MŰEGYETEM 1782

Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Measurement and Information Systems

Performance Analysis of Graph Queries

Author:

Zsolt Kővári

Advisors:

Gábor Szárnyas
Dr. István Ráth

2015

Contents

Kivonat

A gráf alapú adatmodellező, illetve adatbázis-kezelő keretrendszerek esetén kulcsfontosságú a minél nagyobb teljesítmény, illetve rövidebb válaszidő biztosítása, különösen az olyan alkalmazási területeken, ahol egyszerre több, strukturálisan összetett lekérdezést kell újra és újra kiértékelni egy folyamatosan változó (gráf)adatstruktúra felett. A modern, relációs adatmodellt elvető NoSQL technológiák terjedésével egyre nagyobb figyelmet kap az ilyen rendszerek teljesítőképessége és skálázhatósága, így az utóbbi években több olyan benchmark is megjelent, melynek fő célja az ilyen rendszerek teljesítőképességének, különösen a lekérdezések skálázhatóságának szisztematikus kiértékelése.

A legtöbb mérési keretrendszer számításba vesz bemenet és teljesítmény leíró metrikákat, mint például a gráf csomópontjainak száma vagy a lekérdezések válaszideje, ugyanakkor nagyon nehéz az eredmények összehasonlítása, mert a méréshez használt bemenetek (gráfok, lekérdezések komplexitása, illetve a gráfon a mérés során végrehajtott változtatások jellege, összességében a *terhelési profil*) igen nagy méret- és tulajdonságbeli eltéréseket mutatnak - ezekről pedig megelőző kutatási eredmények kimutatták, hogy jelentős és változó mértékben befolyásolhatják az egyes eszközök viselkedését, teljesítménykarakterisztikáját.

A jelen dolgozat elsődleges célja, hogy - korábbi kutatásokat [31] folytatva - kidolgozzon egy olyan mérési módszertant és hozzá kapcsolódó keretrendszert, amelynek segítségével a gráf alapú adatbázis-kezelő rendszerek teljesítménybeli összehasonlítása szisztematikusan és reprodukálhatóan hajtható végre. Fő eredményként a teljesítménymérés céljára különböző gráf topológiákat javasolunk, amelyek jellemzésére gráfmetrikákat definiálunk. A metrikák segítségével jellemezhető az egyes mérések nehézsége, illetve bizonyos eszközök esetén kapcsolatot is találhatunk a gráfokat jellemző metrikák és a lekérdezések futásidői között. A módszertan és a keretrendszer képességeinek bemutatására a dolgozat bemutat egy komplex esettanulmányt, mely magába foglalja számos kísérlet automatizált elvégzését és magasszintű statisztikai analízis eszközökkel támogatott kiértékelését is.

Abstract

Achieving high query evaluation performance represents a major practical challenge in graph-based data management and database systems, especially in application domains where several, structurally complex queries need to be continuously evaluated against a steadily changing input graph. With the prevalence of modern NoSQL databases, the scalability of such systems is getting more and more attention. This resulted in the development of several graph-oriented database benchmarks over the past several years, with the main focus on the systematic assessment of query evaluation performance.

While most of such measurement frameworks take common input and performance metrics as the basis of comparison (e.g. graph node count, query evaluation response time), it remains very difficult to compare the relative difficulty of individual measurements to each other. This is because the characteristics of the input (graphs, queries and their complexity, and graph manipulation operations executed during the measurements, combined the *workload profile*) varies greatly between individual measurement scenarios, and such factors have been shown by previous research results to have a significant and varied effect on tool performance characteristics.

As a continuation of previous research [31], the primary goal of this report is to establish a measurement methodology with its associated set of frameworks and tools, which can help in the systematic and reproducible assessment of performance and scalability of graph-oriented database systems. As the main result, we propose to use a carefully designed corpus of input graphs adhering to various topologies characterized by metrics. These metrics are useful to compare the relative difficulty of measurement scenarios, and for some tools we can use them to establish links between metric values and query evaluation response time. In order to illustrate the capabilities of the methodology and the framework, the report includes a complex case study that incorporates the automatic execution of many experiments, as well as statistical result analysis supported by high level tools.

Chapter 1

Introduction

1.1 Context

Queries are the foundation of data-intensive applications. Therefore, a high-performance query engine is an essential component for a wide range of software systems, including transactional databases, knowledge-based systems and software engineering tools. Traditionally, most of the data was stored in relational databases. However, during the last decade a new generation of databases emerged, utilizing non-relational models. These systems, collectively known as NoSQL databases, include semantic databases (triplestores) and graph databases. As graphs are well-suited to model domains with a rich inner structure—e.g. social networks, public roads and library data—these databases are commonly used in both academic and industrial systems.

1.2 Problem Statement

While there are well-established and widely accepted benchmarks for relational databases [16], these cannot be adapted to graph databases as they usually operate on fundamentally different workloads. Semantic databases have been widely studied, considering different aspects of performance, correctness and completeness [42, 21, 35]. In the model-driven engineering (MDE) community, tool contests presented transformation cases to assess the usability, conciseness and performance of model transformations [27, 41, 45, 44, 33]. The NoSQL community also proposed various benchmarks, published on both the web [40, 43] and in research papers [22, 34].

While all these benchmarks can be used to compare the performance of query engines, deriving general conclusions from the results or comparing the findings between different benchmarks is difficult. The generalizability of benchmark results is mostly limited by the lack of relevant metrics that could be used to assess an engineering problem and predict which technology would be best suited. Existing metrics emphasize a single aspect of the problem (most typically the size of the graph), while internal metrics (e.g. used for optimizing query evaluation engines) are either not documented well or not accessible in a reusable way.

1.3 Contributions

Our research aims to provide a framework for categorizing various benchmarks. In this report, we present a set of common metrics to characterize the complexity of the graph and the queries. We designed and implemented a benchmark for homogeneous graphs, featuring a set of queries and a highly configurable framework capable of generating graphs in different sizes and various topologies. We make use of statistical methods to determine the relationship between the metrics and the performance of the databases.

1.4 Structure of the Report

The structure of the report is as follows. Chapter 2 introduces the theoretical concepts, Chapter 3 discusses the design of the framework, and Chapter 4 presents the architecture and workflow of the benchmark. Chapter 5 shows the benchmark setup and analyzes the benchmark results. Chapter 6 concludes the report and outlines directions for future work.

Chapter 2

Background

2.1 Resource Description Framework

The Resource Description Framework (RDF [12]) is a family of W3C (World Wide Web Consortium) specifications originally designed as a *metadata data model*.

The RDF data model is based on the idea of making statements about *resources* in the form of triples. A triple is a data entity composed of a *subject*, a *predicate* and an *object*, e.g. ”John instanceof Person”, ”John has-an-age-of 34”.

Triples are typically stored in *triplestores*, specialized databases tailored to store and process triples efficiently. Also, some triplestores are capable of *reasoning*, i.e. inferring logical consequences from a set of facts or axioms. Triplestores are mostly used in the semantic and Linked Data domain.

2.2 Foundations of Statistics

In this section, we present the fundamental concepts of the field of statistics. The majority of the definitions can be found in [23].

2.2.1 Population and Sample

A **population** is a large set of objects of a similar nature, and the **sample** is a subset of objects derived from a population [9]. A population includes all of the elements from a set of data, however, a sample consists of one or more observations from the population [10]. The sample size is the number of observations in a sample and commonly denoted by n.

2.2.2 Probability Distributions

The probability distribution links each possible value of a random variable [11] with its probability of occurrence. In the following paragraphs we define various discrete probability distributions related to our work.

Uniform Distribution

A random variable X has a discrete uniform distribution if each of the n values in its range, namely, x_1, x_2, ..., x_n, have equal probability. Formally:

$$f(x_i) = \frac{1}{n} \tag{2.1}$$

where $f(x_i)$ denotes the probability distribution function.

Power-Law Distribution

The power-law distribution describes the probability of the random variable that is equal to x as the following

$$f(x) = c \cdot x^{-\gamma} \tag{2.2}$$

where c is a constant and γ is called the *exponent* or *scale factor*.

Poisson Distribution

The Poisson distribution is characterized by the following elementary probabilities:

$$P(X = k) = \frac{\lambda^k}{k!} e^{-\lambda} \tag{2.3}$$

where $\lambda > 0$ is the shape parameter and $k \in \mathbb{N}$.

2.2.3 Measures of Descriptive Statistics

Descriptive statistics defines a number of values for characterizing a particular data set. Here, we discuss the most important measures related to our work.

Mean

The *mean* or *expected value* of a discrete random variable X denoted by μ or $E(X)$ is

$$\mu = E(X) = \sum_x x f(x) \tag{2.4}$$

where x represents the values of the random variable X, and $f(x)$ denotes the probability distribution function. A mean is a measure of the center of the probability distribution.

Variance

The *variance* of X—denoted by σ^2 or $V(X)$—is equal to the following formula:

$$\sigma^2 = V(X) = E(X-\mu)^2 = \sum_x (x-\mu)^2 f(x) \tag{2.5}$$

The variance is a measure of the dispersion or variability in the distribution, as it represents the average of the squared differences from the mean. For example, a variance of zero indicates that all the values are identical.

Standard Deviation

The *standard deviation* of the random variable X is $\sigma = \sqrt{\sigma^2}$, meaning it is the square root of the variance.

2.2.4 Covariance and Correlation

Covariance

The *covariance* is a measure of the linear relationship between random variables. A covariance between two random variables X and Y can be expressed as follows:

$$cov(X,Y) = E\big[(X-\mu_X)(Y-\mu_Y)\big] \tag{2.6}$$

A positive covariance implies that Y tends to increase as X increases as well, and if $cov(X,Y) < 0$ then Y tends to decrease as X increases [5]. A few examples of different scenarios are illustrated in Figure 2.1. In the terms of (a) and (c), a covariance is observable between X and Y, and in the cases of (b) and (d), the covariance is equal to zero.

Correlation

Similarly to covariance, the *correlation* describes the strength of the relationship between variables. A type of correlation, the Pearson product-moment correlation coefficient is formulated as

$$\rho_{XY} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} \tag{2.7}$$

where $-1 \leq \rho_{XY} \leq 1$, and 1 indicates a positive linear relationship between X and Y, and -1 means negative linearity, finally, 0 implies a correlation does not appear between the variables.

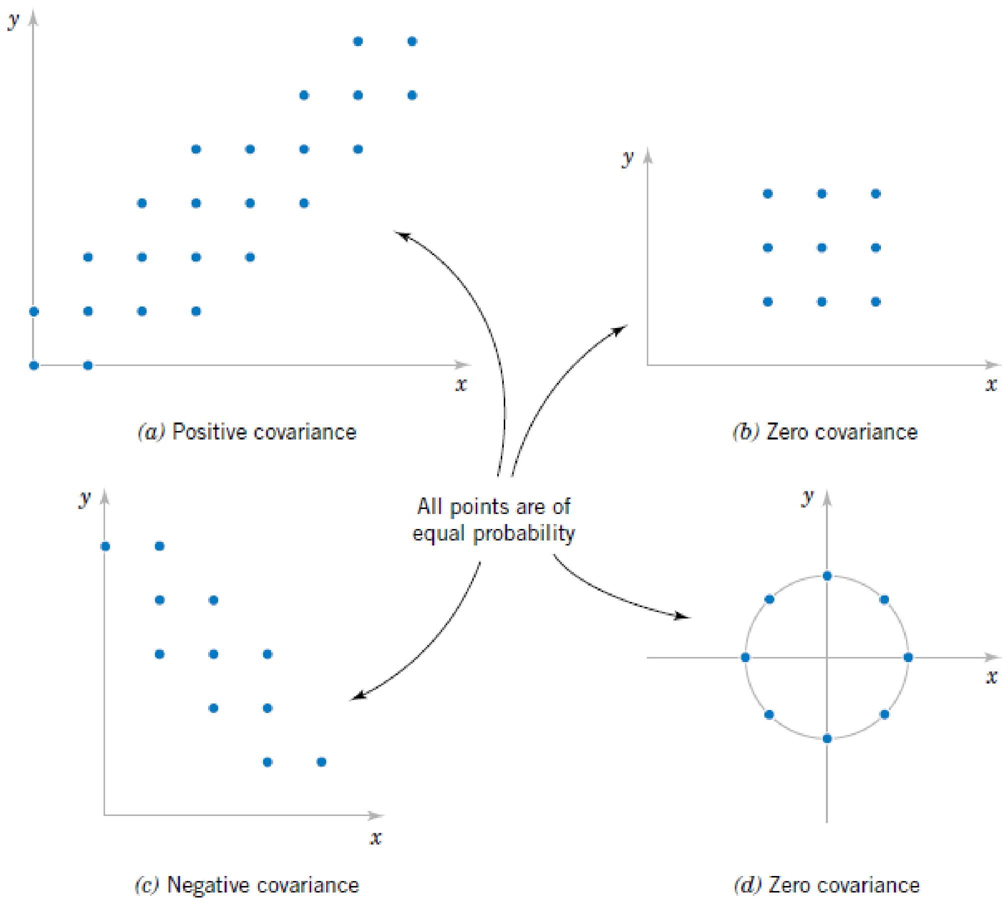


Figure 2.1. Different examples for covariance [23].

2.2.5 Regression Analysis

Regression analysis is a statistical technique for exploring the relationship between two or more variables. A regression model can be considered as an equation that relates a random variable Y to a function of a variable x and a constant β. Formally, a regression model is defined as

$$Y = \beta_0 + \beta_1 x + \epsilon \tag{2.8}$$

where Y is the dependent or response variable, x is referred as the independent variable or predictor, and β_0, β_1 are the regression coefficients—the intercept and the slope. Finally, ϵ symbolizes the random error.

A multiple linear regression model considers k independent variables, and the equation is extended as follows:

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k + \epsilon \tag{2.9}$$

Coefficient of Determination

The coefficient of determination—or the multiple R-squared, denoted by R^2—is a number between 0 and 1 that expresses how appropriately a regression model fits to the data. An R^2 of 1 represents a perfect fit.

P-value

The p-value is a function of the observed sample results (a statistic) that is used for testing a statistical hypothesis.

2.3 Graph Theory

In the following sections we introduce the most important graph metrics and network topologies used in our work. We assume that the reader is already familiar with the basic concepts of graph theory, including undirected graphs, complete graphs, adjacent nodes, node degrees and shortest paths.

2.3.1 Metrics

Degree Distribution

The spread among the degrees of the nodes is characterized by a distribution function $P(k)$ which shows the probability that a randomly selected node's degree is equal to k. $P(k)$ is called as the degree distribution.

Density

For undirected graphs the density is defined as

$$D = \frac{2|E|}{|V|(|V|-1)} \tag{2.10}$$

where $|E|$ denotes the number of edges and $|V|$ represents the number of nodes in the graph.

Clustering Coefficient

The C_n clustering coefficient of a node n is equal to the proportion of connections found among the neighbors of n divided by the maximum number of connections that can be possibly exist between them. Formally, in undirected graphs the clustering coefficient of node n is

$$C_n = \frac{2e_n}{k_n(k_n - 1)} \tag{2.11}$$

where k_n is the number of neighbors of n, and e_n is the number of connected pairs between all neighbors of n [18]. The clustering coefficient always takes a value between 0 and 1.

An example is shown in Figure 2.2. In this case, the clustering coefficient of A is $C_A = \frac{2}{6}$, since it has three neighbors, so $k_A = 3$, and only one connection occurs among its adjacent nodes—between B and C—indicating that $e_A = 1$.

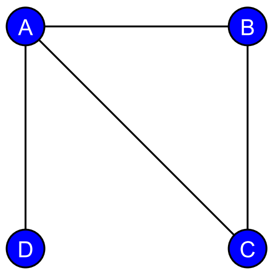


Figure 2.2. An example graph for illustrating the calculation of the clustering coefficient metric.

Betweenness Centrality

The betweenness centrality of an node n is quantified by the number of shortest paths that include n as an intermediate node, divided by the entire number of shortest paths. Consequently, this metric is normalized between 0 and 1.

To demonstrate with an example, assume that we searched the following shortest paths denoted by P_i:

- $P_1 = (v_1, v_3, v_4, v_6, v_5)$
- $P_2 = (v_2, v_3, v_4, v_5)$
- $P_3 = (v_4, v_3, v_5)$

The entire number of shortest paths is equal to 3, and the intermediate nodes are v_3, v_4 and v_6. The betweenness centrality values of the nodes—denoted by B_i—are the following:

- $B_3 = 1$, since v_3 appears in all three paths.
- $B_4 = \frac{2}{3}$ due to v_4 appears in two paths—P_1 and P_2. Note that v_4 in P_3 is the initial node and not an intermediate node.
- $B_6 = \frac{1}{3}$
- $B_1 = B_2 = B_5 = 0$, since they do not appear in the paths as intermediate nodes.

2.3.2 Network Topologies

In the following sections, we introduce the graph topologies that are relevant to our work. Besides their generation algorithms, we also emphasize the degree distributions they follow.

Random Graph

The main concept of the random graph is to create the connections among nodes independently from each other, meaning that the occurrence of an edge between the nodes is not influenced by the other edges.

Two well-known algorithms exist to create random graphs. The first is the $G(N, M)$ model of Erdős-Rényi [24], and the second is the $G(N, p)$ model from Gilbert [26]. The former means that precisely M edges exist among N vertices, and the latter implies that—in the generation algorithm—every pair of nodes becomes adjacent with p probability. The degree distribution of random graphs follows a Poisson distribution.

Small-World Model of Watts-Strogatz

A graph follows a small-world property if the graph has a high average clustering coefficient and small average length of shortest paths.

The generation algorithm of the Watts-Strogatz topology addresses the creation of networks with small-world properties. The algorithm is constructed as follows: initially, the algorithm creates a ring of N number of isolated nodes, which is equal to the entire number of nodes in the graph. In the second step, every node becomes adjacent to K number of their neighbors, thus creating a lattice graph [46]. This implies that every node has a degree K, therefore, its degree distribution fits to a uniform distribution. Besides the variables N and K, another parameter appears in the algorithm, the p probability variable. After creating N nodes and $N \cdot K$ connections, every edge is rewired by p probability and attached to a new, randomly chosen node. Note that the two extremes, $p = 0$ and $p = 1$ entails we obtain a mapping between a lattice and a random graph. As a result, the degree distribution of the Watts-Strogatz model deviates between uniform and Poisson.

Scale-Free Model of Barabási and Albert

The scale-free model of Barabási and Albert addresses the generation of a network that follows a power-law degree distribution and includes a small proportion of nodes that have significantly higher degrees than the average. These types of vertices are often referred as *hubs*.

The generation algorithm creates nodes incrementally and connects them to m disjunct nodes. However, instead of choosing nodes randomly, these new connections per nodes are determined by a *preferential attachment*. When the algorithm creates a new vertex then the probability p_i that this vertex becomes adjacent to an i node is:

$$p_i = \frac{d(i)}{\sum_j d(j)} \tag{2.12}$$

where $d(i)$ denotes the degree of the i node, and j symbolizes the other existing nodes. As a conclusion, the probability of a node becomes adjacent to another one depends on the degree of the latter. If a higher degree belongs to a node than the average, the probability also increases that the node obtains more connections.

Hierarchical Network

The hierarchical network topology [39] is generated by a recursive algorithm illustrated in Figure 2.3. Initially, the 0. iteration constructs a K_5 complete graph[1], called *cluster*. In the first iteration, the algorithm creates four replicas of the K_5 cluster. In the second step in this iteration, the algorithm connects the peripheral nodes from the replicas to the center node. The generation can be continued recursively, as in every *i*. run, the result graph of the $i-1$ iteration is cloned and the peripheral nodes—the *deepest* vertices in the replicas—are attached to the center node.
Finally, the generated hierarchical graph follows a heavy-tail power-law degree distribution, since the graph includes such nodes that have significantly larger degrees—the center nodes—and the probability that these nodes appear is considerably small.

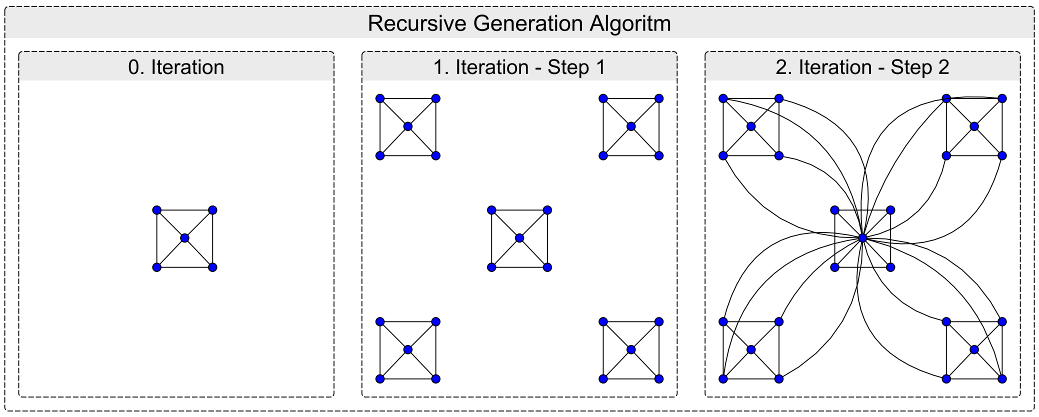


Figure 2.3. The first iteration in the recursive generation algorithm of the hierarchical network.

[1]The diagonal nodes are also connected to each other

Chapter 3

Related Work

In the following sections, we introduce benchmark frameworks that are designed to investigate the performance of RDF databases. In Section 3.5, we summarize the frameworks and compare them to our work which is the introduction of a new approach to existing benchmark frameworks.

3.1 Berlin SPARQL Benchmark

Domain The Berlin SPARQL Benchmark (BSBM) [21] features an e-commerce use case in which a set of products is offered by different vendors and consumers who posted reviews about the products.

Workload BSBM measures the SPARQL query performance of RDF-based tools via realistic workloads of queries based on the use cases. The benchmark defines three different use cases and a suite of benchmark queries—a *mix* of queries—in each of them [2], simulating the search and navigation pattern of a consumer looking for a product. The queries include read and update operations as well.

Models BSBM generates artificial models in different sizes, by using normal distributions among the elements. For example, the number of reviews and different properties per product are distributed following a normal distribution.

Metrics The benchmark defines certain performance metrics that relate to the query execution times from different perspectives. The most important metrics are the following:

- *Queries per Second*: the number of queries executed within a second.
- *Query Mixes per Hour*: the number of mixed queries with different parameters that evaluated within an hour.
- *Overall Runtime*: the overall time that a certain amount of query mix requires.

3.2 DBpedia SPARQL Benchmark

The DBpedia SPARQL Benchmark (DBPSB) proposes a benchmark framework for RDF databases based on the DBpedia [6] knowledge base. It measures the performance of real queries that were issued against existing RDF data [35]. DBPSB generates models in various sizes trying to obtain a similar characteristic belonging to the original DBpedia dataset.
Similarly to BSBM, DBPSB also defines metrics to investigate the performance from different aspects, such as the *Query Mixes per Hour* and *Queries per Second*.

3.3 SP^{2}Bench

The SPARQL Performance Benchmark (SP^{2}Bench) [42] is based on the *Digital Bibliography and Library Project* (commonly known as *dblp*) which provides an open bibliographic information on major computer science publications [7]. The benchmark queries are not explicitly evaluated over the *dblp* dataset, since SP^{2}Bench uses arbitrarily large, artificially generated models for the measurements that are created to follow the characteristics of the original *dblp* dataset, such as the power-law distribution.

SP^{2}Bench is designed to test the most common SPARQL constructs, operator constellations and a broad range of RDF data access patterns. Instead of defining a sequence of use case motivated queries, the framework proposes various systematic queries that cover specific RDF data management approaches.

Similarly to BSBM, SP^{2}Bench also measures additional performance related metrics besides the evaluation time, such as the disk storage cost, memory consumption, data loading time and success rates, hence, every metric captures different aspects from the evaluations.

3.4 Train Benchmark Framework

3.4.1 Overview

The Train Benchmark framework was designed and implemented by the Fault Tolerant Systems Research Group in the Budapest University of Technology and Economics [30]. The benchmark investigates the performance of model validations via different graph queries by measuring the evaluation times of different RDF, SQL, EMF and graph databases.

An overview of the Train Benchmark framework is shown in Figure 3.1, illustrating the framework how connects the model validations to the database systems. In step 1, the framework generates a graph-based model derived from a particular domain. After defining different constraints (rules) on the domain—such as the presence of an edge between two types of nodes—the benchmark injects erroneous elements into the model (step 2.),

which elements are considered as violations of the constraints. During step 3, the framework loads the invalid model to the particular database system.
Every constraint has a corresponding validation pattern that is the negation of the constraint and it includes a pattern of elements that violates the certain constraint (4.). Every validation pattern is defined in the query language of the particular database (5). Step 6 denotes the query evaluation that represents the model validation. The result set of the evaluation contains the invalid elements which violated the constraint. The performance indicator of the databases is the required time of model validation, so the query evaluation time.
Besides the validations, the framework also performs model transformations to alter the amount of invalid elements in the model. Every transformation is followed by another validation.

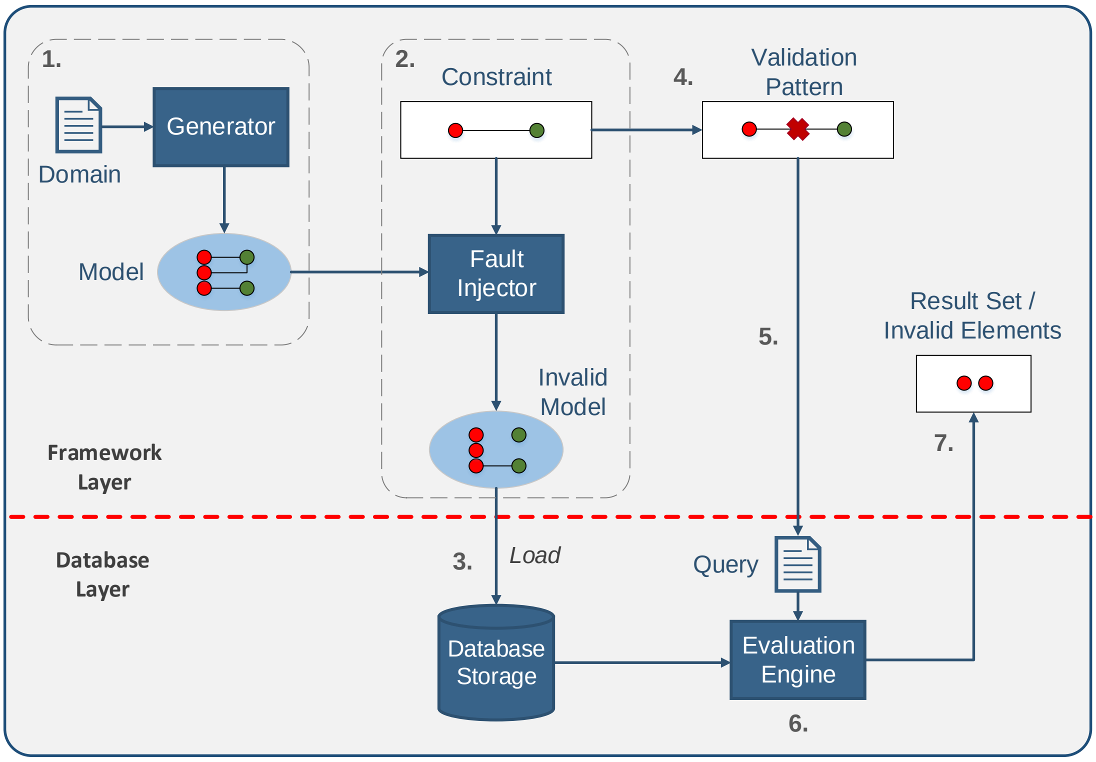


Figure 3.1. An overview of Train Benchmark.

3.4.2 Model Generation

The Train Benchmark framework uses artificially generated graph-based models for the measurements over a *railway* domain, illustrated in Figure 3.2. In the models, a train route is defined by a sequence of sensors, and the sensors are associated with track elements which are either segments or switches. A route follows certain switch positions which describe the required state of a switch belonging to the route. Each route has a semaphore on its entry and exit [15].

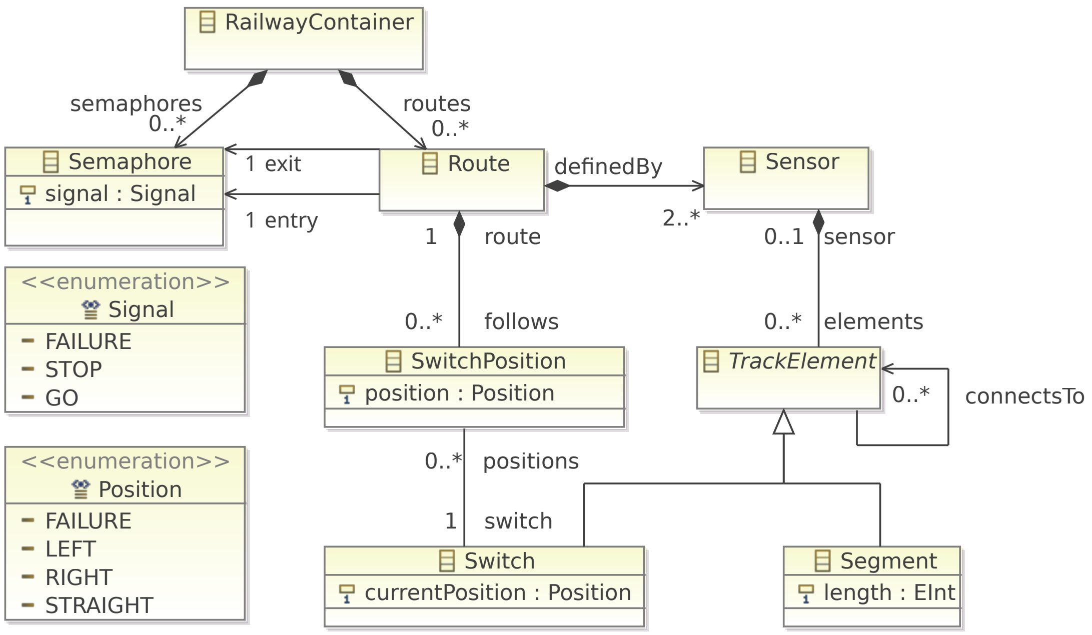


Figure 3.2. The Railway domain of Train Benchmark.

The generated models over the Railway domain are not related to a real-life model, which leads to the fact that the cardinalities of the elements and their distributions do not follow a real model's characteristic, therefore, the measurement results of different databases cannot be claimed to be representative in a real-life use case.

3.4.3 Metric-Based Analysis

The Train Benchmark framework was extended with metric-based analysis in [31] with the fundamental goal of finding correlations between (i) the performance of query evaluations and (ii) query and model metrics.

3.5 Conclusion

The represented benchmark frameworks propose different comprehensive use cases to assess the performance of graph queries typically over RDF data. BSBM and DBpedia use representative queries for the measurements that demonstrate real-life use cases, and the SP^{2}Bench framework concentrates on the creation of various systematic queries that investigate the specific RDF data management approaches. The Train Benchmark framework aims a different aspect of performance by concentrating on model validations via graph queries. As a conclusion, these frameworks guarantee a comprehensive performance evaluation of a workload—the model, query and tool—by emphasizing the impact of *queries* to the performance.

However, in case of an arbitrary workload, the *model* also represents a dominating factor to the performance. DBpedia or the SP^{2}Bench framework do not consider model modifications in their workloads, so they do not investigate the impact of various model characteristics to the performance, they only generate the same structural model in different sizes to measure scalability. BSBM proposes update operations on the models, still, it does not investigate model and performance correlations. Even though Train Benchmark considers model transformations, yet, it modifies only one type of element that impacts the result set of the query evaluation, and the generated models are still considered as static models. As a conclusion, all of the introduced frameworks concentrate on the generation of one static model without altering its internal structure and analyzing their influence to the performance.

The fundamental goal of our work is to extend the metric-based analysis in the Train Benchmark with the particular emphasis on the generation of different models and analyze the relationship between the model-based metrics and the performance of query evaluation.

The main features of the introduced frameworks and our new approach are summarized in Table 3.1.

Feature	BSBM	DBpedia	SP^{2}Bench	Train Benchmark	Our Approach
Based on a real-life model		●	●		
Realistic domain-based queries	●	●			
Systematic queries			●		
Model updates	●			●	
Various models					●
Measurement of scalability	●	●	●	●	●
Performance metrics	●	●	●		
Model metrics				●	●
Various representation formats				●	

Table 3.1. A comparison of existing benchmark frameworks and our approach.

Basically, we concentrate on the generation of various models with different internal networks, and we analyze the models and their impact to performance of query evaluations.

Chapter 4

Design

4.1 Overview of the Approach

In this section, we introduce the main notions about our work to analyze model and performance relationships.

Our concept includes the following systems illustrated in Figure 4.1. We rely on two existing frameworks in our work: the Train Benchmark and MONDO-SAM[1] frameworks. Furthermore, we the elaborate MONDO-MAP (MONDO-Metrics-Based Analysis of Performance)[2] and the Homogeneous Graphs Benchmark.

MONDO-SAM The MONDO-SAM framework was created under the project MONDO (Scalable Modeling and Model Management on the Cloud) [8] with the motivation of providing a common benchmark framework in Model Driven Engineering (MDE) for benchmark developers [29]. MONDO-SAM can be considered as an abstract layer that proposes an evaluation engine to execute arbitrary workflows independently on the current workload. Furthermore, MONDO-SAM also provides tools for serializing and visualizing the benchmark results.

Train Benchmark The Train Benchmark framework is based on the evaluation engine provided by MONDO-SAM, and it proposes a benchmark for measuring continuous model validations and transformations. The Train Benchmark is presented in detail in Section 3.4.

MONDO-MAP The goal of the MONDO-MAP framework is to support model-based performance analysis. To achieve this, it extends MONDO-SAM with additional features, as it provides a framework for analyzing model and performance relationships in the light of an arbitrary workload by characterizing the performance quantitatively with model

[1]The project of MONDO-SAM can be found in `http://github.com/ftsrg/mondo-sam`
[2]The project of MONDO-MAP can found in `http://github.com/ftsrg/mondo-map`

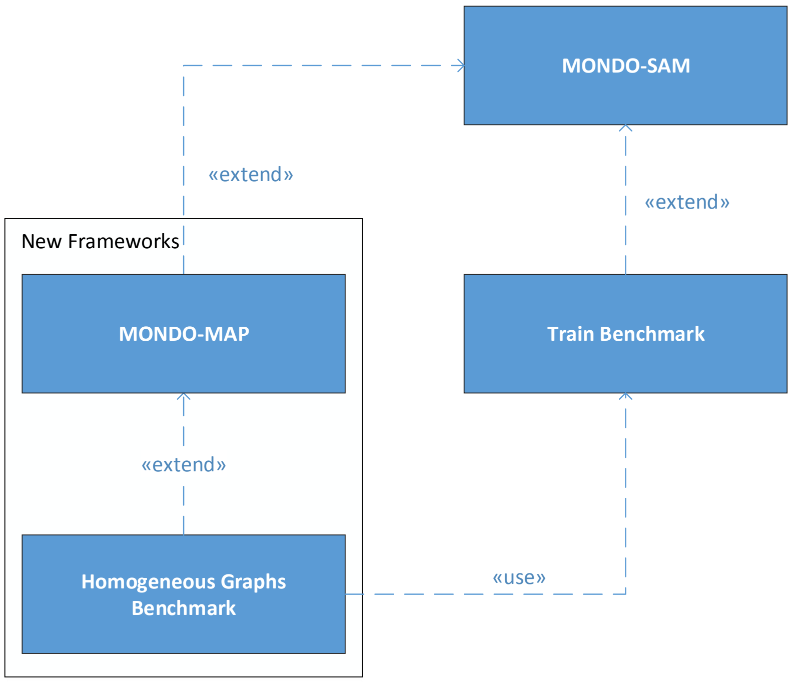
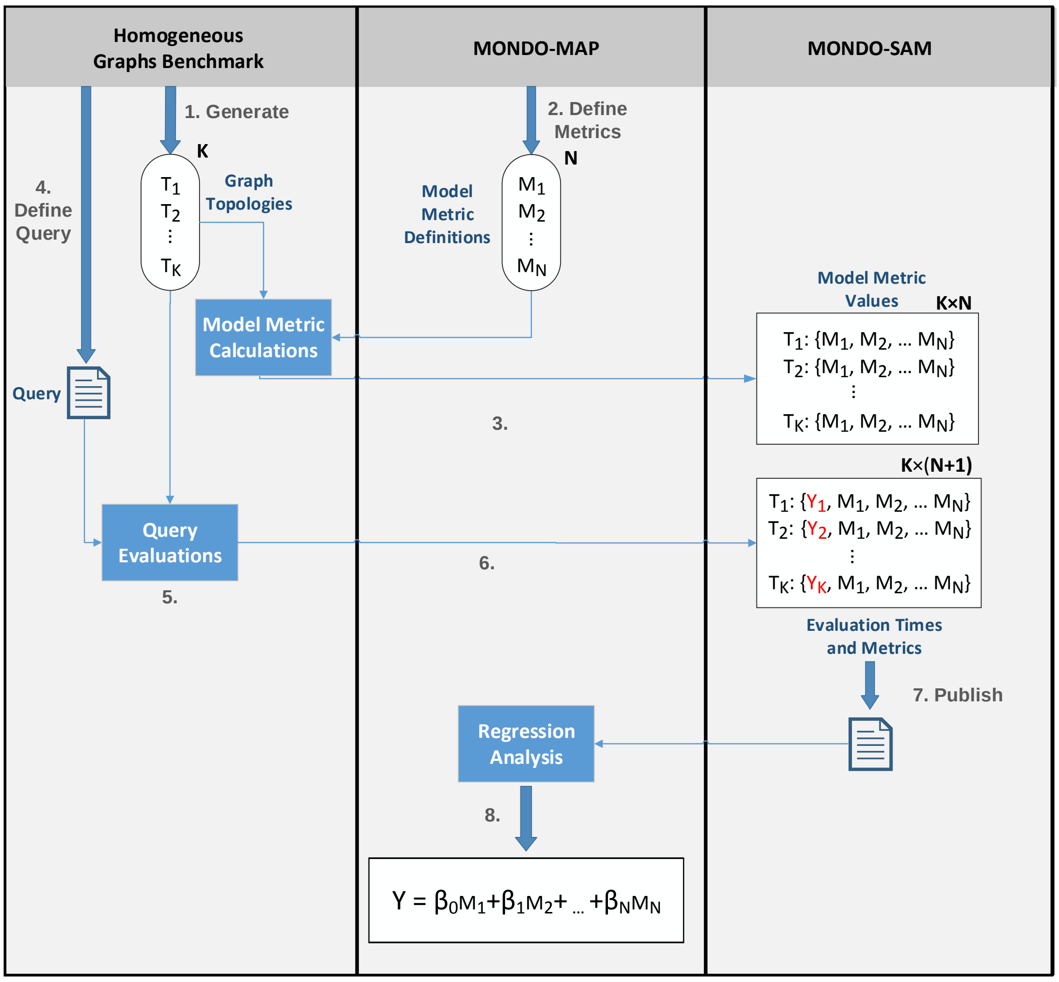


Figure 4.1. An overview of the frameworks in our approach.

related metrics. Similarly to MONDO-SAM, MONDO-MAP is an abstract framework and can be extended by an arbitrary benchmark case.

Homogeneous Graphs Benchmark The Homogeneous Graphs Benchmark (HG Benchmark) extends the MONDO-MAP framework with a benchmark case for RDF databases. It generates homogeneous graphs—well-known network topologies—and it investigates the relationships between the model metrics and performance of query evaluations with respect to an arbitrary query and tool. The goal is to generate artificial graphs with various characteristics and obtain a spread in their descriptive metrics, and thus showing a quantitative connection between model metrics and performance. As Figure 4.1 suggests, in the HG Benchmark we use a part of the components of the Train Benchmark that are adequate for our purpose as well.

Figure 4.2 illustrates the main concept of our work. First, the HG Benchmark generates different K graph topologies. Using the graph metric definitions from MONDO-MAP (2.), we calculate the descriptive metrics for every topology in step 3. In the next two steps (4-5.) we define a query and evaluate it on every topology in the system under benchmark. The measurement result is the query evaluation time (Y) that represents another variable belonging to the topologies besides their metrics (6.). As it was mentioned before, the MONDO-SAM framework is responsible for publishing the benchmark results (7.). Finally, in MONDO-MAP we analyze the results by creating regression models to estimate the influence of metrics to the performance.



Homogeneous Graphs Benchmark
MONDO-MAP
MONDO-SAM
1. Generate
K
T_1 T_2 ⋮ T_K
Graph Topologies
2. Define Metrics
N
Model Metric Definitions
M_1 M_2 ⋮ M_N
4. Define Query
Model Metric Calculations
Model Metric Values
K×N
T_1: {M_1, M_2, … M_N}
T_2: {M_1, M_2, … M_N}
T_K: {M_1, M_2, … M_N}
Query
3.
K×(N+1)
T_1: {Y_1, M_1, M_2, … M_N}
T_2: {Y_2, M_1, M_2, … M_N}
T_K: {Y_K, M_1, M_2, … M_N}
Query Evaluations
6.
5.
Evaluation Times and Metrics
7. Publish
Regression Analysis
8.
$Y = \beta_0 M_1 + \beta_1 M_2 + \dots + \beta_N M_N$

Figure 4.2. The main concept of the metric-based performance analysis.

4.2 Models and Metrics

4.2.1 Real-Life Networks

In the field of graph theory, the internal structures of real-life networks are comprehensively investigated. The main approach in the analysis of these networks is to explore the degree distributions and study the specific metrics—typically the clustering coefficients, average degrees, average shortest paths—that are suited to characterize the graphs appropriately. Based on the degree distributions and metrics, one can draw a conclusion how a particular real-life network shows a similar characteristic to the well-known topologies such as the random graph, scale-free model, small-world model of Watts-Strogatz or the hierarchical network.

For example, the network of World-Wide-Web is studied in [38] and [28] as well, and the authors observe that the degree distribution of the *www* follows a power-law distribution with a heavy tail, which indicates the presence of web pages with significantly higher degrees than the average degree. Since the probability of occurrences of these web pages is considerably low, the connectivity of the world-wide-web can be represented by the scale-free model of Barabási and Albert.

More examples can be found in the study of Barabási and Albert [19], as they review the advances of different publications and investigate the characteristics of different real-life networks. Empirical results prove that the *movie actor collaboration network*, *cellular networks*, *phone call* and *citation* networks also follow power-law distributions. Many of the studied networks can be considered as scale-free models, however, a part of these graphs—for example the network of movie actors—also show small-world properties and a high clustering coefficient in their connectivity similarly to the Watts-Strogatz or hierarchical topology.

As far as the Watts-Strogatz model—and the random graph—are concerned, their specific degree distribution—Poisson—rarely appears in the real-life networks, as it is emphasized in [37]. However, the small-world property of the WS models frequently appears in the real-life networks. In practice, none of the artificial topologies can be identified perfectly to real-life models, however, the representative metrics of these artificial networks can be observed in real-life networks as well.

We assume that those metrics that show high deviations among the different topologies are suited to characterize and identify them individually, furthermore, they may able to characterize an arbitrary graph as well. If these metrics are capable to characterize entirely different networks, then we assume that these metrics are the key to characterize the performance of query evaluations.

4.2.2 Network Topologies and Representative Metrics

Barabási and Albert inspect the natures of the well-known graph topologies in [19], such as the random graph, scale-free and the Watts-Strogatz model. As a main result, they observe that there are significant differences among the topologies regarding specific graph metrics. Based on their study and the research of hierarchical graphs [39], the following metric deviations are assumed between the four topologies, illustrated in Table 4.1[3]. The random graph is considered as a reference point, and every value is compared to its metrics by assuming that the networks are in the same size. As Table 4.1 demonstrates, each

Metric	Random	Hierarchical	Scale-free	Watts-Strogatz
Max Degree	●	●●●	●●●	●
Clustering Coefficient	●	●●●	●	●●
Avg Shortest Path Length	●	●●	●	●●

Table 4.1. Graph topologies and their descriptive metrics

topology can be characterized by different metric values, which leads to the assumption that if the diversity between the topologies may cause high variance in the performance of

[3]The metric values are compared to each other, as one ● indicates the lowest metric value, and ●●● denotes the largest value among the topologies.

a particular query evaluation, then these metrics are adequate to characterize the model and performance relationships quantitatively.

However, the values of metrics in Table 4.1 are misleading due to the reason that the metrics of the Watt-Strogatz model highly depend on the initialization of the network, namely, the value of p probability that is used in the generation.

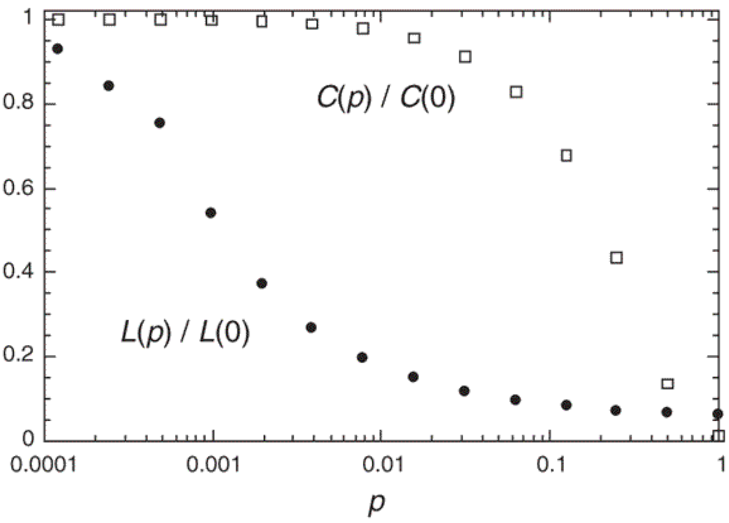


Figure 4.3. Characteristic path length $L(p)$ and clustering coefficient $C(p)$ of Watts-Strogatz model [47]

By modifying p, the Watts-Strogatz model represents a bridge between a lattice and a random graph. As Figure 4.3 illustrates, the clustering coefficient $C(p)$ and the average shortest path ($L(p)$) metrics are changed with respect to p scaling. The values are normalized by $L(0)$ and $C(0)$ that represent the clustering coefficient and average shortest path metrics for a lattice graph. As a conclusion, the Watts-Strogatz model shows significant deviations in these two metrics in the light of the p probability value.

Besides the models in Table 4.1, the metrics also require modifications. The problem is that the maximum degree metric alone does not include a comprehensive information about the internal structure of the network, since it does not emphasize the role of a node with maximum degree in the connectivity of the graph. Hence, we use another metric—the *betweenness centrality*—which characterizes adequately the importance of a higher degree, since the higher value of betweenness centrality belongs to a vertex, the more shortest paths include that node, symbolizing that node represents the center in the graph.

After these modifications, the topologies and the related metrics are shown in Table 4.2. WS-p indicates the Watts-Strogatz model with a probability value of p. The values of the betweenness centrality are determined by our initial assumption considering that the *center* node in a hierarchical network and the *hubs* in a scale-free model may occur more times in the shortest paths due to the fact that they have higher degrees. The main conclusion is that we can achieve a higher deviation among the metric values by using these topologies, and thus, in the following we concentrate on these networks.

Metric	Random	Hierarchical	Scale-free	WS-0.1	WS-0.01	WS-0.001
Max Degree	●	●●●	●●●	●	●	●
Clustering Coefficient	●	●●●	●	●●	●●●	●●●
Avg Shortest Path Length	●	●●	●	●	●●	●●●
Betweenness Centrality	●	●●●	●●	●	●	●

Table 4.2. Graph topologies and their descriptive metrics with extensions

4.3 Metric and Performance Comparison

Showing performance and metric relationship is an essential goal of our approach. The first notion is the search of correlations between the metrics and performance.

A similar problem is studied in [32] and [20], where the authors generate well-known topologies and inspect the connectivity and robustness of the networks. In their case, a network is said to be robust if its performance is not sensitive to the changes in topology. In [32], the *algebraic connectivity* (not discussed in detail in this report) metric is studied to search robustness and metric relationships, however, they show that the algebraic connectivity is not trivially correlated to the robustness of the network.

The authors in [20] investigate the impact of betweenness centrality, algebraic connectivity and average degree to robustness, and they also draw the conclusion that there is no unique graph metric to satisfy both connectivity and robustness objectives while keeping a reasonable complexity, since each metric captures some attributes of the graph.

Partly based on the advances of these two publications and our initial assumption of the topologies and their metrics (Table 4.2), we expect that we cannot find a correlation between one metric and the performance, hence, we suspect that only the ensemble of more metrics is suited to find relationship.

In order to find quantitative connections, we use regression analysis to show how the various metrics impact the performance.

4.3.1 Choosing the Sample

By using regression analysis on a sample, it is important to regard the sample to be unbiased. In our case, a bias in a sample of graph topologies means a variation in the size of the models. Obviously, one topology in the sample with larger amount of nodes can bias the connection between the models and the performance, therefore, our framework must support the generation of *uniform* models[4] with respect to the amount of nodes and edges, even in the case of different topologies.

[4]From now on, under the concept of uniform models we mean models with approximately equal number of nodes and edges, despite the diversity of their internal structures.

Chapter 5

Contributions

In the following sections, we present the main contributions related to the MONDO-MAP framework and the Homogeneous Graphs Benchmark.

5.1 Overall Architecture

Figure 5.1 depicts the frameworks and the main components belonging to them, additionally, it also denotes which components are reused from Train Benchmark. All of the frameworks and their components were elaborated in Java programming language.

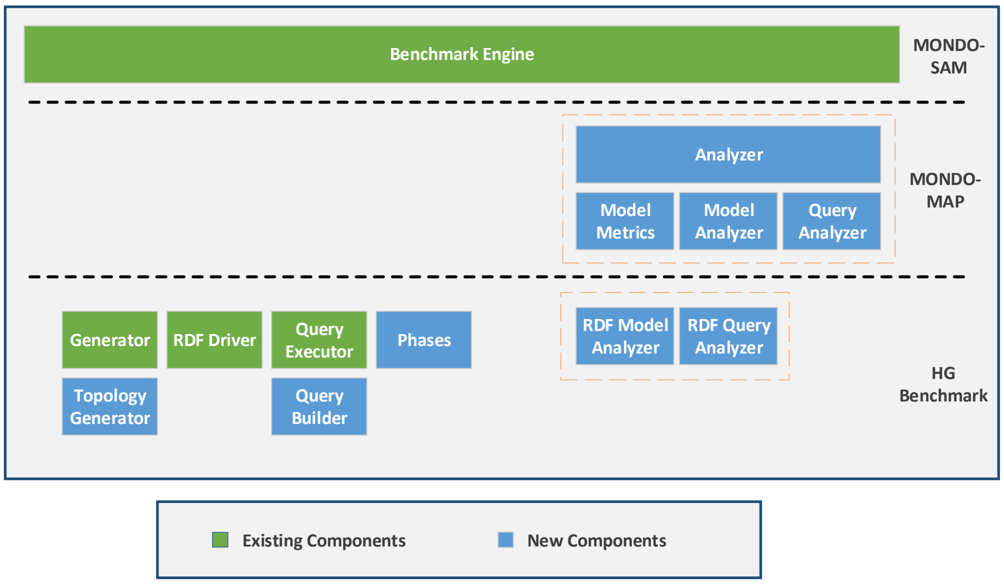


Figure 5.1. The architecture of our approach.

In the following, we introduce the components.

Benchmark Engine The Benchmark Engine in MONDO-SAM is responsible for evaluating an arbitrary sequence of phases consecutively. A phase is considered as the atomic execution unit in a benchmark. The engine also measures the evaluation times of phases, hence, it is the component that measures the performance of query evaluations in our work.

Analyzer Components The Model and Query Analyzer units belong to the MONDO-MAP framework and define an interface for the metrics calculation. The Model Analyzer investigates the model related metrics, and the Query Analyzer relates to the query definitions. As it can be observed, the concrete metric calculations—RDF Model Analyzer and RDF Query Analyzer—appear in the HG Benchmark.

Metrics The definitions of Model Metrics can be found in MONDO-MAP. The model metrics symbolize graph metrics with applying the commonly used naming conventions from graph theory. Note that the HG Benchmark does not contain further RDF-based metrics implying that we use the graph-based naming conventions in our work.

Generators The generator components belong to the HG Benchmark. The abstract Generator unit is utilized from Train Benchmark. Last, the Topology Generators construct different homogeneous graphs fitting to well-known topologies and transform them to RDF format.

RDF Driver The RDF Driver manages the connections between the measured RDF databases and the benchmark framework, furthermore, it also accomplishes the loading of the models.

Query Executor and Query Builder The query evaluations are initiated by the Query Executor component. The Query Builder is responsible for creating and altering the query definitions in runtime.

5.2 Uniform Model Generation

It is an essential requirement of the HG Benchmark to guarantee uniform model generation among the topologies indicating the same size of the generated models. We propose a model generation technique to generate topologies with the same amount of nodes and edges.

5.2.1 Number of Nodes

The random graph and the Watts-Strogatz model are constructed by initializing $|V| = N$ number of vertices, and then the algorithms determine which one of them become adjacent.

In the scale-free model generator, the nodes are created incrementally until $|V| = N$, and a precise number of nodes can be obtained regarding these topologies.

The only problem about generating topologies with a certain number of nodes is the recursive algorithm of the hierarchical graph, which algorithm has to be terminated.

Termination of the Hierarchical Network Generation Algorithm

Since the generation of hierarchical network is recursive instead of being incremental—as in the case of the three other topologies—it is necessary to determine a termination from the recursive algorithm. The termination point is evident, as soon as the number of created nodes reaches the limit, the algorithm has to be stopped. However, it cannot be predicted in which phase the algorithm stops exactly. As a consequence, the possible problems have to be managed.

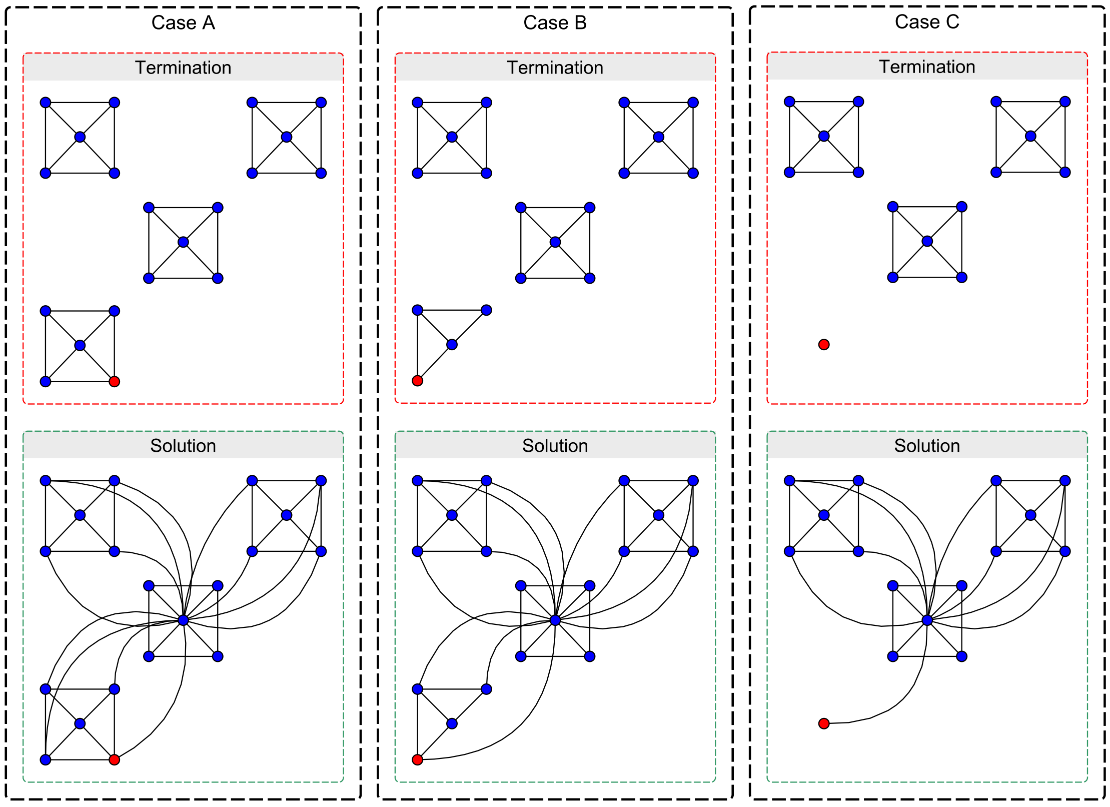


Figure 5.2. Possible termination problems in the hierarchical graph generation.

The possible problematic cases are demonstrated in Figure 5.2. In Case A, B, and C, the expected numbers of nodes are 20, 19, and 16, respectively. As it can be observed in these cases, this limit is always reached before the fourth cloning occurs, since 5 clusters should be created with 25 vertices at the end of this step in the recursion.

In the solution in Case A, the generator stops the cloning procedure and connects the diagonals to the center. Regarding B, the termination happens during the generation of a cluster. As a solution, the last cluster becomes partial, and similarly, every diagonal is attached to the center. Case C represents that scenario when the last cluster only consists of one node. To prevent isolation, the last vertex is considered as a diagonal, and be connected to the center.

5.2.2 Number of Edges

In terms of the random graph, scale-free, and WS model, their generation algorithms can be adjusted arbitrarily, meaning that an optional number of nodes or edges can be achieved. As a matter of fact, reaching a certain amount of nodes or edges in these networks are handled separately.

Unfortunately, the creation of nodes and edges in the hierarchical graph occur together. Since the amount of edges depends on the number of nodes and iterations in the recursive algorithm, it cannot be configured arbitrarily. A solution is that we adjust the other topologies to have the same number of edges as the hierarchical model. This solution requires to calculate the exact number of edges in a hierarchical network with respect to the iteration.

Estimating the Number of Edges in Hierarchical Graphs

The literature relating to hierarchical graphs does not mention the exact number of edges or its correlation to the amount of nodes, hence, we propose a solution to estimate $|E|$ in the recursive algorithm for every iteration.

At first, let us define the necessary variables hereunder:

- i: represents the current iteration in the original hierarchical algorithm.
- c: indicates the number of clones in every iteration.
- n: the cluster size is denoted by n, which is a K_n complete graph.
- F_i: indicates the constructed graph after the i. iteration.
- $|E_{F_i}|$: the number of edges of F_i.

The algorithm works as follows. In the 0. iteration, the hierarchical graph consists of one K_n cluster. Formally, $F_0 = K_n$, and $|E_{F_0}| = |E_{K_n}| = \frac{n\cdot(n-1)}{2}$. In the 1. iteration, the algorithm clones F_0 c times, and connects the peripheral nodes from each F_0 to the center node. It entails that

$$|E_{F_1}| = (c+1)\cdot|E_{K_n}| + c\cdot(n-1) \tag{5.1}$$

since $c+1$ number of K_n can be found in F_1, and $c\cdot(n-1)$ edges can be drawn from the c number of replicas to the center.

Note that in the first iteration K_n can be substituted with F_0, and the algorithm in the first part of the i. iteration creates c replicas of the result of the $i-1$. iteration, namely, F_{i-1}. In the second part of the i iteration, the algorithm connects the clusters from the cloned replicas to the center node. These connections are made between the peripheral nodes in each replica and the center, which indicates that in the i. iteration the algorithm connects $n-1$ peripheral nodes from c number of replicas of F_{i-1}. Due to F_{i-1} includes c^{i-1} number of clusters, we obtain

$$|E_{F_i}| = (c+1)\cdot|E_{F_{i-1}}| + c\cdot c^{i-1}\cdot(n-1) = (c+1)\cdot|E_{F_{i-1}}| + c^i\cdot(n-1) \tag{5.2}$$

Equation 5.2 is equal to the number of edges of a completely finished hierarchical network, however, the generation algorithm in the HG Benchmark is possibly terminated to reach a certain number of nodes which leads to the fact that $|E_{F_i}|$ must be scaled down by the proportion of the maximum ($|V_{F_i}|$) and the required number ($|V_H|$) of vertices. If the hierarchical graph we intend to generate is denoted by H, then

$$|E_H| = |E_{F_i}|\cdot\frac{|V_H|}{|V_{F_i}|} \tag{5.3}$$

where $\frac{|V_H|}{|V_{F_i}|} \leq 1$.

By using Equation 5.3, we can calculate the number of edges of a hierarchical graph and configure the other topologies to reach the same quantity.

Configuring the Random Graph Model

From the two most well-known algorithms, Gilbert's $G(n,p)$ model is adapted to the framework, which implies that the exact value of p has to be determined from the number of edges in the hierarchical graph, $|E_H|$. Based on [25], the p probability can be calculated from the number of nodes and edges as follows:

$$p = \frac{|E_H|}{\binom{|V|}{2}} \tag{5.4}$$

where $|V|$ denotes the number of nodes.

Configuring the Watts-Strogatz Model

Regarding the Watts-Strogatz model, in the beginning of the generation algorithm, K number of consecutive nodes are connected to each other. During the algorithm, by rewiring the edges the amount of $|E|$ is not changed. As a conclusion, in order to achieve a uniform size similarly to the hierarchical graph, K has to be adjusted as $K = \frac{|E_H|}{|V|}$.

Generally, the K value in the algorithm is a constant integer. In order to configure the WS model properly, we extend the algorithm by defining an inclusive lower bound (K_1) and upper bound (K_2) for K, as $K \in [K_1, K_2]$. We also assign a p probability to K that determines the likelihood that K is equal to K_2 and $1 - p$ to K_1. Derived from the equation $K = \frac{|E_H|}{|V|}$, it results in $K_1 = \left\lfloor \frac{|E_H|}{|V|} \right\rfloor$ and $K_2 = \left\lceil \frac{|E_H|}{|V|} \right\rceil$, additionally, the p probability equals to the fractional part, as $p = \left\{ \frac{|E_H|}{|V|} \right\}$. Hence, by turning K to a random variable, we can generate WS models with the same number of edges as $|E_H|$.

Configuring the Scale-Free Model

The scale-free topology is generated incrementally, since every step a new node is inserted to the graph with m new connections. To obtain $|E_H|$ edges, the m variable has to be configured. This leads to $m = \frac{|E_H|}{|V|}$.

In the original generation algorithm, every new vertex connects to a constant number of disjunct nodes, which indicates that m is a constant integer. Similarly to the notion in the Watts-Strogatz model generation (5.2.2), this constant value is converted to a random variable based on a particular probability, derived from $|E_H|$.

5.2.3 Possible Model Configuration

In the artificially generated models the size, topology and density is optionally configurable.

The size of the model—the number of nodes—is calculated by a formula as $|V| = s \cdot 2^i$, where s is the step size constant and i is a positive integer. It implies that an arbitrary model size can be obtained among the topologies.

Besides the size, the density of the graphs is also configurable, so the number of edges in the topologies. Since the hierarchical network is considered as the reference model due to the uniform model generation, the density parameter calibrates the generation algorithm of the hierarchical graph, namely, the size of the K_n clusters with altering n.

5.3 Performance Analysis

5.3.1 Workflow

A workflow in the HG Benchmark is divided into phases that are considered as the atomic execution units. An arbitrary sequence can be created among the phases, which—during the benchmark—are executed consecutively by the workflow engine in MONDO-SAM.

The workflow of the HG Benchmark is represented in Figure 5.3. After loading the model, the framework calculates the model related metrics. Due to the fact that in the current phase of our research we do not consider model transformations, therefore, a particular

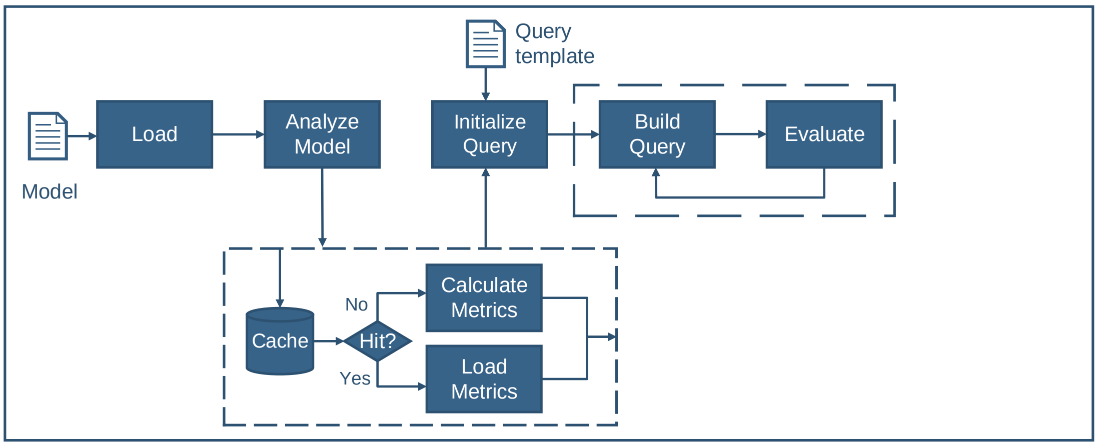


Figure 5.3. The workflow of the HG Benchmark.

model's metrics must be calculated only once in the beginning of the workflow. More importantly, different runs of the benchmark can utilize the previously calculated metrics that belong to the same model. As it can be observed in the model analysis phase, the solution is achieved by using a cache for the calculated metrics and reusing its content if possible.

The features in the Initialize and Build Query phases are strongly correlated. These two phases entail the creation of dynamic queries. The first one provides a default query definition that can be parameterized, and the second phase assembles a complete query for the evaluation, as it injects parameters or alters the entire syntax. The latter operation implies that entirely different queries can be executed in a sequence.

Last, the evaluation phase is responsible for executing the query. The *build* and *evaluate* phases can be repeated implying that more then one query can be evaluated in a sequence, even with different definitions.

5.3.2 Metrics Calculation

As we already emphasized, the MONDO-MAP framework proposes two types of metrics. The first is the set of model descriptive metrics and the second relates to the query definitions. In the following, we introduce them and their calculations in the HG Benchmark.

Model Metrics

The model-based metrics are connected to graph metrics which appear in their naming conventions as well. Since we are concentrating on RDF tools in our work, we also define the corresponding interpretations. The metrics are listed hereunder.

1. **Nodes:** the number of nodes in the graph. In RDF, this equals to the number of unique subject and object values.

2. **Edges:** the number of edges in the graph. Regarding RDF, this is equal to the number of predicates[1] in the data.

3. **Maximum Degree:** the maximum number of predicates per subjects.

4. **Average Degree:** it is determined by calculating the degree of every existing node.

5. **Average Degree Distribution:** denotes the probability that a randomly selected node's degree is equal to the average degree.

6. **Higher Degree Distribution:** the cumulative distribution of those nodes that have higher degrees than the average.

7. **Average Clustering Coefficient:** this metric implies the calculation of clustering coefficient per every node.

8. **Average Shortest Path Length:** the calculation of this metric is most expensive, hence, the framework searches a limited number (100) of shortest paths between randomly selected vertices and calculates their average length.

9. **Maximum Betweenness Centrality:** the value of this metric is determined by the shortest paths. We count the occurrences of every intermediate node in the paths—by determining the betweenness centrality of the vertices—and normalize the values to the $[0, 1]$ interval by dividing them with the number of visited nodes. Since the value of betweenness centrality is assigned to each node separately, we use the maximum of them.

5.3.3 Queries

We investigate the performance of two queries in our HG Benchmark. The first one relates to the concept of *shortest path*, and the second connects to the notion to investigating the *spread of information* in the graphs.

Shortest Path Query

The SPARQL definition of the query is shown below[2]:

```
PREFIX  base: <http://www.semanticweb.org/ontologies/2015/hgbenchmark#>
PREFIX  rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT  (count(*) AS ?count)
WHERE
  { ?sourceStation rdf:type base:Station .
    ?sourceStation (base:neighbor)* ?targetStation
    FILTER ( ?sourceStation = base:_ID1 )
```

[1]With the consideration of rdf:type predicates, the number of edges metric represents the number of triples.

[2]The domain of the homogeneous graphs is based on Stations.

```
    FILTER ( ?targetStation = base:_ID2 )
  }
```

The query uses the * operator from SPARQL Property Paths [14] in the (base:neighbor)* predicate which means an arbitrary length of path with the neighbor predicates. The query is a parameterized query in which we inject two random identifiers instead the ID parameters. Finally, the query searches a path between those two nodes.

Information Spread Query

The query investigates the spread of information in the graph—by starting from a randomly chosen node—and it traverses the graph via the neighbor predicates to a three-hop distance. The concept of information spreading means how fast the information can be forwarded among the nodes in the graph, or in other words, how many vertices can be reached from a certain node.
The SPARQL definition of the query is shown below. As it can observed, in every new navigation we filter the previously found nodes to prevent the traversal of the same nodes again.

```
PREFIX  base: <http://www.semanticweb.org/ontologies/2015/hgbenchmark#>
PREFIX  rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT  ( COUNT(*) AS ?count)
WHERE
  { ?station1 rdf:type base:Station
    FILTER ( ?station1 = base:_ID ) .
    ?station1 base:neighbor ?station2 .
    ?station2 base:neighbor ?station3
    FILTER (?station1 != ?station3) .
    ?station3 base:neighbor ?station4
    FILTER (?station1 != ?station4 && ?station2 != ?station4) .
  }
```

5.3.4 Tools

The tools used in this report are listed in Table 5.1. The last two columns illustrate whether the execution of our defined queries is feasible in the particular tool or not. As the table suggests, the first query cannot be evaluated in 4store, since it does not support the usage of property paths.

Name	Implementation Language	Version	Query 1	Query 2
Blazegraph [3]	Java	1.5.2	●	●
4store [1]	C	1.1.5		●
Sesame [13]	Java	2.7.9	●	●

Table 5.1. The implemented tools in HG Benchmark

Chapter 6

Evaluation

6.1 Benchmarking Environment

The benchmarking environment involves a single machine that contains a quad-core Intel Xeon Processor L5420 (2.5 GHz) and 16 GBs of RAM. In order to alleviate the effect of transient on the measurements and minimize the noise in the results, a 64-bit Ubuntu 14.04.2 LTS was installed. Additionally, Oracle JDK version 1.8.0 was used as the Java environment.

6.2 Benchmark Configuration

Topologies We generate five topologies for the measurements such as the scale-free model, hierarchical network and Watts-Strogatz models with different probability p values as 0.1, 0.01 and 0.001. Due to these configurations of p, we reach a deviation between Watts-Strogatz models and random graphs.
The topologies are generated via our uniform model generation approach (Section 5.2). Every topology is generated five times with different densities that are adjusted to five different values based on the size of K_n complete graphs found in the hierarchical network. These K_n complete graphs are configured to K_3, K_4, K_5 K_6 and K_7.

Queries The queries are parameterized with random values 20 times and executed on every graph. Each measurements is performed 4 times.

Model Size The configuration of the model sizes are shown in Table 6.1 representing number of nodes and edges. The latter is given by an interval since we generate the graph with different densities.

Nodes	Min Edges	Max Edges
5 000	23 531	42 633
10 000	49 140	89772
20 000	100656	185420
40 000	209439	381435
80 000	421110	800314

Table 6.1. The number of nodes and edges in generated graphs.

6.3 Samples

In order to analyze the relationships between model metrics and the performance of query evaluations, we create different samples of the measurements and investigate them respectively. The triple of a specific tool, query and model size (number of nodes) defines a sample. A sample contains the measurement results for a particular query evaluated by a tool on a model of a certain size. As a result, the number of different samples is equal to the product of unique tools, queries and model sizes.

6.3.1 Sample Size

A sample includes the measurements executed on 5 different topologies, each of them appears 5 times in the sample with different densities that are configured between the various topologies equally. One sample contains 20 evaluations of a parameterized query. Every measurement is repeated 4 times in a sequence, however, we discard the first evaluation times and use the median of the remaining measurements.
The dimensions in a sample and their occurrences are illustrated in Table 6.2. Since the triple of a tool, query and model size defines a different sample, these values represent one value in a sample. As it can be observed, the sample size is equal to 500.

	Tool	Query	Model Size	Query Evaluation	Topology	Density	$\prod$
Occurrence	1	1	1	20	5	5	500

Table 6.2. The dimensions and their occurrence in a sample.

6.4 Model Analysis

6.4.1 Density

One essential expectation was in our work to generate different graphs with the same number of nodes and edges. Figure 6.1 depicts the density of the topologies. The x-axis represents the sizes of the models, the y-axis shows the values belonging to density. Every column contains the information of one specific topology, furthermore, the different densities are separated by colours in the legend.

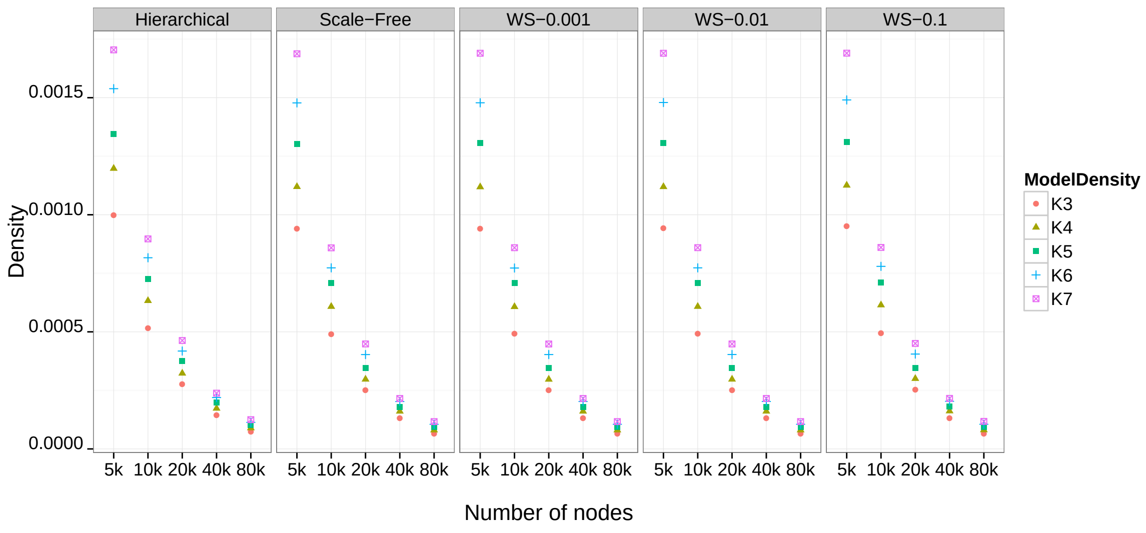


Figure 6.1. The density of the graph topologies.

As it can be observed, the topologies with different networks have approximately the same density with the respect to the number of nodes, which implies the number of edges is almost equal. A small deviation is shown between the hierarchical graph and the other topologies. This is explained by the fact that the recursive generation algorithm of the hierarchical network must be terminated before its end in order to obtain an arbitrary number of nodes. Hence, we can only estimate the amount of edges in the graph with a dispersion. The standard deviation of the densities is $1.11 \cdot 10^{-5}$, and divided by the maximum number of density we obtain 0.028, which means a 2.8% difference between the topologies with respect to their density.

6.4.2 Clustering Coefficient

Our goal is to achieve a deviation in metric values per topologies such as the clustering coefficient metric. Figure 6.2 depicts the average clustering coefficients in the topologies. The x-axis represents the number of nodes in the graphs, the y-axis denotes the values of the metric, furthermore, every value is separated by the topology in legend.
The plot shows how the values spread in the $[0, 1]$ interval according to our early expectation (Section 4.2.2). One topology has more corresponding metric value in the same size—e.g. hierarchical—due to the fact that we generated 5 instances of every topology with different densities, which implies a different clustering coefficient metric per instance as well.

6.4.3 Shortest Path Length

The average lengths of shortest paths in the topologies are demonstrated in Figure 6.4. It can be observed that as we decrease the p probability in the Watts-Strogatz models (from 0.1 to 0.001), the lengths of the shortest paths increase. This negative correlation

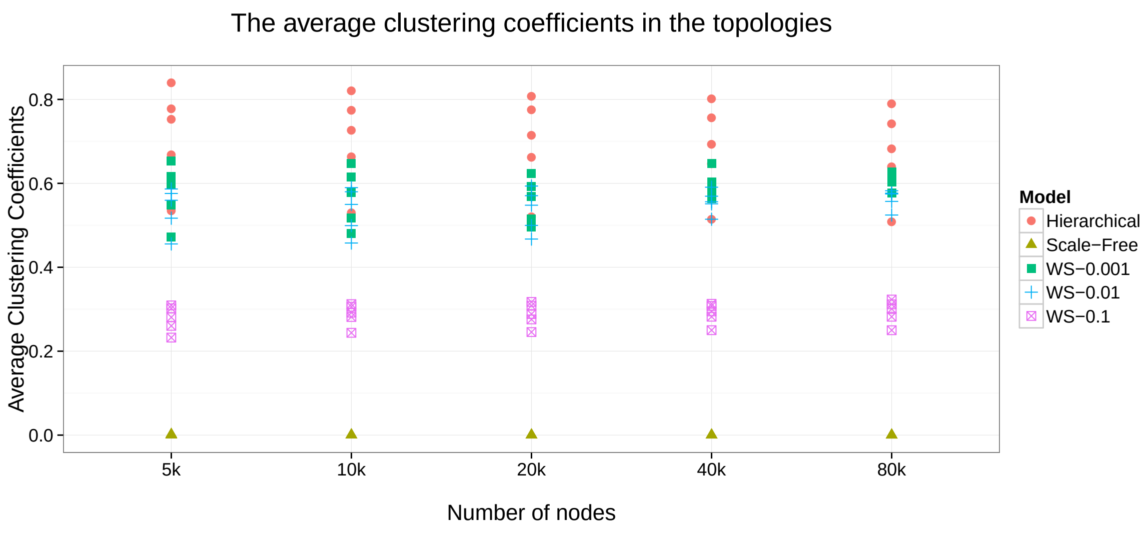


Figure 6.2. The average clustering coefficient in the graph topologies.

shows the same results that we expected and can be explained by the operation of the generation algorithm belonging to the Watts-Strogatz model. As we increase the p value, the algorithm rewires every edge with p probability—i.e. it deletes an edge and connects it to a new random node—and this entails a bigger interconnectivity in the graph showing a small-world property. On the contrary, a low p value (e.g. 0.001) modifies a subtle set of the edges, therefore, the graph still shows similar characteristics than a lattice graph with high average shortest path length.

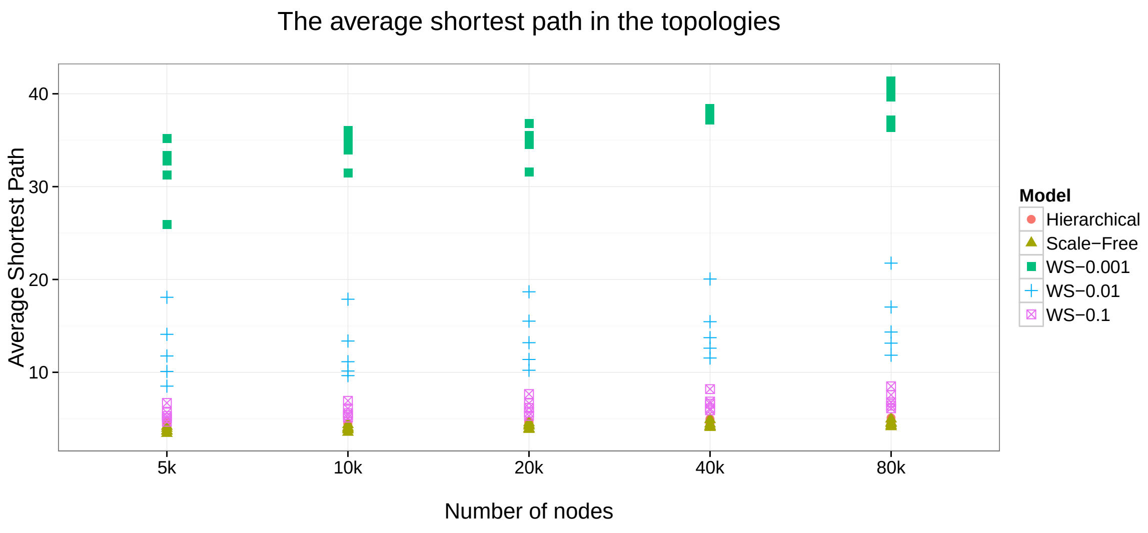


Figure 6.3. The average shortest path in the graph topologies.

6.4.4 Betweenness Centrality

Figure 6.4 illustrates the betweenness centrality metric per topologies. Regarding this metric, we assumed a more significant deviation among the topologies, as we expected that the *hubs*—the nodes with higher degrees—in scale-free models appear more times in the shortest paths implying a higher betweenness centrality. Instead, the WS-0.001 model shows higher values in this metric than the scale-free model except in the case of the largest graph.
A gap is observed between the hierarchical network and the other topologies due to the fact the center node in the hierarchical graph dominates the calculation of the betweenness centrality metric.

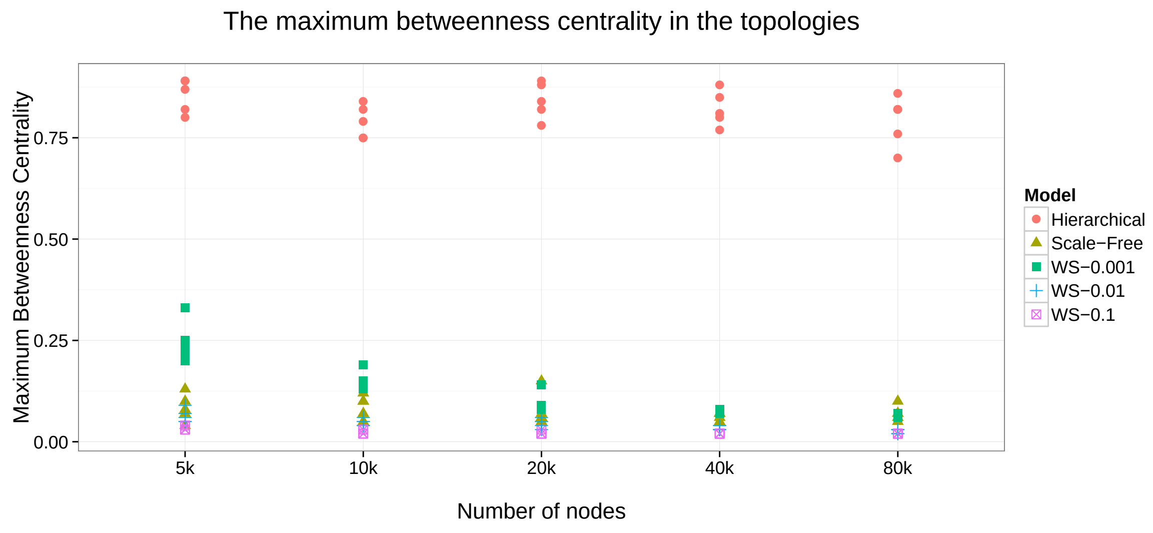


Figure 6.4. The maximum betweenness centrality in the graph topologies.

6.5 Performance Analysis

6.5.1 Hypothesis

Our hypotheses about the results of query evaluations are the following.

Reachability Query Regarding the first query—Reachability—we assume that the larger clustering coefficients and higher degrees may cause a performance loss of the tools. If the graph has a higher clustering coefficient then there is a possibility that the execution of the query revisits a certain node more times via its neighbors. This implies that the evaluation contains more—unnecessary—navigations.
A node with higher degree also can affect the performance, since if the graph traversal visits this certain node with higher degree than there is a possibility that it also investi-

gates its neighbors. Nodes with high degree typically appear in the scale-free models—the hubs—and the hierarchical networks—the center nodes.

The assumption comes naturally that the average shortest path length also can dominate the execution time. Among the different Watts-Strogatz models, the ones with lower p values include a higher average shortest path metric, suggesting a growing in the evaluation time. However, considering the other topologies as well, it cannot be determined forward which specific characteristic dominates better.

Navigations Query The Navigations query searches nodes in three-hop distances starting from a random vertex. We believe that the nodes with higher degrees impact the performance mostly, namely, the center nodes in the hierarchical graph and the hubs in the scale-free models. We fundamental question is that whether our defined metrics are suitable to characterize the performance appropriately or not.

6.5.2 Highlights of the Analysis

For the performance analysis we created and measured 50 different samples, each of them contains 500 observations—i.e. measurement results. In the following sections we do not intend to show the results in details of every sample, hence, we only concentrate on the samples that contain interesting or unexpected results.

Blazegraph is highly sensitive to the average shortest path of the graph

The measurement results of Blazegraph is illustrated in Figure 6.5. The box plot [4] shows the results in following dimensions. On the x-axis denotes the graph topologies and the y-axis represents the evaluation time in milliseconds on a logarithmic scale. Every column includes different results belonging to a specific model size.

The most important observation is the deviation in evaluation time among the topologies. Besides the execution times increases with respect to the number of nodes, the performance also varies among the networks being in the same size. A significant difference is shown between the Watts-Strogatz models (WS-0.001, WS-0.01) and the other networks.

The created regression models are listed in Table 6.3. Every measurement result and metric were normalized for the calculations, and the table includes the best fitted regression models on the measurements. In every case, the average shortest path metric seemed to be the best predictor. Regarding different model sizes, the value of adjusted R^2—i.e. coefficient of determination—varies between 0.68 and 0.9 that show well-fitted regression models.

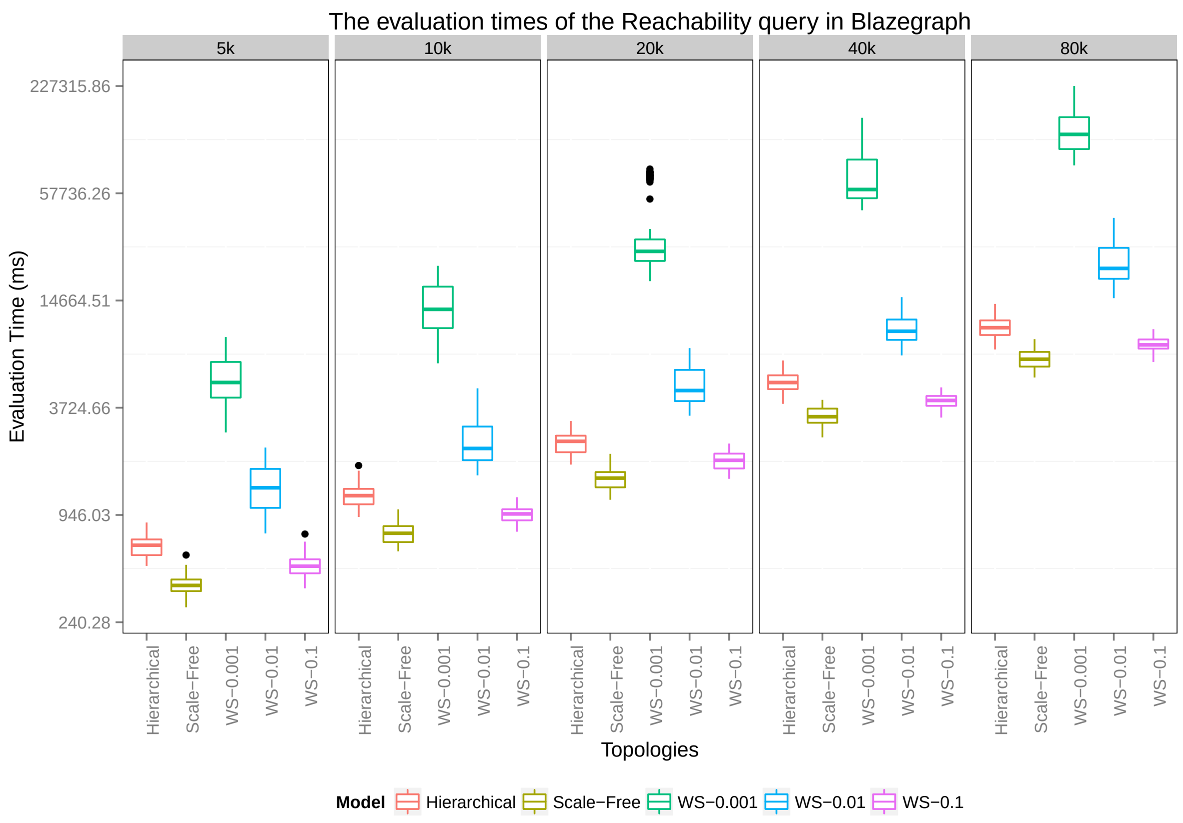


Figure 6.5. The measurement results of the Reachability query in Blazegraph.

Model Size	Metric	Adjusted R^2	Regression Coefficient	P-value
5 000	Avg Shortest Path	0.9	0.949	$2.2 \cdot 10^{-16}$
10 000	Avg Shortest Path	0.8586	0.9268	$2 \cdot 10^{-16}$
20 000	Avg Shortest Path	0.6807	0.8254	$2.377 \cdot 10^{-16}$
40 000	Avg Shortest Path	0.7647	0.8745	$3.3 \cdot 10^{-16}$
80 000	Avg Shortest Path	0.7642	0.8745	$3.3 \cdot 10^{-16}$

Table 6.3. The best fitted regression models of Blazegraph.

Sesame is mostly unaffected by the topology of the model

The evaluation results of Sesame are illustrated in Figure 6.6. As it can be observed, the optimization in Sesame is not sensitive to the different topologies, as the various characteristics of the networks still cause nearly the same evaluation time. A higher difference only occurs between the different model sizes. Interestingly, a small difference is also observable between the Watts-Strogatz models, as the p values increases—implying the decrease of the average shortest path metric—the evaluation times still grow.
Due to the approximately equal results, we cannot create an appropriate well-fitted regression model for Sesame.

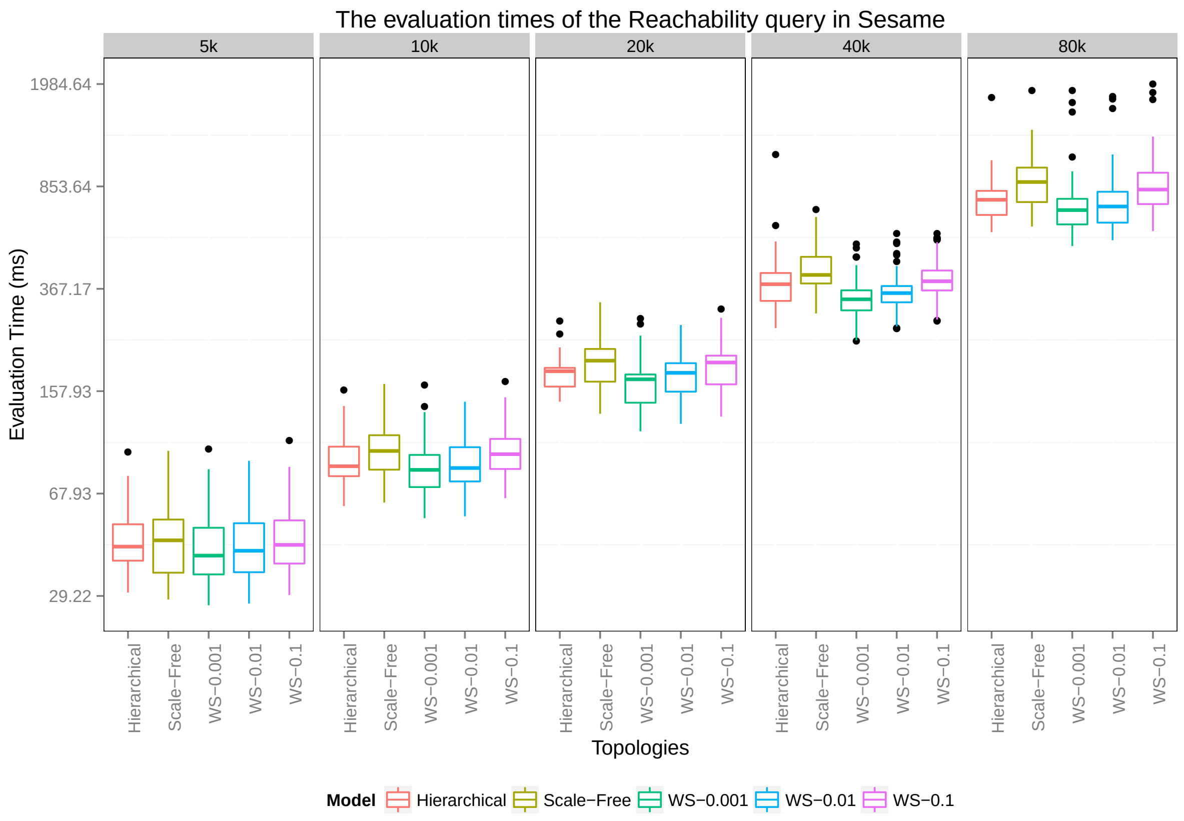


Figure 6.6. The measurement results of the Reachability query in Sesame.

The clustering coefficient does not influence the performance for the Reachability test

Interesting to show that our assumption about the clustering coefficient metric is proved to be false. The adjusted R^2 values of the regression models containing the clustering coefficient are considerably small. In the case of Sesame, the adjusted R^2 is approximately 0.06 per different model sizes, and regarding Blazegraph, this value is equal to 0.13. Both of them indicate that the clustering coefficient has a minimal impact to the performance, in the light of these workloads.

Negligible relationship between the hubs and the performance of the Reachability test

The results of the Reachability query shows that the nodes which have significantly larger degrees do not affect the performance of the evaluations. At first sight, we would assume that the occurrence of hubs implies that the execution of the query must visits a significantly larger amount of nodes. Since, if the evaluation reaches a hub during the graph traversal than it possibly must visit its neighbors as well, causing a performance loss. However, in the case of Blazegraph, the Watts-Strogatz models with higher average shortest path lengths seem to be dominating, despite the fact that the same Reachability test in a hierarchical or scale-free network may visit more vertices than in the case of the WS models.

Navigations on graphs are dominated by hubs

The measurement results of the Navigation query is illustrated in Figure 6.7. As we assumed, the hubs—i.e. the significantly higher degree nodes—affect the performance of this query evaluation. A more interesting question is that whether our metrics can characterize this relationship appropriately or not.

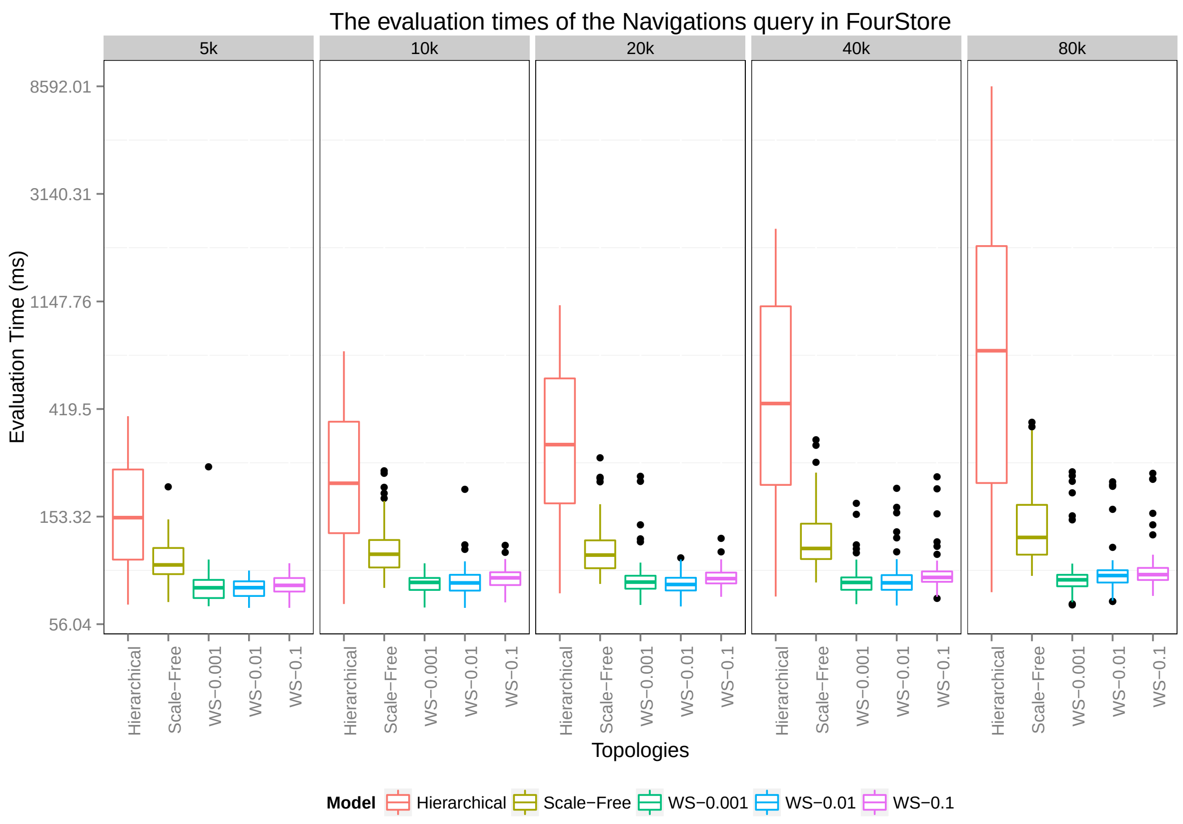


Figure 6.7. The measurement results of the Navigations query in FourStore.

Our created regression models are shown in Table 6.4. These are the ones that provided the best fit to the data including the betweenness centrality and maximum degree metric. The main conclusion is that our metrics partly can characterize the relationship, since the adjusted R^2 values vary between the 0.4396 and 0.5228 interval.

As far as the other tools are concerned, such as Sesame and Blazegraph, the same metrics can be used to obtain the best fitted regression, and the adjusted R^2 values vary between the 0.4018 and 0.5173 interval. These results also show that only these two metrics were

Model Size	Metrics	Adjusted R^2	P-value
5 000	Betweenness + Maximum Degree	0.5228	$2.2 \cdot 10^{-16}$
10 000	Betweenness + Maximum Degree	0.5204	$2.2 \cdot 10^{-16}$
20 000	Betweenness + Maximum Degree	0.5125	$2.2 \cdot 10^{-16}$
40 000	Betweenness + Maximum Degree	0.4815	$2.2 \cdot 10^{-16}$
80 000	Betweenness + Maximum Degree	0.4396	$2.2 \cdot 10^{-16}$

Table 6.4. The best fitted regression models of Fourstore.

not adequate to characterize the performance properly regarding these workloads and samples.

6.6 Conclusions

Model Generation We showed that our model generation approach was able to construct different topologies with the same density with respect to the same number of nodes. As it was illustrated, the clustering coefficient and average shortest path length metrics were deviated appropriately among the topologies. Unfortunately, the betweenness centrality metric seemed to be dominated by the hierarchical network, and it also showed a significantly lower value in the scale-free models, than we originally expected.

Performance The measurement results illustrated that the different internal structure of the graphs are able to cause a deviation in evaluation time, regarding the Reachability and Navigations query as well. In the first query, Blazegraph seems to be sensitive to the average shortest path length metric, on the contrary, the performance of Sesame is not affected by the topologies regarding these workloads.

Hierarchical Graph The analysis of the topologies show that the hierarchical graph has significantly different descriptive metrics than the other topologies, furthermore, our measurement results of the Navigation query also imply that the hierarchical network is overly dominating to the performance by comparing with the other networks. Our generation technique also showed that—by relying on the hierarchical graph—we cannot achieve an arbitrary density in the graph which is possible with the generation algorithms of the other topologies. We had to adapt the other networks to the limitations of the hierarchical graph. As a conclusion, the hierarchical graph is not capable to be used in our framework in the future due to its limitations.

Chapter 7

Summary

7.1 Scientific Contributions

We achieved the following scientific contributions:

- We proposed a concept to investigate the relationships between graph-based metrics and performance of query evaluations.
- We provided an approach of uniform graph generation among different topologies.
- We proposed a solution to estimate the number of edges in the hierarchical network topology.

7.2 Practical Accomplishments

We achieved the following practical accomplishments:

- We designed a framework for metric-based performance analysis.
- We elaborated a benchmark framework to assess the performance of query evaluations on homogeneous graph topologies.

7.3 Future Work

The following tasks are addressed as future works:

- Along the query and model metrics, we would like to investigate the effect of *query on model metrics* [31]. The query on model metrics could be used to characterize specific cardinalities of the model (e.g. the number of vertex types, the number of hubs, etc.)

- The evaluation in this report focused on *semantic databases*. In the future, we also plan to evaluate the performance of *graph databases*, e.g. Neo4j [36] and OrientDB [17].
- To achieve better prediction, we would like to experiment with non-linear regression models.

Acknowledgements

I would like to thank my supervisors Gábor Szárnyas and Dr. István Ráth for their generous advice. I wish to express my gratitude to Ágnes Salánki and Imre Kocsis for their help in statistics. Also, I would like to thank Dávid Cseh for supporting my work in the cloud.

List of Figures

List of Tables

Bibliography

[1] 4store. http://4store.org/.

[2] Berlin SPARQL Benchmark (BSBM) Specification V3.1. http://wifo5-03.informatik.uni-mannheim.de/bizer/berlinsparqlbenchmark/spec/.

[3] Blazegraph. https://www.blazegraph.com/product/.

[4] Box Plot. http://www.physics.csbsju.edu/stats/box2.html.

[5] Covariance. http://mathworld.wolfram.com/Covariance.html.

[6] DBpedia. http://wiki.dbpedia.org/.

[7] Digital Bibliography and Library Project. http://dblp.uni-trier.de/db/.

[8] MONDO. http://www.mondo-project.org/.

[9] Population and Sample Definition. http://www.statistics.com/glossary&term_id=812.

[10] Populations and Samples. http://stattrek.com/sampling/populations-and-samples.aspx?Tutorial=AP.

[11] Random Variable. http://www.stat.yale.edu/Courses/1997-98/101/ranvar.htm.

[12] Resource Description Framework. http://www.w3.org/RDF/.

[13] Sesame. http://rdf4j.org/.

[14] SPARQL Property Paths. http://www.w3.org/TR/sparql11-property-paths/.

[15] The TTC 2015 Train Benchmark Case for Incremental Model Validation. https://www.sharelatex.com/github/repos/FTSRG/trainbenchmark-ttc-paper/.

[16] Transaction Processing Performance Council. http://www.tpc.org/tpch/.

[17] OrientDB Graph-Document NoSQL DBMS. http://www.orientdb.org/, October 2015.

[18] A. L. Barabási, Z. N. Oltvai. Understanding the cell's functional organization nature genetics. *Network Biology*, 5:101–114, 2004.

[19] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Rev. Mod. Phys.*, 74:47–97, Jan 2002.

[20] Alireza Bigdeli, Ali Tizghadam, and Alberto Leon-Garcia. Comparison of network criticality, algebraic connectivity, and other graph metrics. In *Proceedings of the 1st Annual Workshop on Simplifying Complex Network for Practitioners*, SIMPLEX '09, pages 4:1–4:6, New York, NY, USA, 2009. ACM.

[21] Christian Bizer and Andreas Schultz. The berlin sparql benchmark. *International Journal On Semantic Web and Information Systems*, 2009.

[22] Miyuru Dayarathna and Toyotaro Suzumura. Graph database benchmarking on cloud environments with xgdbench. *Autom. Softw. Eng.*, 21(4):509–533, 2014.

[23] George C. Runger Douglas C. Montgomery. *Applied Statistics and Probability for Engineers.*

[24] Erdös, P. and Rényi, A. On random graphs, I. *Publicationes Mathematicae (Debrecen)*, 6:290–297, 1959.

[25] P. Erdős and A Rényi. On the evolution of random graphs. In *Publication of the Mathematical Institute of the Hungarian Academy of Sciences*, page 20, 1960.

[26] E. N. Gilbert. Random graphs. *Ann. Math. Statist.*, 30(4):1141–1144, 12 1959.

[27] Tassilo Horn, Filip Krikava, and Louis M. Rose, editors. *Proceedings of the 8th Transformation Tool Contest part of the Software Technologies: Applications and Foundations (STAF 2015) federation of conferences, L'Aquila, Italy, July 24, 2015*, CEUR Workshop Proceedings. CEUR-WS.org, 2014.

[28] L. A. Huberman, B. A. ; Adamic. Growth dynamics of the world-wide web. In *Nature*, volume 401, page 131, 1999.

[29] Benedek Izsó, Gábor Szárnyas, István Ráth, and Dániel Varró. MONDO-SAM: A Framework to Systematically Assess MDE Scalability. In *BigMDE 2014 2nd Workshop on Scalable Model Driven Engineering*, page 40. ACM, ACM, 2014.

[30] Benedek Izsó, Gábor Szárnyas, István Ráth, and Dániel Varró. Train benchmark technical report. `https://www.sharelatex.com/github/repos/FTSRG/trainbenchmark-docs/builds/latest/output.pdf`, 2014.

[31] Benedek Izsó, Zoltán Szatmári, Gábor Bergmann, Ákos Horváth, and István Ráth. Towards Precise Metrics for Predicting Graph Query Performance. In *2013 IEEE/ACM 28th International Conference on Automated Software Engineering (ASE)*, pages 412–431, Silicon Valley, CA, USA, 11/2013 2013. IEEE, IEEE. Acceptance Rate: 23%.

[32] A. Jamakovic and S. Uhlig. On the relationship between the algebraic connectivity and graph's robustness to node and link failures. In *Next Generation Internet Networks, 3rd EuroNGI Conference on*, pages 96–102, May 2007.

[33] Steffen Mazanek, Arend Rensink, and Pieter Van Gorp. Transformation Tool Contest 2010. *Malaga, Spain*, 41, 2010.

[34] Robert Campbell McColl, David Ediger, Jason Poovey, Dan Campbell, and David A. Bader. A performance evaluation of open source graph databases. In *Proceedings of the First Workshop on Parallel Programming for Analytics Applications*, PPAA '14, pages 11–18, New York, NY, USA, 2014. ACM.

[35] Mohamed Morsey, Jens Lehmann, Sören Auer, and Axel-Cyrille Ngonga Ngomo. Dbpedia sparql benchmark: Performance assessment with real queries on real data. In *Proceedings of the 10th International Conference on The Semantic Web - Volume Part I*, ISWC'11, pages 454–469, Berlin, Heidelberg, 2011. Springer-Verlag.

[36] Neo Technology. Neo4j. `http://neo4j.org/`, October 2015.

[37] M. E. J. Newman, S. H. Strogatz, and D. J. Watts. Random graphs with arbitrary degree distributions and their applications. *Phys. Rev. E*, 64:026118, Jul 2001.

[38] A. L. Barabási R. Albert, H. Jeong. The diameter of the world wide web. In *Nature*, volume 401, pages 130–131, 1999.

[39] Erzsébet Ravasz and Albert-László Barabási. Hierarchical organization in complex networks. *Phys. Rev. E*, 67:026112, Feb 2003.

[40] Marko A. Rodriguez. Titan Provides Real-Time Big Graph Data. `http://thinkaurelius.com/2012/08/06/titan-provides-real-time-big-graph-data/`, 2012.

[41] Louis M. Rose, Christian Krause, and Tassilo Horn, editors. *Proceedings of the 7th Transformation Tool Contest part of the Software Technologies: Applications and Foundations (STAF 2014) federation of conferences, York, United Kingdom, July 25, 2014*, volume 1305 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2014.

[42] Michael Schmidt, Thomas Hornung, Michael Meier, Christoph Pinkel, and Georg Lausen. Sp2bench: A sparql performance benchmark. In Roberto de Virgilio, Fausto Giunchiglia, and Letizia Tanca, editors, *Semantic Web Information Management*, pages 371–393. Springer Berlin Heidelberg, 2010.

[43] Sparsity Technologies. DEX scalability with high-performance. `http://sparsity-technologies.com/blog/dex-scalability-with-high-performance/`, 2011.

[44] Pieter Van Gorp, Steffen Mazanek, and Louis Rose. Fifth Transformation Tool Contest. *arXiv preprint arXiv:1111.4407*, 2011.

[45] Pieter Van Gorp, Louis M. Rose, and Christian Krause. Sixth Transformation Tool Contest. *arXiv preprint arXiv:1311.7536v1*, 2013.

[46] Wagon, Stan and Weisstein, Eric W. Lattice graph. `http://mathworld.wolfram.com/LatticeGraph.html`.

[47] Duncan J. Watts and Steven H. Strogatz. Collective dynamics of small-world networks. *Nature*, 393:440–442, June 1998.