork, Aaron Roth, et al. 2014. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.* 9, 3–4 (2014), 211–407.

[45] 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).

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

[47] 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 ACM SIGMOD International Conference on Management of Data (SIGMOD'15)*. ACM, New York, NY, 619–630. DOI: https://doi.org/10.1145/2723372.2742786

[48] Wenfei Fan, Yinghui Wu, and Jingbo Xu. 2016. Functional dependencies for graphs. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'16)*. ACM, 1843–1857.

[49] Javier D. Fernández, Axel Polleres, and Jürgen Umbrich. 2015. Towards efficient archiving of dynamic linked open data. *Proc. Diachron* 1377 (2015), 34–49.

[50] Javier D. Fernández, Jürgen Umbrich, and Axel Polleres. 2015. BEAR: Benchmarking the efficiency of RDF archiving. Technical Report. TR no. 02/2015. Department of Information Systems and Operations, Vienna University of Economics and Business. https://epub.wu-wien.ac.at/4615/1/BEAR_techReport_022015.pdf.

[51] Alfio Ferrara, Davide Lorusso, Stefano Montanelli, and Gaia Varese. 2008. Towards a benchmark for instance matching. In *Proceedings of the Central Europe Workshop on Ontology Matching (OM'08)*, Pavel Shvaiko, Jerome Euzenat, Fausto Giunchiglia, and Heiner Stuckenschmidt (Eds.), Vol. 431. Retrieved from CEUR-WS.org.

[52] Santo Fortunato and Marc Barthelemy. 2007. Resolution limit in community detection. *Proc. Natl. Acad. Sci. U.S.A.* 104, 1 (2007), 36–41.

[53] George Garbis, Kostis Kyzirakos, and Manolis Koubarakis. 2013. Geographica: A benchmark for geospatial RDF stores (long version). In *Proceedings of the 12th International Semantic Web Conference (ISWD'13)*. 343–359. DOI: https://doi.org/10.1007/978-3-642-41338-4_22

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

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

[56] 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*. MIT Press, 2672–2680.

[57] GraphAnalysis.org. 2009. *HPC Scalable Graph Analysis Benchmark*. Retrieved from http://www.graphanalysis.org/benchmark/.

[58] Michael Grossniklaus, Stefania Leone, and Tilmann Zaschke. 2013. *Towards a Benchmark for Graph Data Management and Processing*. Technical Report.

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

[60] Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin. 2005. LUBM: A benchmark for {OWL} knowledge base systems. *Web Semant.: Sci. Serv. Agents WWW* 3, 2–3 (2005), 158–182. DOI: https://doi.org/10.1016/j.websem.2005.06.005

[61] Tim Hellmann and Mathias Staudigl. 2014. Evolution of social networks. *Eur. J. Operat. Res.* 234, 3 (2014), 583–596. DOI: https://doi.org/10.1016/j.ejor.2013.08.022

[62] Paul W. Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. 1983. Stochastic blockmodels: First steps. *Soc. Netw.* 5, 2 (1983), 109–137.

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

[64] 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: https://doi.org/10.14778/3007263.3007270

[65] ISO. 2008. *ISO/IEC 9075-1:2008 Information Technology–Database Languages–SQL–Part 1: Framework (SQL/Framework)*. Retrieved from http://www.iso.org/iso/catalogue_detail.htm?csnumber=45498.

[66] Amit Krishna Joshi, Pascal Hitzler, and Guozhu Dong. 2016. LinkGen: Multipurpose linked data generator. In *Proceedings of the International Semantic Web Conference (ISWC'16)*, Paul Groth, Elena Simperl, Alasdair Gray, Marta Sabou, Markus Krötzsch, Freddy Lecue, Fabian Flöck, and Yolanda Gil (Eds.). Springer International Publishing, Cham, 113–121.

[67] Jungeun Kim and Jae-Gil Lee. 2015. Community detection in multi-layer graphs: A survey. *SIGMOD Rec.* 44, 3 (Dec. 2015), 37–48. DOI: https://doi.org/10.1145/2854006.2854013

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

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

[70] Tamara G. Kolda, Ali Pinar, Todd Plantenga, and Comandur Seshadhri. 2014. A scalable generative graph model with community structure. *SIAM J. Sci. Comput.* 36, 5 (2014), C424–C452.

[71] Gueorgi Kossinets and Duncan J. Watts. 2006. Empirical analysis of an evolving social network. *Science* 311, 5757 (2006), 88–90. Retrieved from arXiv:http://science.sciencemag.org/content/311/5757/88.full.pdf. DOI: https://doi.org/10.1126/science.1116869

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

[73] 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, 611–617. DOI: https://doi.org/10.1145/1150402.1150476

[74] Mayuresh Kunjir, Prajakta Kalmegh, and Shivnath Babu. 2014. Thoth: Towards managing a multi-system cluster. *Proc. VLDB* 7, 13 (2014), 1689–1692.

[75] 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: https://doi.org/10.1103/PhysRevE.80.016118

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

[77] LDBC. 2015. *Semantic Publishing Benchmark v2.0*. Retrieved from http://ldbcouncil.org/developer/spb.

[78] 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 *Proceedings of the International Semantic Web Conference*. Springer, 300–312.

[79] Jurij Leskovec, Deepayan Chakrabarti, Jon Kleinberg, and Christos Faloutsos. 2005. 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, 133–145. DOI: https://doi.org/10.1007/11564126_17

[80] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2005. Graphs over time: Densification laws, shrinking diameters and possible explanations. In *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining (KDD'05)*. ACM, New York, NY, 177–187. DOI: https://doi.org/10.1145/1081870.1081893

[81] 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, 695–704. DOI: https://doi.org/10.1145/1367497.1367591

[82] 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. *Proc. VLDB Endow.* 1, 1 (2008), 274–288.

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

[84] Graph Aware Ltd.2015. *GraphGen: Graph Generator for Neo4j*. Retrieved from http://graphgen.graphaware.com/.

[85] Jiaheng Lu. 2017. Towards benchmarking multi-model databases. In *Proceedings of the 8th Biennial Conference on Innovative Data Systems Research (CIDR'17)*. Retrieved from http://cidrdb.org/cidr2017/gongshow/abstracts/cidr2017_20.pdf.

[86] Jiaheng Lu and Irena Holubová. 2019. Multi-model databases: A new journey to handle the variety of data. *ACM Comput. Surv.* 52, 3 (June 2019). DOI: https://doi.org/10.1145/3323214

[87] 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, 125–139. DOI: https://doi.org/10.1007/11762256_12

[88] 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.

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

[90] 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 *Proceedings of the International Semantic Web Conference*. Springer, 140–149.

[91] Andrew McGregor. 2014. Graph stream algorithms: A survey. *ACM SIGMOD Rec.* 43, 1 (2014), 9–20.

[92] 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'16) and the 3rd Workshop on Linked Data Quality (LDQ'16) colocated with 13th European Semantic Web Conference (ESWC'16)*. 20–35. Retrieved from http://ceur-ws.org/Vol-1585/mepdaw2016_paper_03.pdf.

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

[94] G. A. Miller. 1957. Some effects of intermittent silence. *Amer. J. Psychol.* 70 (1957), 311–314.

[95] George A. Miller. 1995. WordNet: A lexical database for English. *Commun. ACM* 38, 11 (Nov. 1995), 39–41. DOI: https://doi.org/10.1145/219717.219748

[96] Mohamed Morsey, Jens Lehmann, Sören Auer, and Axel-Cyrille Ngonga Ngomo. 2012. Usage-centric benchmarking of RDF triple stores. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI'12)*. AAAI Press, 2134–2140. Retrieved from http://dl.acm.org/citation.cfm?id=2900929.2901031.

[97] 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, 454–469. DOI: https://doi.org/10.1007/978-3-642-25073-6_29

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

[99] David F. Nettleton. 2016. A synthetic data generator for online social network graphs. *Soc. Netw. Anal. Min.* 6, 1 (2016), 44. DOI: https://doi.org/10.1007/s13278-016-0352-y

[100] Shirui Pan and Xingquan Zhu. 2013. Graph classification with imbalanced class distributions and noise. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI'13)*. 1586–1592.

[101] Vassilis Papakonstantinou, Giorgos Flouris, Irini Fundulaki, Kostas Stefanidis, and Giannis Roussakis. 2016. Versioning for linked data: Archiving systems and benchmarks. In *Proceedings of the Workshop on Benchmarking Linked Data (BLINK'16), Co-located with the 15th International Semantic Web Conference (ISWC'16)*. Retrieved from http://ceur-ws.org/Vol-1700/paper-05.pdf.

[102] Ashwin Paranjape, Austin R. Benson, and Jure Leskovec. 2017. Motifs in temporal networks. In *Proceedings of the 10th ACM International Conference on Web Search and Data Mining (WSDM'17)*. ACM, New York, NY, 601–610. DOI: https://doi.org/10.1145/3018661.3018731

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

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

[105] Arnau Prat-Pérez and David Dominguez-Sal. 2014. How community-like is the structure of synthetically generated graphs? In *Proceedings of the Workshop on GRAph Data Management Experiences and Systems (GRADES'14)*. ACM, New York, NY. DOI: https://doi.org/10.1145/2621934.2621942

[106] Arnau Prat-Pérez, David Dominguez-Sal, Josep-M. Brunat, and Josep-Lluis Larriba-Pey. 2016. Put three and three together: Triangle-driven community detection. *ACM Trans. Knowl. Discov. Data* 10, 3 (2016), 22.

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

[108] 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'18)*.

[109] Shi Qiao and Z. Meral Özsoyoğlu. 2015. RBench: Application-specific RDF benchmarking. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'15)*. ACM, New York, NY, 1825–1838. DOI: https://doi.org/10.1145/2723372.2746479

[110] 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 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 425–438.

[111] Carlos R. Rivero, Andreas Schultz, Christian Bizer, and David Ruiz. 2012. Benchmarking the performance of linked data translation systems. In *Proceedings of the Workshop on Linked Data on the Web (WWW'12)*. Retrieved from http://ceur-ws.org/Vol-937/ldow2012-paper-09.pdf.

[112] Sherif Sakr. 2013. Processing large-scale graph data: A guide to current technology. *IBM Developerworks* (2013), 15.
[113] Sherif Sakr. 2016. *Big Data 2.0 Processing Systems: A Survey*. Springer.
[114] Sherif Sakr, Faisal Moeen Orakzai, Ibrahim Abdelaziz, and Zuhair Khayyat. 2016. *Large-scale Graph Processing Using Apache Giraph*. Springer.
[115] Sherif Sakr and Eric Pardede (Eds.). 2011. *Graph Data Management: Techniques and Applications*. IGI Global. DOI: https://doi.org/10.4018/978-1-61350-053-8
[116] 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: https://doi.org/10.1007/978-3-319-25007-6_4
[117] 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. Retrieved from http://dl.acm.org/citation.cfm?id=1287369.1287455.
[118] Michael Schmidt, Thomas Hornung, Michael Meier, Christoph Pinkel, and Georg Lausen. 2010. *SP2Bench: A SPARQL Performance Benchmark*. Springer, Berlin, 371–393. DOI: https://doi.org/10.1007/978-3-642-04329-1_16
[119] Martin Simonovsky and Nikos Komodakis. 2018. GraphVAE: Towards generation of small graphs using variational autoencoders. *arXiv preprint arXiv:1802.03480* (2018).
[120] Hotea Solutions. 2016. *The TPC Benchmark-H*. Retrieved from http://www.tpc.org/tpch/.
[121] DBLP team. 2016. *DBLP Computer Science Bibliography*. Retrieved from http://dblp.uni-trier.de/.
[122] Riccardo Tommasini, Emanuele Della Valle, Andrea Mauri, and Marco Brambilla. 2017. RSPLab: RDF stream processing benchmarking made easy. In *Proceedings of the International Semantic Web Conference*. Springer, 202–209.
[123] 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. DOI: https://doi.org/10.1145/1900008.1900067
[124] 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, 37–42. DOI: https://doi.org/10.1145/1592665.1592675
[125] W3C. 2004. *OWL Web Ontology Language Overview*. Retrieved from http://www.w3.org/TR/owl-features/.
[126] 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: https://doi.org/10.4028/www.scientific.net/AMR.756-759.2634
[127] Sui-Yu Wang, Yuanbo Guo, Abir Qasem, and Jeff Heflin. 2005. *Rapid Benchmarking for Semantic Web Knowledge Base Systems*. Springer, Berlin, 758–772. DOI: https://doi.org/10.1007/11574620_54
[128] Xi Wang, Mahsa Maghami, and Gita Sukthankar. 2011. Leveraging network properties for trust evaluation in multi-agent systems. In *Proceedings of the IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*. IEEE Computer Society, 288–295.
[129] Yue Wang and Xintao Wu. 2013. Preserving differential privacy in degree-correlation based graph generation. *Trans. Data Priv.* 6, 2 (2013), 127.
[130] Huanhuan Wu, James Cheng, Silu Huang, Yiping Ke, Yi Lu, and Yanyan Xu. 2014. Path problems in temporal graphs. *Proc. VLDB Endow.* 7, 9 (May 2014), 721–732. DOI: https://doi.org/10.14778/2732939.2732945
[131] Hongyan Wu, Toyofumi Fujiwara, Yasunori Yamamoto, Jerven Bolleman, and Atsuko Yamaguchi. 2014. BioBenchmark toyama 2012: An evaluation of the performance of triple stores on biological data. *J. Biomed. Semant.* 5, 1 (2014), 32. DOI: https://doi.org/10.1186/2041-1480-5-32
[132] 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.
[133] Jaewon Yang and Jure Leskovec. 2015. Defining and evaluating network communities based on ground-truth. *Knowl. Info. Syst.* 42, 1 (2015), 181–213.
[134] Yuan Yao, Jiufeng Zhou, Lixin Han, Feng Xu, and Jian Lü. 2011. *Comparing Linkage Graph and Activity Graph of Online Social Networks*. Springer, Berlin, 84–97. DOI: https://doi.org/10.1007/978-3-642-24704-0_14
[135] Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. 2018. GraphRNN: Generating realistic graphs with deep auto-regressive models. In *Proceedings of the International Conference on Machine Learning*. 5694–5703.
[136] Yunpeng Zhao. 2017. A survey on theoretical advances of community detection in networks. *Wiley Interdisc. Rev.: Comput. Stat.* 9, 5 (2017). DOI: https://doi.org/10.1002/wics.1403 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/wics.1403

Received August 2019; revised December 2019; accepted January 2020