
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 modell központú rendszerek tervezése során kulcsfontosságú a modellen kiértékelt lekérdezések optimális teljesítménye, ennek elérése érdekében pedig a megfelelő végrehajtó eszközök megválasztása. Az elmúlt években különböző NoSQL adatbázis-kezelő rendszerek váltak népszerűvé, amelyek célja a gyors lekérdezés kiértékelés és skálázhatóság biztosítása.

A lekérdezés kiértékelésének teljesítményét nagy mértékben befolyásolja a modell topológiája és a lekérdezés komplexitása. Célunk az, hogy bizonyos, a modellt és a lekérdezést leíró metrika felhasználásával, kapcsolatot találjunk a metrikák és a teljesítmény között, úgy, hogy az adott metrikákat ismerve, a teljesítmény megjósolható legyen.

A metrikák alapján tervezési szintű döntéseket hozhatunk az optimális teljesítményre törekedve, továbbá, ez a tudás lehetőséget teremt a valósidejű lekérdezés optimalizálás területén is arra, hogy a modellt és a lekérdezést jellemző metrikák alapján döntsünk optimalizálást érintő kérdésekben.

Különböző NoSQL adatbázis-kezelő rendszereket vizsgálva, regressziós analízis segítségével olyan metrikák után kutatunk, amelyek képesek az adott rendszer futási teljesítményét jellemezni. A vizsgálandó adathalmazok előállításához, egy valós modellből kiindulva, különböző eloszlású és topológiájú gráfokat generálunk.

További célunk az, hogy eldöntsük, vajon egy tetszőleges struktúrájú adatmodellre képesek vagyunk-e a regressziós analízisünk alapján a teljesítményre becslést adni, illetve a modellhez a megfelelő, készletünkben lévő adatbázis-kezelő rendszert társítani, a legjobb teljesítményre koncentrálva.

Végezetül, a valós idejű lekérdezések optimalizálásától motiválva, választ keresünk arra, hogy egy tetszőleges adatmodellre, költséghatékony metrikaszámítással a teljesítmény előre megbecsülhető-e.

# Abstract

Achieving the optimal performance of query evaluations plays an important goal in model-based systems. In the last few years, different NoSQL database systems were introduced to provide a better performance and solve the problem of scalability.

The performance of query evaluations depends on the topology of the model, and the complexity of the particular query. Our primary goal is that — by defining model and query-related quantitative metrics — to find a considerable connection between metrics and the performance, and thus, be able to predict the query evaluation time.

Based on the metrics and their effect to the performance, we are able to make decisions in design to achieve on the optimal performance. Furthermore, this knowledge can be utilized in the area of real-time query optimization engines as well.

We investigate various NoSQL database systems via regression analysis in order to find different model-related metrics that are suited to characterize their performance appropriately. Based on a real model, we generate graphs with various topologies and distributions to find metrics from different aspects.

Furthermore, we explore that whether an arbitrarily structured model's performance is predictable via our regression analysis, and also predict which database system from our scope can be associated to the model in order to achieve an optimal performance.

Finally, being motivated by the real-time optimization engines, we search answers whether by reducing the cost of metric calculations, an arbitrary model's performance is still predictable.

# Introduction

# Chapter 1

# Background

## 1.1 RDF

## 1.2 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 [24].

### 1.2.1 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 [13]. A population includes all of the elements from a set of data, however, a sample consists of one or more observations from the population [14]. The sample size is the number of observations in a sample and commonly denoted by $n$.

### 1.2.2 Probability Distributions

The probability distribution links each possible value of a random variable [15] with its probability of occurrence. The following paragraphs we define different 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, $x_1$, $x_2$, ..., $x_n$, has equal probability. Formally:

$$f(x_i) = \frac{1}{n} \tag{1.1}$$

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

$$f(x) = c \cdot x^{-\gamma} \tag{1.2}$$

where $c$ is a constant and $\gamma$ is called as the *exponent* or *scale factor*.

#### Poisson Distribution

The Poisson distribution is characterized by the following elementary probabilities:

$$P(X = k) = \frac{\lambda^k}{k!} e^{-\lambda} \tag{1.3}$$

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

### 1.2.3 Related Measurements

#### Mean

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

$$\mu = E(X) = \sum_x x f(x) \tag{1.4}$$

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 $\sigma^2$ or $V(X)$—is equal to the following formula:

$$\sigma^2 = V(X) = E(X - \mu)^2 = \sum_x (x - \mu)^2 f(x) \tag{1.5}$$

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.

### 1.2.4 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:

$$cov(X, Y) = E\big[(X - \mu_X)(Y - \mu_Y)\big] \tag{1.6}$$

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 [6]. A few examples of different scenarios are illustrated in Figure 1.1 ( [24]). In the terms of (a) and (b), a covariance is observable between $X$ and $Y$, and in the cases of (c) and (d), the covariance is equal to zero.

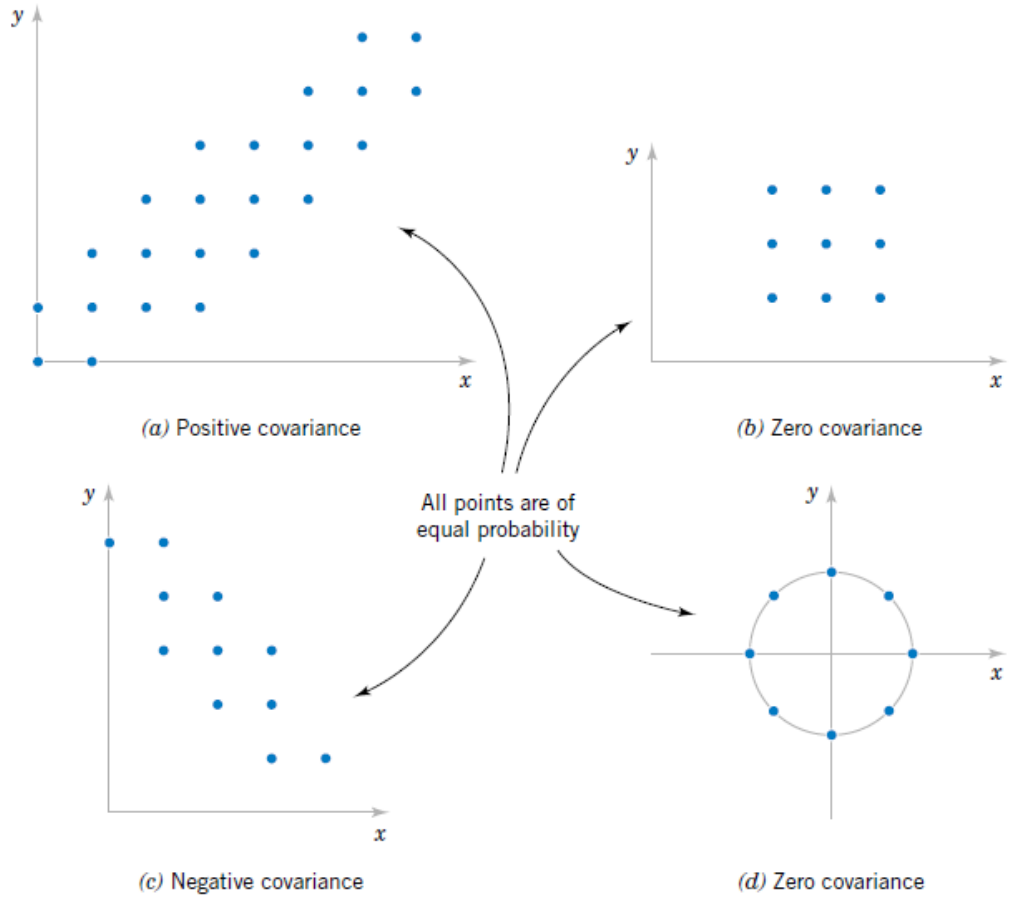


**Figure 1.1.** Different examples for covariance.

#### 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

$$\rho_{XY} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} \tag{1.7}$$

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 is interpreted as a correlation does not appear between the variables.

### 1.2.5 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 $\beta$. Formally, a regression model is defined as

$$Y = \beta_0 + \beta_1 x + \epsilon \tag{1.8}$$

where $Y$ is the dependent or response variable, $x$ is referred as the independent variable or predictor, and $\beta_0$, $\beta_1$ are the regression coefficients—the intercept and the slope. Finally, $\epsilon$ symbolizes the random error.

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

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k + \epsilon \tag{1.9}$$

## 1.3 Graph Theory

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

### 1.3.1 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.

### Clustering Coefficient

The $C_n$ clustering coefficient of an $n$ node 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 written, in undirected graphs the clustering coefficient of $n$ node is $C_n = \frac{2 \cdot e_n}{k_n \cdot (k_n - 1)}$, where $k_n$ is the number of neighbors of $n$, and $e_n$ is the number of connected pairs between all neighbors of $n$ ( [20]). The clustering coefficient is always quantified between 0 and 1.

An example is showed in Figure 1.2. 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 $e_A = 1$.

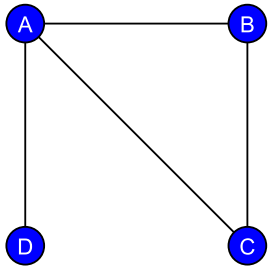


**Figure 1.2.** An example graph for illustrating the calculation of the clustering coefficient metric.

### Betweenness Centrality

The betweenness centrality of an $n$ node 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 three, 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.

### 1.3.2 Network Topologies

In the following sections, we introduce the graph topologies that connects 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 [25], and the second is the $G(N, p)$ model from Gilbert [27]. The former means that precisely $M$ number of 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 is considered to follow a small-world property if the graph shows 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 [37]. This implies that every node has $K$ degree, 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 by altering the $p$ value in the algorithm between 0 and 1, 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 $m$ number of 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:

$$p_i = \frac{d(i)}{\sum_j d(j)} \tag{1.10}$$

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 is depend 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 [35] is generated by a recursive algorithm illustrated in Figure 1.3. 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 are considerably small.

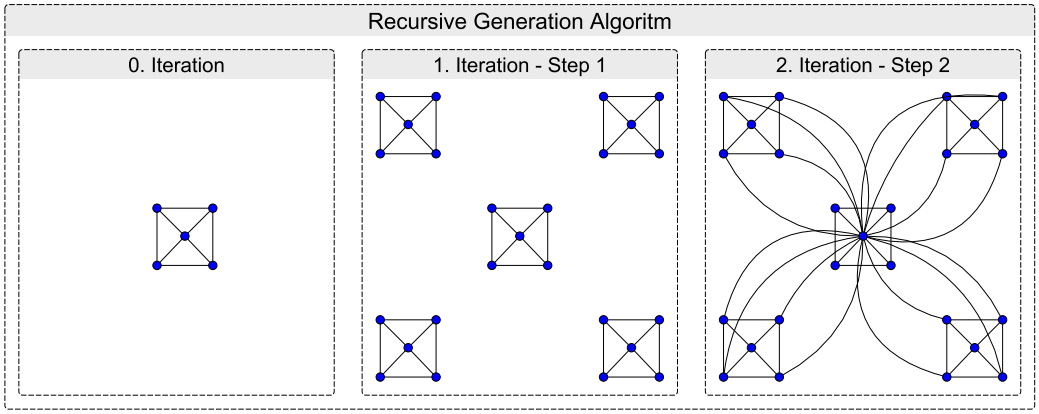


**Figure 1.3.** The first iteration in the recursive generation algorithm of the hierarchical network.

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

# Chapter 2

# Related Work

In the following sections, we introduce four benchmark frameworks that are designed to investigate the performance of RDF databases. In the end of the chapter 2.5, we summarize the frameworks and compare them to our work which is the introduction of a new approach to the existing benchmark frameworks.

## 2.1 Berlin SPARQL Benchmark

The Berlin SPARQL Benchmark (BSBM) is settled in an e-commerce use case in which a set of products is offered by different vendors and consumers who posted reviews about the products [23]. BSBM measures the SPARQL query performance of RDF-based tools via realistic workloads of use case motivated queries. The benchmark defines three different use cases and a suite of benchmark queries—a *mix* of queries—in each of them [4], simulating the search and navigation pattern of a consumer looking for a product. The queries include read and update operations as well.

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

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

- *Queries per Second*: It is equal to the number of queries executed within a second.
- *Query Mixes per Hour*: Denotes 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 require.

## 2.2 DBpedia SPARQL Benchmark

The DBpedia SPARQL Benchmark (DBPSB) proposes a benchmark framework for RDF databases based on the DBpedia [7] knowledge base, and it measures the performance of real queries that were issued against existing RDF data [32]. 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*.

## 2.3 SP$^2$Bench

The SPARQL Performance Benchmark (SP$^2$Bench) [36] is based on the *Digital Bibliography and Library Project* (commonly known as *dblp*) which provides an open bibliographic information on major computer science publications [8]. 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.

## 2.4 Train Benchmark Framework

### 2.4.1 Overview

The basic idea and elaboration of the Train Benchmark framework was introduced 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 and graph databases.

An overview of the Train Benchmark framework is depicted in Figure 2.1, illustrating the framework how connects the model validations to the database systems. In the 1. step, 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 the 3. step, 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 own 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.

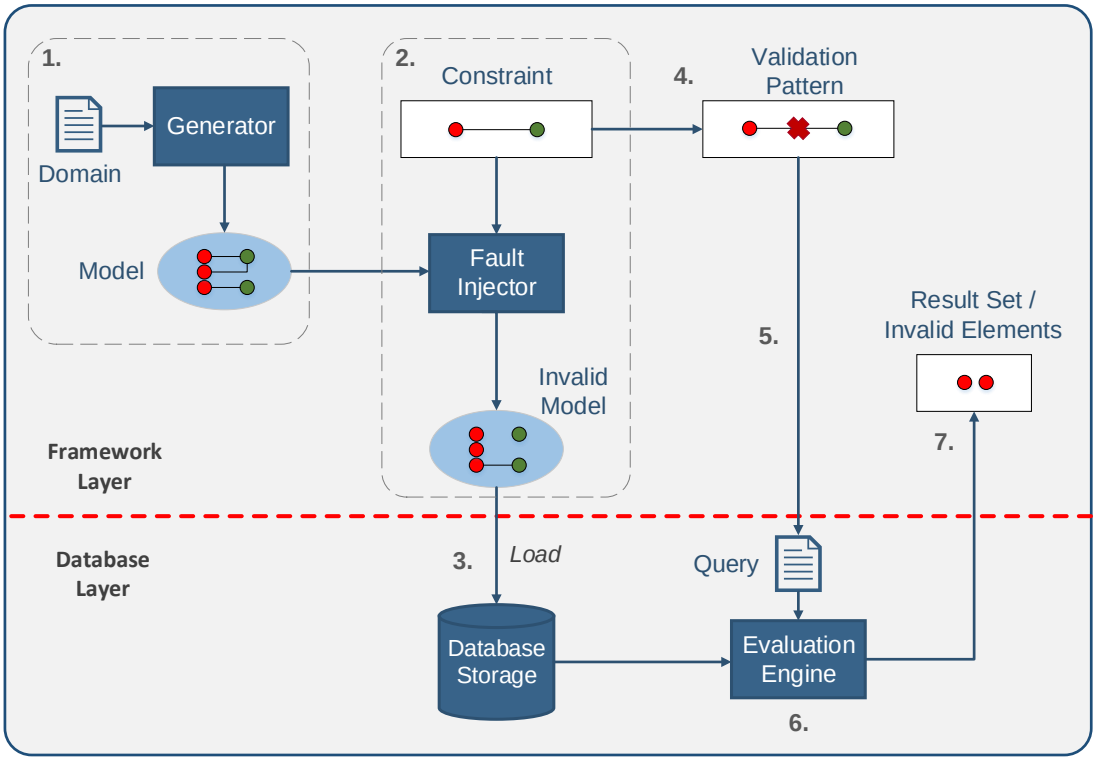


**Figure 2.1.** An overview of Train Benchmark.

### 2.4.2 Model Generation

The Train Benchmark framework uses artificially generated graph-based models for the measurements over a *Railway* domain, illustrated in Figure 2.2. 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 (with a specific length) 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 [18].

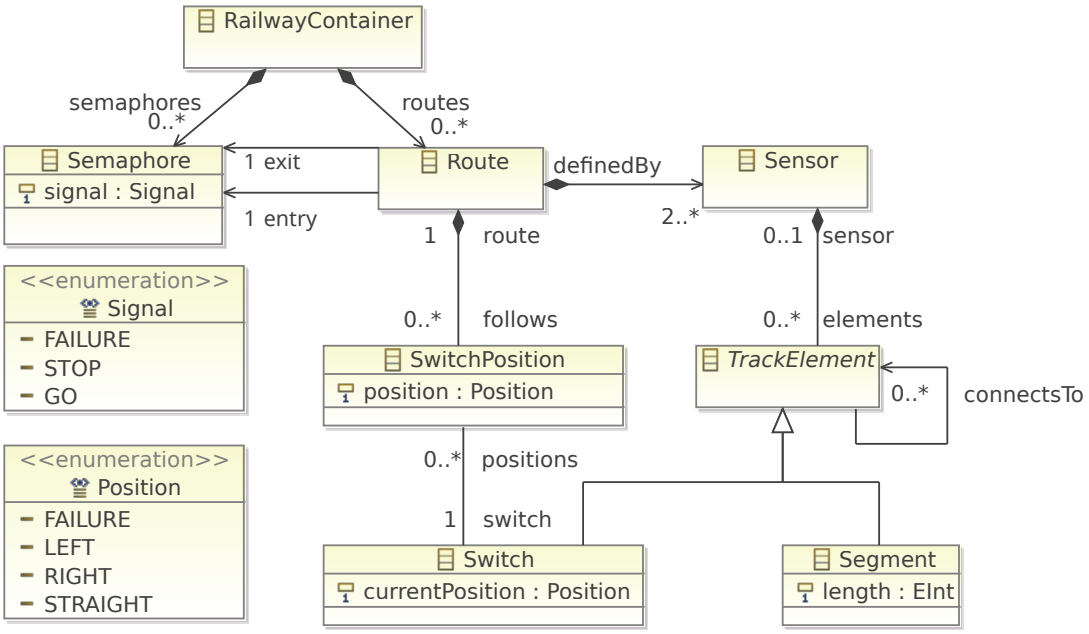


**Figure 2.2.** The Railway domain of Train Benchmark.

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.

## 2.5 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 comparison by concentrating on model validations via graph queries. As a conclusion, these frameworks guarantee a comprehensive performance evaluation of a workload—the ensemble of 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. BSBM, DBpedia or the SP$^2$Bench framework do not consider model modifications in their workloads, and thus, 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. Even though Train Benchmark considers model transformations, yet, it modifies only a subtle set of the elements during the measurements, 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 effect to the performance.

In order to introduce a new approach to the field of graph-based benchmark frameworks, we focus on elaborating a framework that proposes various models with different characteristics and investigates the relationships between the performance and the internal structure of the model.

The main features of the introduced frameworks and our new approach are summarized in Table 2.1.

| Feature | BSBM | Dbpedia | SP$^2$Bench | Train Benchmark | Our Approach |
|---|---|---|---|---|---|
| Based on a real-life model | | • | • | | • |
| Realistic domain-based queries | • | • | | | |
| Systematic queries | | | • | | • |
| Model transformations | | | | • | |
| Dynamic models | | | | | • |
| Measurement of scalability | • | • | • | • | • |
| Performance metrics | • | • | • | | |
| Model metrics | | | | | • |
| Query metrics | | | | | • |
| Extended scope besides RDF | | | | • | |

**Table 2.1.** A comparison between the existing benchmark frameworks and our approach.

Basically, we concentrate on the generation of dynamic models with different internal connectivities based on a real-life data, and we assess the evaluation time of systematic queries that test specific features of the models. In this part of our research, we highly focus on model and performance correlations.

# Chapter 3

# Design

## 3.1 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 3.1. We rely on two existing frameworks in our work: the Train Benchmark and MONDO-SAM frameworks. Furthermore, we the elaborate MONDO-MAP (MONDO-Metrics-Based Analysis of Performance) and the Homogeneous Graphs Benchmark.

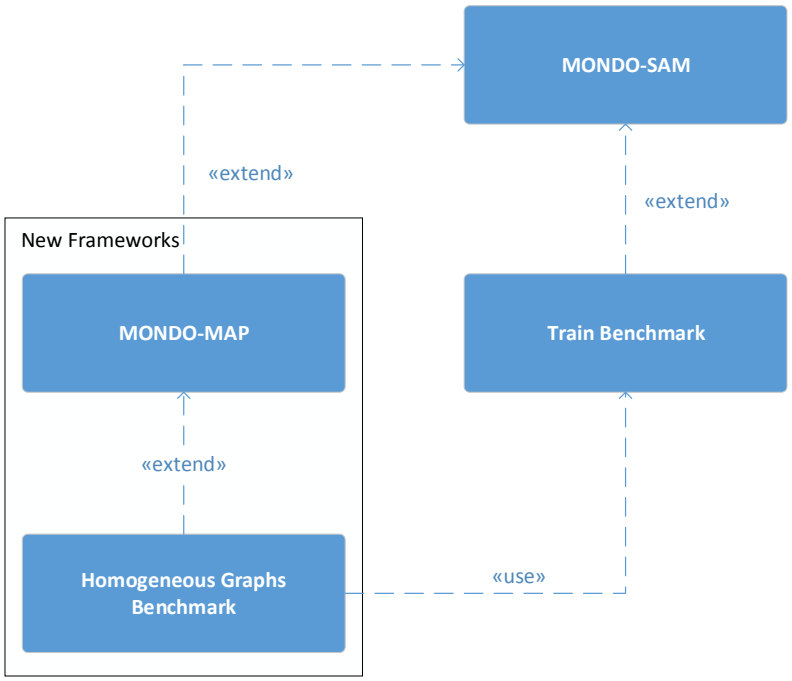


**Figure 3.1.** An overview of the frameworks in our approach.

**MONDO-SAM** The MONDO-SAM framework was created under the project MONDO (Scalable Modeling and Model Management on the Cloud) [11] 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 2.4.

**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 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 database. The goal is to generate artificial graphs with various characteristics and obtain a fluctuation in their descriptive metrics, and thus showing a quantitative connection between the model metrics and performance. As Figure 3.1 suggests, in the HG Benchmark we use a part of the components of Train Benchmark that are adequate for our purpose as well.

Figure 3.2 illustrates the main concept of our work. First, the HG Benchmark generates different $K$ number of graph topologies. Using the graph metric definitions from MONDO-MAP (2.), we calculate the descriptive metrics for every topology in step 3. As it is showed, every topology contains an own values for every $N$ number of metrics, and they are stored in MONDO-SAM. In the next two steps (4-5.) we define a query and evaluates it on every topology in the particular RDF database that we intend to measure. 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.

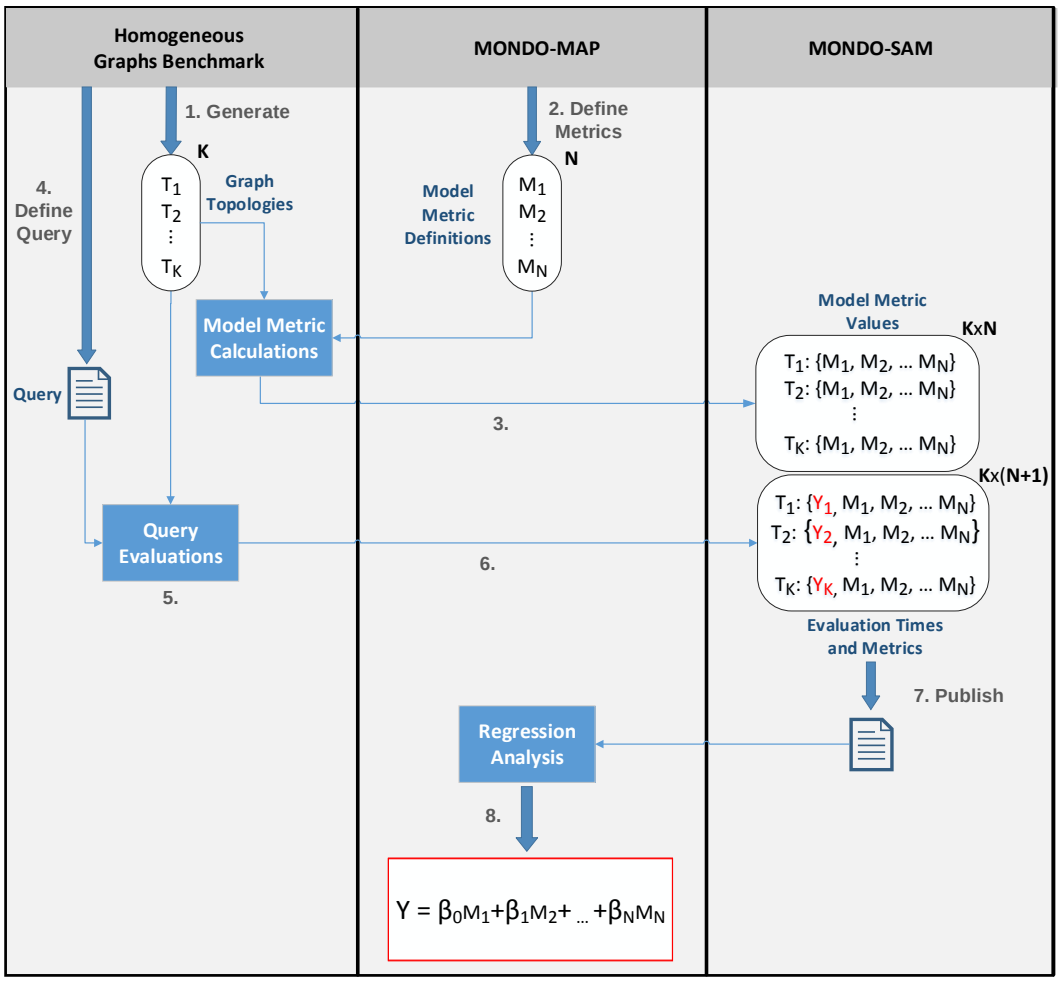


**Figure 3.2.** The main concept of the metric-based performance analysis.

## 3.2 Models and Metrics

### 3.2.1 Real-Life Networks

In the discipline 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 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 hierarchical network.

For example, the network of world-wide-web is studied in [34] and [28] as well, and the authors observe that the degree distribution of the *www* follows a power-law distribution

with a heavy tail[1], 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 [21], as they review the advances of different publications and investigate the characteristics of different real-life networks (Table). 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 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 [33]. Actually, none of the artificial topologies can be identified perfectly to real-life models, however, the ensemble of the specific attributes of these well-known topologies are frequently appear in them. As a conclusion, we conjecture that the characteristics belonging to the well-known topologies are the foundations of our solution, namely, finding those representative metrics of graphs that are suited to characterize not only the model appropriately, but the relationship between model and performance as well.

### 3.2.2 Network Topologies and Representative Metrics

Barabási and Albert inspect the natures of the well-known graph topologies in [21], 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 [35], the following metric deviations are assumed between the four topologies, illustrated in Table 3.1. 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 3.1 demonstrates, each

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

**Table 3.1.** 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 fluctuations in the performance of

[1]The concept of heavy tail means that there is a larger probability of receiving significantly higher values, than it is normally expected [10].

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

However, the values of metrics in Table 3.1 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.

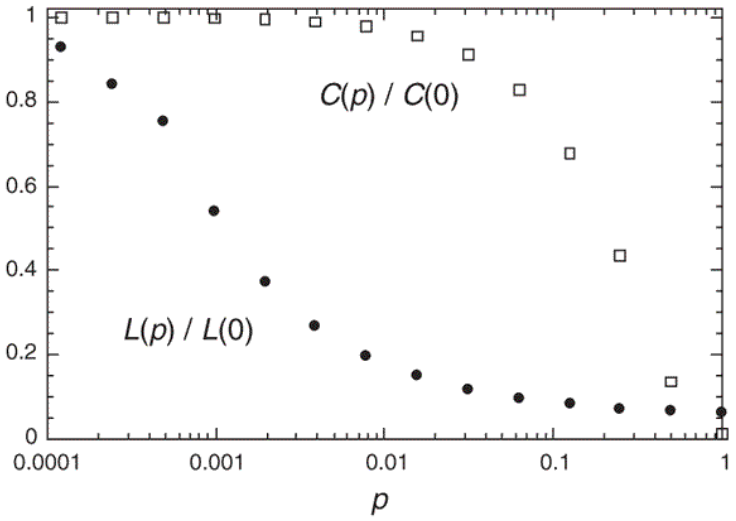


**Figure 3.3.** Characteristic path length $L(p)$ and clustering coefficient $C(p)$ of Watts-Strogatz model

By modifying $p$, the Watts-Strogatz model represents a bridge between a lattice and a random graph. As Figure 3.3 illustrates[2], the clustering coefficient ($C(p)$) and 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 3.1, 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 introduce 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 minor modifications, the topologies and the related metrics are showed in the extended Table 3.2. The WS-x abbreviations symbolize the Watts-Strogatz models indicating that the $p$ value is equal to $x$. The values of the betweenness centrality are determined by our initial conjecture 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

[2]The original figure can be found in [38].

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 3.2.** Graph topologies and their descriptive metrics with extensions

## 3.3 Metric and Performance Comparison

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

A similar problem is studied in [31] and [22], 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 [31], the algebraic connectivity 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 [22] 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 3.2), we expect that we cannot find a correlation between one metric and the performance, hence, we conjecture 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.

### 3.3.1 Sample Choosing

By using regression analysis on a sample, it is inevitable to regard the sample to be unbiased. In our case, a bias in a sample of graph topologies means a fluctuation 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 with respect to the amount of nodes and edges, even in the case of different topologies as well.[3]

[3]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 4

# Contributions

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

## 4.1 Overall Architecture

Figure 4.1 depicts the frameworks and the main components belonging them, additionally, it also denotes that which components are reused from Train Benchmark.

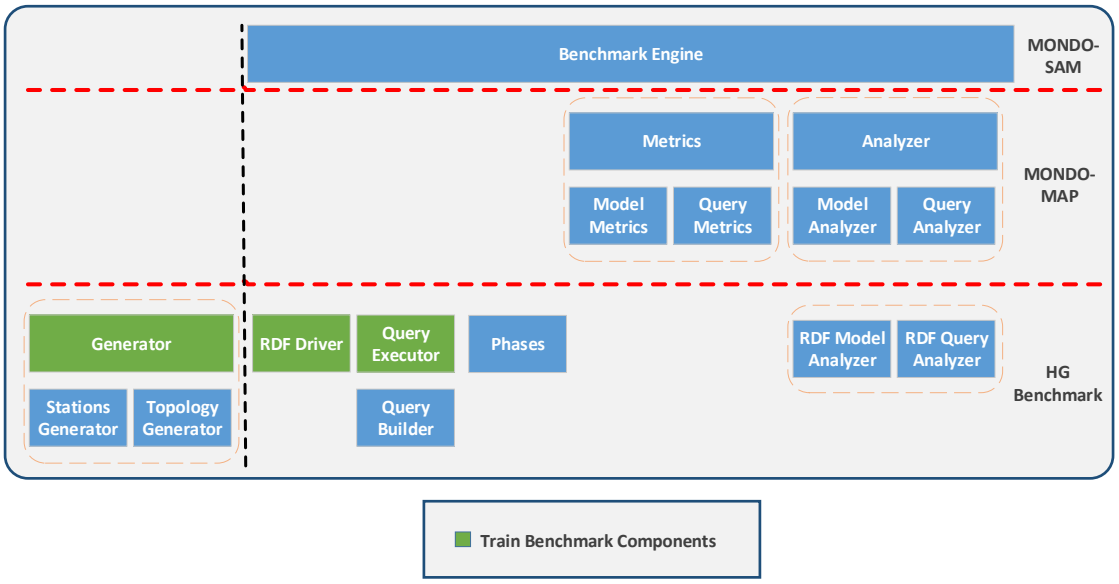


**Figure 4.1.** The architecture of our approach.

In the following parts, 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 metrics—such as Model Metrics and Query 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.
The query metrics connect to the definitions of the queries, and they do not relate to the query performance or the result set of the evaluations.

**Generators** The generator components belong to the HG Benchmark. The abstract Generator unit is utilized from Train Benchmark. The Stations Generator component is responsible for transforming the real-life British Railway Stations model to RDF. 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 achieved by the Query Executor component. The Query Builder is responsible for creating and altering the query definitions in runtime.

## 4.2 British Railway Stations

In order to test our regression models on an entirely different graph, we introduce a real-life model to our work. We use a model of train stations provided by the *Network Rail* company that runs, maintains, and develops Britain's rail tracks [12]. Network Rail publishes a number of different data available to developers, one of them is a *daily extracts and updates of train schedules* [19] that we intend to use.

The data contains approximately 10 000 British stations that belong to train schedules as destinations. We concentrate on the network of stations, and map it to a graph where every

vertex symbolizes a station. An edge is drawn between two nodes—so two stations—if these stations appear consecutively in a destinations path belonging to a schedule.

## 4.3 Uniform Model Generation

It is an essential expectation from 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.

### 4.3.1 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 $|V| = N$. As a conclusion, a precise number of nodes can be obtained regarding these topologies.

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

#### The Termination of the Hierarchical Network Algorithm

Since the generation of hierarchical network is recursive instead of being incremental—as in the case of the three other topologies—it is inevitable to determine a termination from the recursion. 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 failures have to be managed.

The possible problematic cases are demonstrated in Figure 4.2. 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 clone 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.

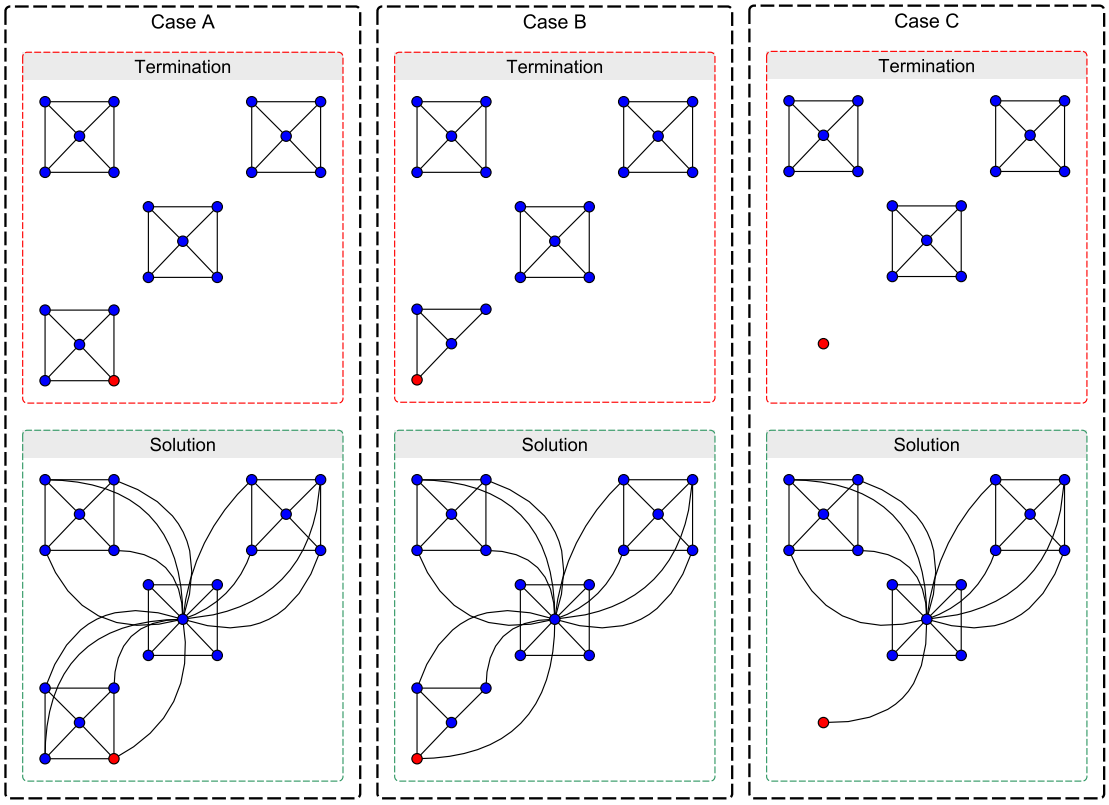


**Figure 4.2.** Possible termination problems in the hierarchical graph generation

### 4.3.2 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. Actually, 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 occurs 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.

#### Estimate 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 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 cluster is a $K_n$ complete graph.
- $F_i$: It indicates the constructed graph after the $i$ iteration.
- $|E_{F_i}|$: The number of edges of $F_i$.

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

$$|E_{F_1}| = (c+1) \cdot |E_{K_n}| + c \cdot (n-1) \tag{4.1}$$

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 $c$ number of 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 the algorithm—in the $i$ iteration—connects $n-1$ peripheral nodes from $c$ number of replicas of $F_{i-1}$, and due to $F_{i-1}$ includes $c^{i-1}$ number of clusters, we obtain

$$|E_{F_i}| = (c+1) \cdot |E_{F_{i-1}}| + c \cdot c^{i-1} \cdot (n-1) = (c+1) \cdot |E_{F_{i-1}}| + c^i \cdot (n-1) \tag{4.2}$$

We are not finished yet, since Equation 4.2 is equal to the number of edges of a completely finished hierarchical network. However, the generation algorithm in MONDO-MAP 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

$$|E_H| = |E_{F_i}| \cdot \frac{|V_H|}{|V_{F_i}|} \tag{4.3}$$

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

By using Equation 4.3, we can calculate the number of edges of a hierarchical graph and configure the other topologies to reach the same quantity.

#### Configure Random Graph

From the two 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 [26], the $p$ probability can be calculated from the number of nodes and edges as follows:

$$p = \frac{|E_H|}{\binom{|V|}{2}} \tag{4.4}$$

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

#### Configure 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 ($K_1$) and upper ($K_2$) bound for $K$, as $K \in [K_1, K_2]$, and we also assign a $p$ probability to $K$ that determines the likelihood of $K$ 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 $|E_H|$.

#### Configure Scale-Free Model

The scale-free topology is generated incrementally, since every step a new node is inserted to the graph with $m$ number of new connections. To obtain $|E_H|$, 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 (4.3.2), this constant value is converted to a random variable based on a particular probability, derived from $|E_H|$.

### 4.3.3 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 $n$.

## 4.4 Performance Analysis

### 4.4.1 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 evaluation engine in MONDO-SAM.

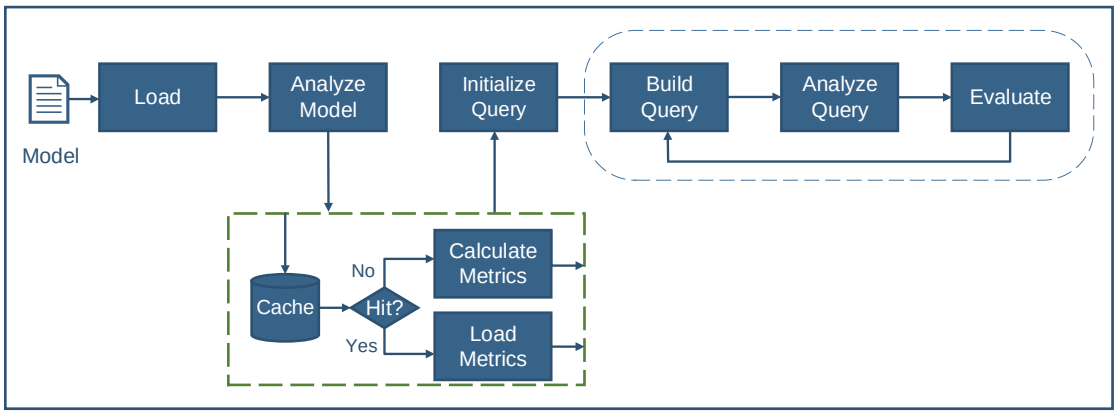


**Figure 4.3.** The workflow of the HG Benchmark.

The workflow of the HG Benchmark is represented in Figure 4.3. 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 model's metrics can be calculated only once in the beginning of the workflow, and 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 building and evaluating phases can be repeated implicating that more then one query can be evaluated in a sequence, even with different definitions.

### 4.4.2 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 syntaxes. 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 requires the most cost, hence, the framework searches a limited number (100) of shortest paths between randomly chosen vertices and calculates the average length of them.
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—which is 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 individual per nodes, we use the maximum of them.

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

### 4.4.3 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 investigate the spread of information in the graphs.

#### Shortest Path Query

The SPARQL definition of the query is shown below:

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

The query uses the * operator from SPARQL Property Paths [17] in the (base:neighbor)* predicate which means an arbitrary length of path with the neighbor predicates. The query is considered as a parameterized query in which we inject two random identifiers instead the ID parameters. Finally, the query searches a path between those two randomly chosen 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 that 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 showed 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) .
```

```
}
```

### 4.4.4 Tools

We concentrate on RDF tools in our work which are listed in Table 4.1. 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 | Programming Language | Version | Query 1 | Query 2 |
|---|---|---|---|---|
| AllegroGraph [2] | C, C++ | 5.0.0 | • | • |
| Blazegraph [5] | Java | 1.5.2 | • | • |
| 4store [1] | C | 1.1.5 | | • |
| Jena TDB [3] | Java | 2.13.0 | • | • |
| Sesame [16] | Java | 2.7.9 | • | • |

**Table 4.1.** The implemented tools in Train Benchmark

### 4.4.5 Regression Analysis

#### MARS

# Chapter 5

# Evaluation

(size/evaluation time plot?) sample results per tools regression -> residuals -> series mars metrics calculation

## 5.1 Sample

### 5.1.1 Framework Configuration

### 5.1.2 Evaluated Queries

## 5.2 Benchmarking Environment

## 5.3 How to Read the Charts

## 5.4 Model Analysis

## 5.5 Performance Analysis

### 5.5.1 Multiple Regression Analysis

### 5.5.2 MARS

## 5.6 Conclusions

# Chapter 6

# Summary

# Acknowledgements

Thanks!

# List of Figures

# List of Tables

# Bibliography


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

[2] AllegroGraph. http://franz.com/agraph/allegrograph/.

[3] Apache Jena TDB. https://jena.apache.org/documentation/tdb/index.html.

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

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

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

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

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

[9] A guide on probability distributions. http://dutangc.free.fr/pub/prob/probdistr-main.pdf.

[10] Heavy tail distribution. https://reference.wolfram.com/language/guide/HeavyTailDistributions.html.

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

[12] Network Rail. http://www.networkrail.co.uk/about-us/.

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

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

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

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

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

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

[19] Train Schedules Data. `http://nrodwiki.rockshore.net/index.php/SCHEDULE`.

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

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

[22] 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.

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

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

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

[26] 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.

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

[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] 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.

[32] 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.

[33] 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.

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

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

[36] 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.

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

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