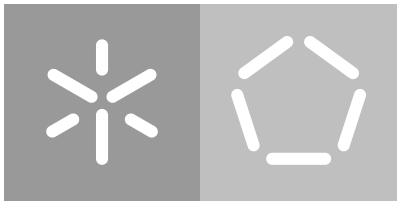


Universidade do Minho
Escola de Engenharia
Departamento de Informática

Diogo Batista Lima

Development and implementation of a repository for transcriptional regulatory information

November 2019



Universidade do Minho
Escola de Engenharia
Departamento de Informática

Diogo Batista Lima

Development and implementation of a repository for transcriptional regulatory information

Master Dissertation
Master Degree in Bioinformatics

Supervisors
Oscar Manuel Lima Dias

November 2019

DIREITOS DE AUTOR E CONDIÇÕES DE UTILIZAÇÃO DO TRABALHO POR TERCEIROS

Este é um trabalho académico que pode ser utilizado por terceiros desde que respeitadas as regras e boas práticas internacionalmente aceites, no que concerne aos direitos de autor e direitos conexos.

Assim, o presente trabalho pode ser utilizado nos termos previstos na licença abaixo indicada.

Caso o utilizador necessite de permissão para poder fazer um uso do trabalho em condições não previstas no licenciamento indicado, deverá contactar o autor, através do RepositóriUM da Universidade do Minho.

Licença concedida aos utilizadores deste trabalho



Atribuição
CC BY

https://creativecommons.org/licenses/by/4.0/

ACKNOWLEDGEMENTS

A realização desta dissertação contou com importantes apoios, aos quais estou profundamente grato. Em primeiro lugar agradeço ao meu orientador Professor Doutor Oscar Dias por todo o apoio, aconselhamento e amizade que me cedeu ao longo de todo o projeto. Realço e sinto-me profundamente agradecido pela confiança e oportunidades que me foram concedidas.

Naturalmente, não podia deixar de vincar uma forte referência ao Fernando. Tenho uma gratidão imensa por todo o apoio que prestaste e por tudo o que me ensinaste. Obrigado pelo acompanhamento, pelas conversas, pela picanha assada a saber a sardinha e pela tua companhia desde o primeiro dia.

Um particular agradecimento ao Bastos, Davide, Marta e Sophia por tudo o que me ajudaram, assim como a todos os membros do BiSBII. Não poderia deixar de dedicar uma particular menção ao José Miguel pela amizade, por todas as boleias e por aturar os meus devaneios e feitio. Deixo também uma nota de agradecimento ao Emanuel, Óscar e a todos os colegas de curso pela amizade e pela partilha de conhecimentos.

Apesar de passarmos um ano totalmente distinto, agradeço também ao Amaro, André, Cláudio e Renato pela amizade e pelas constantes conversas da bola. A ti Sónia, vinte dissertações não seriam suficientes para te agradecer. Foste e serás a minha melhor amiga, a minha melhor conselheira e a minha melhor companhia. Devote mais do que alguma vez seria justo colocar em papel. Que a concretização deste projeto abra portas para concretizarmos os nossos objetivos, de preferência lado a lado (com o Zero, a Katara e o Link a nosso redor).

Aos meus pais, pobres coitados que foram sujeitos a grafos e dados da *UniProt* mais vezes do que qualquer pessoa saudável deveria, agradeço do fundo do meu coração. O meu sucesso será o vosso sucesso. Finalmente, ao Ricardo, sem esquecer a Vânia, agradeço por me tornar o homem que sou hoje. Que o teu rebento tenha tanto orgulho do pai, como o futuro tio tem.

This work is a result of the project 22231/01/SAICT/2016: "Biodata.pt – Infraestrutura Portuguesa de Dados Biológicos", supported by the PORTUGAL 2020 Partnership Agreement, through the European Regional Development Fund (ERDF).

BioData.pt

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

ABSTRACT

The development of an integrated framework to infer genome-scale regulatory networks requires comparative genomics or experimental data. The former requires the compilation of transcriptional regulatory data, as the results of predictions may depend on the quantity and quality of template data. The automatic inference of gene regulatory networks has multiple applications in systems biology, such as the integration of regulatory data in *Genome-Scale Metabolic (GSM)* models to improve the accuracy of phenotypic predictions.

In this work, we have implemented a state-of-the-art repository of prokaryotic *Transcriptional Regulatory Network (TRN)*s and several tools to analyse and handle the regulatory information. Regulatory networks are a set of highly interconnected entities, therefore a graph database management system was used to store the retrieved data. *Neo4j* was the chosen framework due to its efficiency, abundant documentation and compatibility with multiple programming languages, such as *Python* and *Java*.

The repository implementation was performed by retrieving regulatory data from five online databases and five studies of TRNs on prokaryotic organisms. Resources of genomics and proteomics data, such as *Universal Protein Resource (UniProt)* and *National Center for Biotechnology Information (NCBI)* were also used to complete the repository. Based on the data warehousing procedures, *Extract Transform Load (ETL)* tools were used to assemble the main resource of regulatory information of this repository. The development of the repository also required the implementation of several subsystems for multiple purposes, such as data extraction, transformation, loading, request and visualization.

The unification of all extracted data resulted in the implementation of the most extensive resource of TRNs reported for prokaryotic organisms, with regulatory data for 510 species and approximately 1500 unique regulators controlling the expression levels of over 46000 target genes. In addition, a web application was developed to provide a user-friendly graphical interface for interested parties lacking programming knowledge, available at `http://bit.ly/ProTReND_DB`.

Keywords: Transcriptional regulatory networks, gene regulation, repository development, graph databases, data warehousing

RESUMO

O desenvolvimento de uma plataforma para inferir redes regulatórias requer dados de genómica comparativa ou experimentais. Isto implica a compilação de dados de redes regulatórias de transcrição, visto que os resultados das inferências podem depender da quantidade e qualidade dos dados modelo. A inferência automática de redes de regulação de genes tem várias aplicações em biologia de sistemas, nomeadamente a integração de dados regulatórios em modelos metabólicos à escala genómica de forma a melhorar a precisão de previsões fenotípicas.

Neste trabalho, foi implementado um repositório que reflete o estado da arte em redes regulatórias de transcrição, além de várias ferramentas para analisar e manipular os seus conteúdos. Posto isto, a escolha de sistema de gestão de bases de dados passou pela plataforma *Neo4j* (base de dados de grafos), devido à sua reconhecida eficiência, documentação abundante e compatibilidade com várias linguagens de programação, como *Python* e *Java*. A implementação do repositório foi possível através da extração de dados regulatórios de cinco bases de dados online, cinco estudos de redes regulatórias de transcrição em organismos procariontes e de recursos de dados de genómica e proteómica, nomeadamente, UniProt e NCBI. Baseado nas ferramentas ETL, tipicamente utilizadas em *data warehousing*, dividiu-se o repositório em vários sub-sistemas, direcionados para tarefas específicas como extrair, transformar e carregar dados. Adicionalmente, foram desenvolvidas ferramentas para automatizar processos de requisição e visualização de dados do repositório.

A união de todos os dados extraídos, resultou na implementação da fonte de redes regulatórias de transcrição de organismos procariontes mais extensa, até então reportada, fornecendo dados regulatórios para 510 espécies e aproximadamente 1500 reguladores distintos, responsáveis pelo controlo dos níveis de expressão de mais de 46000 genes. Complementarmente, foi desenvolvida uma aplicação *web* de forma a disponibilizar uma interface gráfica a possíveis interessados sem conhecimentos de programação. A plataforma encontra-se disponível em `http://bit.ly/ProTReND_DB`.

Palavras chave: Redes regulatórias de transcrição, regulação génica, desenvolvimento de um repositório, *Neo4j*, *data warehousing*

CONTENTS

1 INTRODUCTION 1
- 1.1 Context and motivation 1
- 1.2 Goals 2
- 1.3 Document organization 3

2 CONCEPTS AND STATE OF THE ART 5
- 2.1 Background 5
- 2.2 Resources of regulatory information 7
  - 2.2.1 *RegulonDB* 9
  - 2.2.2 *DBTBS* 10
  - 2.2.3 *CoryneRegNet* 10
  - 2.2.4 Collectf 11
  - 2.2.5 RegPrecise 11
  - 2.2.6 *GEO* 11
  - 2.2.7 *ArrayExpress* 12
  - 2.2.8 *M3D* 12
  - 2.2.9 *COLOMBOS* 12
  - 2.2.10 Other regulatory databases 13
  - 2.2.11 Literature 13
- 2.3 Resources of genomics and proteomics data 16
  - 2.3.1 *UniProt*: a universal protein resource 16
  - 2.3.2 NCBI: the National Center for Biotechnology Information 17
  - 2.3.3 *BioCyc* 17
- 2.4 Methodologies for TRN reconstruction 18
  - 2.4.1 Comparative genomics approach: template network-based 18
  - 2.4.2 Comparative genomics approach: *TFBS* data-based 18
  - 2.4.3 Gene expression data approach: *de novo* reverse engineering 20
- 2.5 Integration of regulatory information in *GSM* models 22
  - 2.5.1 *GSM* models: analysis and simulation 22
  - 2.5.2 Integration of transcriptional regulatory data in GSM models 24
- 2.6 Data warehousing 25
  - 2.6.1 Extraction 26

2.6.2 Transformation 26
2.6.3 Loading 27
2.7 Database management systems 27
2.7.1 Graph databases 31
2.7.2 *Neo4j* graph database management system 34
3 MATERIALS AND METHODS 36
3.1 Repository architecture 36
3.2 Data retrieval subsystem 38
3.2.1 Data extraction 38
3.2.2 Graph transformation 51
3.2.3 *Staging Area* loading 53
3.3 Knowledge expansion subsystem 54
3.4 Integration subsystem 55
3.4.1 Data transformation 55
3.4.2 Data loading 63
3.5 Request subsystem 68
3.6 Reporting subsystem 68
3.7 Web interface development 69
4 RESULTS AND DISCUSSION 70
4.1 *ProTReND* repository 70
4.2 *ProTReND* database structure 70
4.3 Request subsystem: command line interface 79
4.4 *ATRNI*: showcase 83
4.5 *ProTReND* web interface 87
5 CONCLUSION 93
5.1 Concluding remarks 93
5.2 Future directions 94
5.3 Outcomes 94

A SUPPORTING MATERIAL 110
A.1 *BLASTp* search results of regulators sharing a common name 110
A.2 *TRN Collection*: node and edge properties 111

LIST OF FIGURES

Figure 1	Negative control of the *trp* operon expression	6
Figure 2	*TFBS* data-based reconstruction of TRNs	20
Figure 3	Data warehousing and ETL tools	26
Figure 4	Comparison of relational and graph databases	33
Figure 5	Comparison of *SQL* and *Cypher*	34
Figure 6	A *Neo4j* prototype network	35
Figure 7	Architecture of the TRN repository	37
Figure 8	Data extracted from *RegPrecise*	45
Figure 9	*Graph transformation* of sample data	52
Figure 10	Merging of regulators in the *TRN Collection*	58
Figure 11	Merging binding sites	60
Figure 12	Automatic TRN Integrator input structure	65
Figure 13	Illustration of the *ProTReND* database schema	71
Figure 14	*ProTReND* database node and edge distribution	73
Figure 15	Frequency of extracted data by source	75
Figure 16	Organisms with most regulatory data	77
Figure 17	Frequency of regulator mechanisms	78
Figure 18	Common regulators in multiple organisms	79
Figure 19	Request command line interface: Organisms	80
Figure 20	Request command line interface: TRNs	80
Figure 21	Request command line interface: Regulation data	82
Figure 22	Request command line interface: Ubiquitous regulators	82
Figure 23	TRN example prior to *ATRNI* execution	83
Figure 24	Decision flowchart of the *ATRNI* tool	86
Figure 25	TRN example after the *ATRNI* execution	87
Figure 26	*ProTReND*: home page	88
Figure 27	*ProTReND*: gene search engine	89
Figure 28	*ProTReND*: organism detail page	90
Figure 29	*ProTReND*: regulator detail page	91
Figure 30	*ProTReND*: gene detail page	92

LIST OF TABLES

Table 1	Online resources with regulatory information (part 1)	8
Table 2	Online resources with regulatory information (part 2)	9
Table 3	Literature with complete TRNs (part 1)	14
Table 4	Literature with complete TRNs (part 2)	14
Table 5	*NoSQL* database models (part 1)	30
Table 6	*NoSQL* database models (part 2)	31
Table 7	Collected sources of transcriptional regulatory information	39
Table 8	Transcriptional regulatory data from *RegulonDB*	40
Table 9	Gene data from *RegulonDB*	41
Table 10	Transcriptional regulatory data from *DBTBS*	42
Table 11	Transcriptional regulatory data from *Collectf*	43
Table 12	Transcriptional regulatory data from *RegPrecise*	45
Table 13	Regulator data from *RegPrecise*	46
Table 14	Transcriptional regulatory data from *CoryneRegNet*	47
Table 15	Transcriptional regulatory data from *Fang et al.*	48
Table 16	Transcriptional regulatory data from *Faria et al.*	48
Table 17	Transcriptional regulatory data from *Arrieta-Ortiz et al.*	49
Table 18	Transcriptional regulatory data from *Turkarslan et al.*	50
Table 19	Transcriptional regulatory data from *Galán-Vásquez et al.*	51
Table 20	Summary of the contents of the *ProTReND* database	71
Table 21	Average execution time of TRN request	81
Table 22	*ATRNI* mock-up data	84
Table S1	BLASTp search results of common regulators	110
Table S2	*TRN Collection*: compendium of node properties	111
Table S3	*TRN Collection*: compendium of edge properties	115

ACRONYMS

A

API Application Programming Interface.

AR Atomic Regulon.

ATRNI Automatic Transcriptional Regulatory Network Integrator.

B

BLASTP Protein-protein Basic Local Alignment Search Tool.

C

CDS Central Data Storage.

CHIP Chromatin Immunoprecipitation.

CHIP-SEQ Chromatin Immunoprecipitation Sequencing.

CORYNEREGNET Coryne Regulatory Networks.

CSS Cascading Style Sheet.

CSV Comma Separated Values.

D

DBMS Database Management System.

DBTBS Database of Transcriptional regulation in *Bacillus subtilis*.

DNA Deoxyribonucleic Acid.

DPINTERACT DNA-Protein Interaction.

E

E-UTILS Entrez Programming Utilities.

EBI European Bioinformatics Institute.

ETL Extract Transform Load.

F

FASTA Fast-All.

FBA Flux Balance Analysis.

G

GENIE3 Gene Network Inference with Ensemble of Trees.

GEO Gene Expression Omnibus.

GRAM Genetic Regulatory Modules.

GSM Genome-Scale Metabolic.

H

HTML Hyper Text Markup Language.

HTTP Hyper Text Transfer Protocol.

J

JSON JavaScript Object Notation.

K

KBASE The Department of Energy Systems Biology Knowledgebase.

L

LEMONE Learning Module Networks.

M

M3D Many Microbe Microarrays Database.

MRNA Messenger Ribonucleic Acid.

MYCOREGNET Mycobacteria Regulation Networks.

MYSQL My Structured Query Language.

N

NCBI National Center for Biotechnology Information.

NOSQL Non Structured Query Language.

O

ODB Operon Database.

P

PROM Probabilistic Regulation of Metabolism.

PROTREND Prokaryotic Transcriptional Regulatory Network Database.

PWM Position Weight Matrix.

R

REGULONDB Regulon Database.

REST Representational State Transfer.

RFBA Regulatory Flux Balance Analysis.

RNA Ribonucleic Acid.

S

SBML Systems Biology Markup Language.

SMD Stanford Microarray Database.

SQL Structured Query Language.

T

TAXONOMY ID Taxonomy Identifier.

TF Transcription Factor.

TFBA Transcriptional-controlled Flux Balance Analysis.

TFBS Transcription Factor Binding Site.

TREMBL Translated European Molecular Biology Laboratory.

TRN Transcriptional Regulatory Network.

TSV Tab Separated Value.

TU Transcription Unit.

U

UNIPROT Universal Protein Resource.

UNIPROTKB Universal Protein Resource Knowledgebase.

URL Uniform Resource Locator.

X

XLSX Excel Microsoft Office Open XML Format Spreadsheet File.

XML Extensible Markup Language.

1

INTRODUCTION

1.1 CONTEXT AND MOTIVATION

The rise of next-generation sequencing techniques was responsible for a significant increase in publicly available genome sequences [1]. The relevance of systems biology increases with the amount of available biological data. This interdisciplinary field is aimed at understanding the complex interactions among various cellular components [2].

Understanding the complexity of sub-cellular interactions by systems biology led to the development of GSM models. These mathematical models represent the metabolic events of the organisms, helping therefore to elucidate their physiology [3]. Using different approaches, such as *Flux Balance Analysis (FBA)* [4], GSM models allow predicting the phenotype of an organism in various environmental and genetic conditions [5]. The implementation of GSM models has also contributed to the development of various fields such as metabolic engineering, as these systems biology tools provide detailed insight into the molecular machinery of organisms. Hence, the various applications of a GSM model, such as strain optimization, the discovery of novel metabolic pathways and synthetic lethality analysis, may enlighten metabolic engineering efforts [6].

One of the limitations of GSM models is the lack of information on gene regulation, thus impairing the accuracy of their phenotype predictions. This limitation can be overcome by integrating TRNs into the GSM models leading potentially to less false positives in the phenotype predictions [7]. The reconstruction of TRNs allows predicting regulatory mechanisms such as the RNA polymerase activity, binding of *Transcription Factor (TF)*s to *Transcription Factor Binding Site (TFBS)*s within *Deoxyribonucleic Acid (DNA)* coding sequences or operons, and sigma factors [8]. Collecting and understanding regulatory information is pivotal for further discovering novel interactions and molecular pathways which can lead to improvements of model predictions.

This project is part of a larger endeavor that aims at integrating TRNs into GSM models assembled with *merlin* [9] or other computational tools for GSM model reconstruction. The main focus of this larger project is on developing a software capable of inferring TRNs from previously available regulatory information and genomics data. Hence, this will require the development of an online repository capable of hosting all the regulatory data and supporting the development of the project.

A digital repository is a collection of information accessible via a network without prior knowledge of the structure of the collection [10]. Thus, the repository designed in this project will allow a simplified access to both data storage and sharing besides supporting the TRN inferring software. This will require the development of multiple computational tools for collecting and storing data retrieved from published TRNs and existing databases with regulatory information. As further discussed in this study, there are many different repositories containing regulatory data though there is still missing a unified platform that enables TRN reconstruction using prokaryotic genome data [11].

1.2 GOALS

The main goal of this work is to develop and implement a repository for storing TRNs of prokaryotic organisms. The development of the repository will initially focus into insights of well documented organisms such as *Escherichia coli* and *Bacillus subtilis*, and later expand upon other microorganisms.

The development of the repository can be divided in four different tasks, which can be seen as complementary:

- Extracting and processing TRNs and other regulatory data.
- Implementing the repository with extracted data.
- Designing additional software tools to integrate new data in the repository.
- Developing a system for retrieving and analyzing the contents of the repository.

Initially, the development of the repository proposed in this work will be focused on researching and obtaining available curated regulatory networks. While highly relevant to this work, the available curated TRNs do not provide enough data for reconstructing novel TRNs. Thus, the repository is also aimed at collecting crucial comparative genomics and TFBS data among other information from multiple resources.

Hence the importance of developing computational tools to automatize data extraction and pre-processing as well as performing homology and TFBS motif searches.

The database of the repository is to be implemented using *Neo4j* [12], a high performance *Non Structured Query Language (NoSQL)* [13] graph database management system. In addition, the computational tools developed in the scope of this project are to be combined with the container engine technology of *Docker* [14]. The containers created with *Docker* will store the code of the software together with additional requirements, such as other computational tools, libraries and configuration files.

1.3 DOCUMENT ORGANIZATION

Chapter 2 - Concepts and state of the art

This chapter focuses on contextualizing this work and describing the appropriate state of the art. Section 2.1 details the type of biological data subject to analysis on this document, by briefly explaining regulatory mechanisms and how TRNs will be integrated into a repository. Section 2.2 summarizes the features of the most relevant regulatory databases and discusses relevant literature. Section 2.3 consists of an overview of resources of genomics and proteomics data. Section 2.4 further explains methodologies for TRN reconstruction and validation. Additionally, section 2.5 briefly describes GSM models and approaches for integrating transcriptomics data within model simulations. A short overview on data warehousing and the ETL tools is provided in section 2.6. Finally, section 2.7 contextualizes database development by thoroughly describing different database management systems and focusing on the advantages of using graph databases in this work.

Chapter 3 - Materials and methods

The rationale behind the development and implementation of the repository, as well as the methodologies used throughout the process, are described in this chapter. Section 3.1 summarizes the architecture of the proposed repository. Sections 3.2, 3.3, 3.4, 3.5 and 3.6 provide extensive insight into the multiple subsystems developed in this work, namely, *Data retrieval*, *Knowledge expansion*, *Integration*, *Request* and *Reporting* subsystems. Chapter 3.7 summarizes the development of the web interface *Prokaryotic Transcriptional Regulatory Network Database (ProTReND)*.

Chapter 4 - Results and discussion

The results of the development of the repository and respective tools are presented in this chapter. Section 4.1 provides a short summary of the main results obtained in this project. Section 4.2 presents a detailed analysis on the structure and contents of the proposed repository. A demonstration of the command line interface implemented for requesting the contents of the database, is performed in section 4.3. Section 4.4 contains a demonstration of the *Automatic Transcriptional Regulatory Network Integrator (ATRNI)* tool for loading and updating TRNs in the database. Finally, ProTReND, the web application developed in this work, is briefly discussed in section 4.5.

Chapter 5 - Conclusion

A final conclusion of the work developed in this dissertation is provided in this chapter, as well as future work perspectives. Additionally, this chapter contains a short overview of the outcome of this project.

2

CONCEPTS AND STATE OF THE ART

2.1 BACKGROUND

The basic physiology of every living being is defined by cells, the "building blocks of life" [15]. Within an eukaryotic organism, all cells share the same DNA, thus sharing the same genetic code [16]. However, cells across the organism have different functions and are structurally distinct. This is explained by gene expression, meaning that although all cells share the same genetic material, there is a variation on which genes are "turned on" or "turned off" through transcription [17]. Prokaryotic organisms are unicellular and depend on gene expression regulation to adjust to changing environmental conditions. Within a prokaryotic organism, gene expression is controlled by signals released within the cell and by external stimuli, thus forming a regulatory network of both activated and silenced genes.

There are several regulation mechanisms, though prokaryotic organisms share common basics. *Ribonucleic Acid (RNA)* polymerases bind to well-defined DNA sequences, named promoters. This initiates gene transcription through the synthesis of *Messenger Ribonucleic Acid (mRNA)* molecules which will then be translated into proteins on the ribosomes [18]. A set of contiguous genes and its respective promoter forms an operon. Gene expression can be interrupted due to a repressor molecule that binds to the promoter region, stopping the binding of the RNA polymerase, thus halting gene transcription and effectively "turning off" the gene. An example for this negative control mechanism can be seen on Figure 1.

On the other hand, there are control mechanisms that enable the binding of activator molecules to the promoter region, allowing the RNA polymerase to bind and consequently activating the expression of such gene [19]. Most of these molecules, responsible for activating and repressing gene expression, are regulatory proteins called TFs. The regions of non-coding DNA in the genome where the transcription factors

bind at to regulate gene expression, constitute cis-regulatory elements named TFBSs. These regions are typically short DNA sequences of about 15-20 nucleotides.

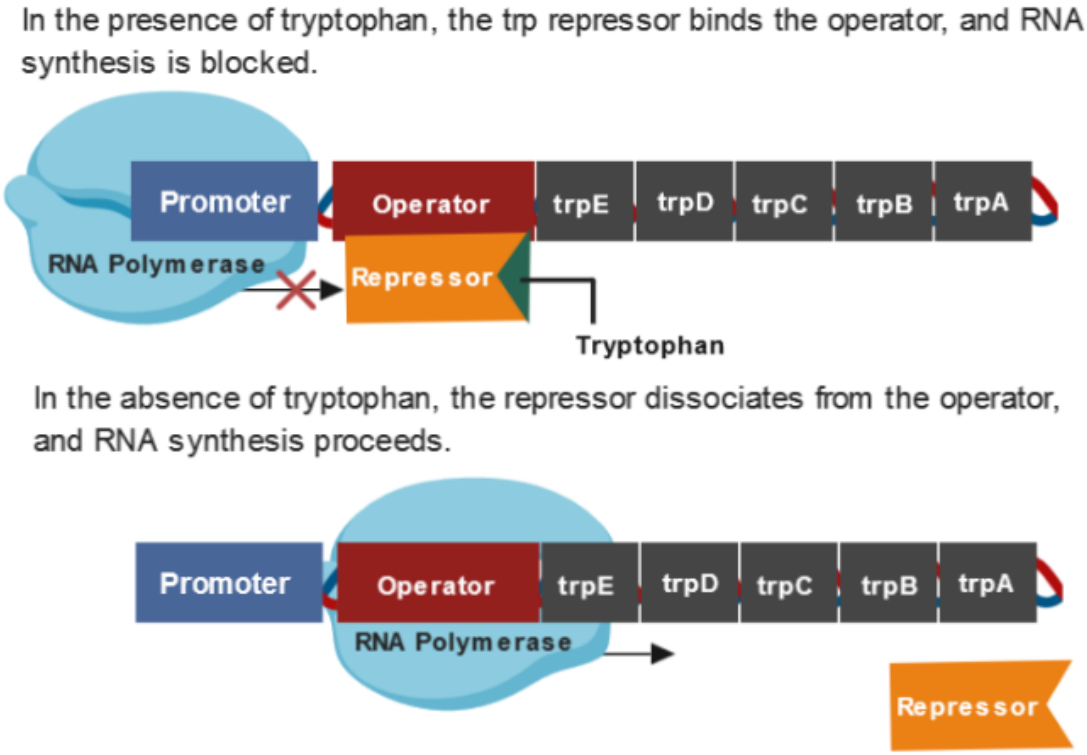


Figure 1.: The *trp* operon is transcribed solely on the absence of tryptophan. Alternatively, this amino acid can bind to the tryptophan repressor, which is responsible for repressing the *trp* operon expression. The repressor is a tryptophan transcription factor that binds close to the promotor region blocking the initiation of the operon transcription by the RNA polymerase. This phenomenon is known as negative control of gene regulation and is present in many bacteria. A set of interactions similar to this one, forms part of a TRN (adapted from Rye et al, 2017 [20]).

It is important to note that the control of gene expression in eukaryotic and prokaryotic organisms has different levels of complexity. In prokaryotic cells, gene expression is mostly controlled during transcription, whilst regulatory events can occur more frequently at the epigenetic, post-transcriptional and post-translational levels in eukaryotic cells [21]. In eukaryotic organisms, a complex compendium of interactions among several proteins is often responsible for controlling the expression of a single gene, while prokaryotic gene regulation usually involves a single repressor or activator, operating at the operon level [21]. Hence, due to a higher complexity and sheer vari-

ety of regulatory mechanisms in eukaryotic cells, it is simpler to analyse and predict regulatory interactions occurring in prokaryotic organisms.

The collection of known gene regulatory interactions occurring within an organism may be thought of as a TRN. Although different methodologies have been proposed throughout the years, there are two major approaches currently being used for reconstructing prokaryotic TRNs, namely comparative genomics-data and gene expression data-driven methodologies [22]. Yet, there are no clear advantages associated with each approach.

The deciding factor may be associated with the database providing the regulatory data [11], or the amount and type of available data. Organism-specific databases are regularly used in data-driven methods as these are intended for validation purposes, while for comparative genomics approaches, databases that cover a wider range of organisms are a better source of information. Regardless of the data source, usually the best approach for TRN reconstruction is a hybrid pipeline combining different complementary methods and algorithms.

2.2 RESOURCES OF REGULATORY INFORMATION

There are many different online databases that focus on documenting regulatory events happening inside of prokaryotic cells. From these online resources, it is possible to retrieve valuable information such as curated TRNs of prokaryotic organisms as well as TFBS and gene expression data. Tables 1 and 2 summarize the most relevant online resources for retrieving regulatory information such as curated TRNs and gene expression data.

Table 1.: Summary of the most relevant databases with transcriptional regulatory data or gene expression datasets, including organism coverage, availability of regulatory data and data access.

Database	**Organism (s)**	**Regulatory data**	**Data access**
RegulonDB [23]	*Escherichia coli*	Yes	Downloadable files
DBTBS [24]	*Bacillus subtilis*	Yes	No documentation
CoryneRegNet [25]	*Corynebacterium* and *Escherichia coli*	Yes	Downloadable files
Collectf [26]	Multiple	Yes	Downloadable files
RegPrecise [27]	Multiple	Yes	No documentation
GEO [28]	Multiple	No	E-Utils and R
ArrayExpress [29]	Multiple	No	REST Web Service and R
M3D [30]	Prokaryotes	No	Downloadable datasets
COLOMBOS [31]	Prokaryotes	Yes	REST Web Service and R

Table 2.: Availability of gene expression datasets and description of the most relevant databases with transcriptional regulatory and gene expression data.

Database	GE datasets	Description
RegulonDB [23]	Yes	Repository of *Escherichia coli* K12 regulatory data
DBTBS [24]	No	Repository of *Bacillus subtilis* regulatory data
CoryneRegNet [25]	No	Repository of *Corynebacterium* regulatory data
Collectf [26]	No	Extensive repository of regulatory data for 320 prokaryotic species
RegPrecise [27]	No	Extensive data repository covering over 450 prokaryotic species
GEO [28]	Yes	Repository of functional genomics data
ArrayExpress [29]	Yes	Repository of functional genomics data
M3D [30]	Yes	Repository of microbial gene expression data
COLOMBOS [31]	Yes	Repository of functional genomics data

2.2.1 *RegulonDB*

Regulon Database (RegulonDB) is a relational database of manually curated information regarding the transcriptional regulatory network of *Escherichia coli* K-12 [23]. This database supports queries for genes, operons, regulons and genetic sensory response units [32]. The database has comprehensive information about each gene, including its sequence, ontology, external links and respective operons. Importantly, each gene contains detailed information regarding the corresponding regulatory pro-

teins, which can be activators, repressors or both. The organization of genes in regulons is particularly useful as it further elucidates the regulatory interactions among the organism, thereby being an important tool for the TRN reconstruction process [33]. All the aforementioned information can be easily retrieved as a text file (or other formats), thus facilitating its integration into a new repository.

One of the most unique features of RegulonDB is a set of web tools that enables visual representation of the genome. In this web platform, it is possible to visualize the genomic context of each gene, thus enabling easy navigation both downstream and upstream of the gene which in turn allows for the contextualization of the operon and promoter regions.

2.2.2 *DBTBS*

Database of Transcriptional regulation in *Bacillus subtilis* (DBTBS) is an online repository of regulatory data of *Bacillus subtilis* [24]. This database was constructed using a collection of experimentally validated gene regulatory relationships and corresponding TFBSs, which makes up the main data source for transcription factors, regulatory operons and motif conservation regarding *Bacillus subtilis* exclusively. However, the repository does not contain gene expression data nor documentation regarding the programmatic access to the data.

2.2.3 *CoryneRegNet*

Coryne Regulatory Networks (CoryneRegNet) is a database for corynebacterial TRNs [25]. Even though the focus of the platform is on storing *Corynebacterium* regulatory data, there is also data for *Escherichia coli* K-12 strain. *CoryneRegNet* also provides a graphical tool for visualization of the genomic context. Most of the genes described in this database are associated with regulatory data, which was obtained from experimental evidences or inferred computationally. All data regarding each organism can be obtained as data files in text-file format. The platform contains documentation for programmatic access through *Representational State Transfer (REST)* Web Service, although it is unavailable at the time of writing.

2.2.4 *Collectf*

Collectf is an online database of experimentally validated TFBSs for several bacterial organisms [26]. The TFBS data is contextualized within the corresponding TRN, providing regulatory interactions between regulators and target genes. Among other features, it is possible to browse the database by organism phylum, thus quickly accessing regulatory data for several organisms at once. Although all regulatory data can be exported in several formats, such as *Fast-All (FASTA)*, *Tab Separated Value (TSV)* or *JASPAR*, this platform does not provide programmatic access through an *Application Programming Interface (API)*. Despite this limitation, it is possible to obtain all data into a single TSV file.

2.2.5 *RegPrecise*

RegPrecise is a database based on the regulon approach [27]. In prokaryotic organisms, groups of genes are usually regulated by a common regulatory protein, thus belonging to the same regulon. This database contains regulons obtained from manually curated experimental data as well as regulons inferred with comparative genomics approaches. The latter are sets of draft regulons reconstructed by automated propagation procedures based on template curated data.

RegPrecise is the most extensive source of TF-TFBS-gene interactions analyzed at the time of writing, providing regulatory data for over 450 prokaryotic species, including a vast majority of *Bacteria* species and two *Archaea* species. Additionally, the authors of this database have assembled an extensive reconstruction of the regulatory network of *Bacillus subtilis* using both experimental evidences and *in silico* predictions [22]. Nonetheless, the database does not provide programmatic access through an API nor downloadable flat files, thereby hindering data extraction.

2.2.6 *GEO*

Gene Expression Omnibus (GEO) is an open-access repository of gene expression data [28] that is part of NCBI [34]. This online database contains a large variety of gene expression datasets for both eukaryotic and prokaryotic organisms. Even though GEO is not focused on regulatory networks, this platform provides pivotal data for inferring TRNs.

The datasets can be accessed manually or programmatically through *Entrez Programming Utilities (E-Utils)* [35]. Moreover, the platform has very detailed documentation on advanced querying for improving search results. GEO also provides integrated tools such as clustering (with heatmaps as well) and statistical tests to ease the analysis of curated datasets.

2.2.7 *ArrayExpress*

ArrayExpress [29] is an archive of high-throughput functional genomics data that is part of *European Bioinformatics Institute (EBI)* [36] and *ELIXIR* infrastructure [37]. This data repository contains a large variety of gene expression datasets for both eukaryotic and prokaryotic organisms. Data can be accessed manually, although there are several options for obtaining it automatically. In fact, one is able to retrieve information from *ArrayExpress* with *R* [38] using the *Bioconductor* [39] package or other types of advanced querying. Though *ArrayExpress* is lacking insights into gene regulatory interactions, this database complements GEO by being one of the most complete sources of gene expression data.

2.2.8 *M3D*

Many Microbe Microarrays Database (M3D) is an open-access archive for microbial gene expression data [30]. This database contains gene expression datasets for prokaryotic organisms such as *Escherichia coli* and *Shewanella oneidensis*. Additionally, it also contains gene expression datasets for the baker's yeast, *Saccharomyces cerevisiae*. There is no documentation for programmatic access, but all datasets can be browsed and downloaded. The platform has a downloadable version of *CLR*, which is a direct inference algorithm for *de novo* reverse engineering gene expression data (see chapter 2.4.3).

2.2.9 *COLOMBOS*

COLOMBOS is a open-access web-based platform that provides a repository of prokaryotic gene expression data as well as a set of tools to explore such data [31]. Among other features, this web platform allows visualization of gene ontology information and heatmaps as well as gene network creation. The last-mentioned feature

is of particular interest, as it allows representing gene data into a network structure. This means that gene expression data can be visualized and manipulated in a similar fashion to a graph-based database. Relevant regulatory information such as genes, regulons and operons can also be easily obtained.

Most information available in the database was retrieved from GEO and *ArrayExpress*. All data can be accessed and downloaded manually, as well as programmatically using *R* or a REST Web Service [40]. *COLOMBOS* eases the access to regulatory interactions, such as molecular pathways or transcriptional regulatory events.

2.2.10 *Other regulatory databases*

cTFbase is an online database that only contains comparative genomics data of TFs in cyanobacteria [41]. Accordingly, the platform does not provide information regarding regulatory interactions, having only detailed descriptions of cyanobacteria TFs. *TractorDB* is an online collection of computationally predicted TFBSs in gamma-proteobacterial genomes [42].

DBD is a database of predicted TFs for several organisms, which can be seen as an additional source of regulatory information [43]. However, the search engine is offline at the time of writing. *Operon Database (ODB)* is a repository of conserved operons in multiple species, derived from available literature and databases [44]. Finally, *PRODORIC2* is a database about gene regulation in prokaryotes, that includes manually curated TFBSs and an integrated virtual footprint motif search algorithm for predicting TF regulons [45].

Furthermore, there are many other databases that would be worth mentioning, albeit they are not available at the moment of writing, due to offline servers or outdated back-end implementation; *Stanford Microarray Database (SMD)* [46], *DNA-Protein Interaction (DPInteract)* [47], *MTBRegList* [48], *Mycobacteria Regulation Networks (MycoRegNet)* [49], *ExtraTrain* [50] and *RegTransBase* [51].

2.2.11 *Literature*

Several studies have been published which focus on inferring and curating TRNs for well documented bacteria. Tables 3 and 4 summarize a set of papers which expanded upon previous TRNs, constituting important sources of regulatory data.

Table 3.: Studies which have expanded upon previous TRNs, including year of publication, target organism and number of regulators identified in the network.

Authors	**Year**	**Organism**	**Regulators**
Fang et al. [52]	2017	*Escherichia coli*	147
Faria et al. [33]	2016	*Bacillus subtilis*	275
Arrieta-Ortiz et al. [53]	2015	*Bacillus subtilis*	215
Turkarslan et al. [54]	2015	*Mycobacterium tuberculosis*	206
Galán-Vásquez et al. [55]	2011	*Pseudomonas aeruginosa*	90

Table 4.: Studies which have expanded upon previous TRNs, including number of target genes in the network and used data sources.

Authors	**Target Genes**	**Sources**
Fang et al. [52]	1764	*RegulonDB* and *Chromatin Immunoprecipitation (ChIP)* gene expression data
Faria et al. [33]	2500	*DBTBS*, *SporeWeb*, *Subtwiki*, *RegPrecise* and a microarray gene expression dataset from *GEO*

(To be continued)

Authors	Target Genes	Sources
Arrieta-Ortiz et al. [53]	3086	*Subtwiki* and two microarray gene expression datasets from *GEO*
Turkarslan et al. [54]	2519	*Chromatin Immunoprecipitation Sequencing (ChIP-seq)* gene expression and TF overexpression data
Galán-Vásquez et al. [55]	593	Literature

Recently, *Fang et al.* [52] published a TRN for *Escherichia coli*, providing further insight regarding the regulatory machinery of this gram-negative bacteria. The authors reconstructed a high confidence TRN consisting of 147 TFs regulating 1764 genes, across 3797 regulatory interactions. In further detail, the data was gathered from curated ChIP [56] and *RegulonDB* data. The authors have inferred regulatory interactions for 1364 transcriptional units, sets of co-transcribed genes, using supervised machine learning algorithms.

Moreover, *Faria et al.* [33] published a regulatory network for *Bacillus subtilis*, a model gram-positive bacterium, and reconciled the results with gene expression data. The authors extracted regulatory data from several sources, such as *DBTBS*, *SporeWeb* [57], *Subtwiki* [58] and *RegPrecise*. Then, the authors validated the reconstructed network with gene expression data from GEO. The proposed regulatory network describes 275 regulators and approximately 2500 target genes. It is worth noting that the reconstructed regulatory network includes data for other regulators besides TFs, such as riboswitches and sigma factors, thus comprising a substantial diversity of regulatory mechanisms.

Arrieta-Ortiz et al. [53] published a TRN for *Bacillus subtilis*, thus deepening existing knowledge regarding the regulatory machinery of this microorganism. The resulting TRN was reconstructed from previously validated regulatory interactions, available in *Subtiwiki*, and further inferred from two transcriptomics datasets. The TRN reconstruction process was achieved by employing the *Inferelator* algorithm [59]. Hence, by combining the aforementioned data sources it was possible to infer a TRN for *Bacillus subtilis* covering over 3000 genes. The studies performed by *Faria et al.* and

Arrieta-Ortiz et al. can be seen as complementary, as both authors aimed towards the same goal: expanding existing knowledge regarding the regulatory events occurring in *Bacillus subtilis*.

Additionally, *Turkarslan et al.* [54] published a TRN for *Mycobacterium tuberculosis*, a pathogenic bacterium responsible for tuberculosis infection. The proposed regulatory network was expanded upon a ChIP-seq [60] and a TF overexpression [61] datasets, describing 206 TFs.

Finally, *Galán-Vásquez et al.* [55] published a regulatory network for *Pseudomonas aeruginosa*, an opportunistic pathogen bacterium prominent for its resistance to antibiotics and disinfectants [62]. The resulting regulatory network was compiled from data previously available in relevant literature and consists of 1020 regulatory interactions involving 593 genes.

2.3 Resources of genomics and proteomics data

2.3.1 *UniProt: a universal protein resource*

UniProt is a central hub for a massive collection of functional information on proteins [63]. This online resource encompasses amino acid sequences, protein names and functions, as well as external links to other resources. *Universal Protein Resource Knowledgebase (UniProtKB)* is one of the multiple databases hosted by UniProt, and it can be divided into two sections: *Swiss-Prot* and *Translated European Molecular Biology Laboratory (TrEMBL)* [64]. The former contains manually annotated and reviewed data, on the other hand, the latter contains computationally analyzed records awaiting manual curation. Thus, UniProtKB entries are classified by curation level, enabling data filtering by a confidence degree. The *Swiss-Prot* section of the database contains over 560000 curated entries describing proteins across a myriad of organisms, including *Bacteria* and *Archaea* species.

Furthermore, each UniProtKB entry is associated with a unique *Uniprot Accession*, making it possible to label different proteins with a unique identifier. An entry might also include the common name of the given protein and the corresponding gene locus tag. Protein amino acid sequences are easily retrievable in various formats, such as FASTA, *Extensible Markup Language (XML)* as well as flat text files. Hence, UniProtKB is a highly relevant resource that provides crucial information to annotate proteins.

For instance, the amino acid sequences of regulatory proteins are key components for certain TRN reconstruction approaches (see chapters 2.4.1 and 2.4.2).

2.3.2 *NCBI: the National Center for Biotechnology Information*

NCBI is an online compendium of resources for biological data which can be divided in six sections; Literature, Health, Genomes, Genes, Proteins and Chemicals [34]. *GenBank* [65] is a widely used resource from NCBI, which provides nucleotide sequences for almost 260000 species. *PubMed* is another example of a highly relevant resource from NCBI, comprising over thirty million citations for biomedical literature from a variety of sources.

The NCBI *Taxonomy Database* [66] is another relevant resource from NCBI providing standardized organism names and taxonomic lineages for manifold species and respective strains. Additionally, every organism portrayed in the NCBI *Taxonomy Database* is associated with a unique *Taxonomy Identifier (Taxonomy ID)*, which can be used as standard nomenclature for comparing organisms. As the *Taxonomy Database*, *GenBank*, *PubMed* and GEO are integrated within the NCBI ecosystem. Hence, different types of relevant information for reconstructing TRNs can be easily attainable from a single hub page.

2.3.3 *BioCyc*

BioCyc is an open-access online database housing pathway and genome data for manifold organisms [67]. It is comprised of multiple sub-platforms such as *EcoCyc* [68] and *BsubCyc* [67] which provide information on *Escherichia coli* and *Bacillus subtilis* respectively. All sub-platforms are integrated, thus it is possible to obtain information regarding different organisms using the same tools. *BioCyc* contains a great variety of genomics and metabolomics data. It is an extensive database that also includes features such as comparative omics analysis.

As *RegulonDB*, *BioCyc* has tools that allow for regulatory network hierarchy visualization, aiding the analysis of relationships between genes and regulons. Unlike other regulatory resources, *BioCyc* lacks gene expression datasets. The contents of *BioCyc* can be downloaded as *Systems Biology Markup Language (SBML)* files [69] or other formats. There are also several ways for programmatically accessing the data, such as REST Web Services and Python [70] APIs.

2.4 Methodologies for TRN reconstruction

2.4.1 *Comparative genomics approach: template network-based*

The template network-based methodologies are based on the assumption that orthologous genes share a common DNA ancestral sequence that diverged after speciation [71]. Hence, orthologous transcription factors might regulate orthologous genes. Based on this assumption, template network-based methodologies can use known and curated TRNs of model organisms to search for orthologous TFs on the target genome [72]. Once the TFs and co-regulated genes have been found in the target genome, it is possible to reconstruct a new TRN for the organism under study based on the curated template.

The template network-based methodologies can be especially precise when the organism being studied is phylogenetically close to the reference organism [72]. This implies that this approach is particularly useful when targeting new strains of known prokaryotic organisms. The orthologous TFs search is in fact considerably easy to perform, as there are already available powerful computational tools, such as bidirectional best hits among other procedures [73].

Nevertheless, there are two main disadvantages associated with these methodologies. Firstly, this approach requires a high-quality and well curated reference TRN, from well documented organisms, such as *Escherichia coli*. Consequently, if the organism under study is not very similar phylogenetically to the reference organism, these methods will provide biased or false regulatory interactions.

The second disadvantage is related to the scale of the TRN reconstruction. In short, employing the template network-based methodologies would require the collection of well-curated TRNs at the genome-scale. Even for the best documented organisms, this scale of information is not easily accessible. In fact, it seems that there are no comprehensive databases covering genome-scale TRNs [74]. Most of the available regulatory information can be found on databases or literature containing curated regulatory interactions, which are merely a part of complete genome-scale TRNs.

2.4.2 *Comparative genomics approach: TFBS data-based*

As with the template network-based methodologies, TFBS data-based methods can also be classified as comparative genomics approaches. In general, the first step of

these methodologies consists on selecting a model organism for which data on TFs and TFBSs can be collected automatically. This data is then used to predict TFBSs in the target genome. If a certain TFBS is found, it is assumed that both the TF and set of co-regulated genes have a similar role in the target organism. The lack of TFBS data for model TFs may be overcome by using TFBS data from orthologous genes regulated by the given TFs.

As in the template network-based methodologies, selecting an appropriate reference genome sequence is of the utmost importance. For example, *Bacillus subtilis* is widely used as a reference model for gram-positive bacteria, whereas *Escherichia coli* is usually selected as the model microorganism of gram-negative bacteria [72]. After obtaining a high-quality dataset, a motif pattern recognition algorithm is used for assembling a *Position Weight Matrix (PWM)* which is then used for searching binding sites in the target genome [75]. One of the most commonly used algorithms to generate PWMs is *W-AlignACE* [76].

Previous studies have shown conservation of co-regulated genes by orthologous TFs [77]. Based on this assumption, a final validation can be performed to confirm the TFBS conservation. A search is performed, within a defined threshold, for binding sites upstream of the operons that are regulated by the TF. The TFBS prediction is assumed correct if the binding site is conserved within that threshold. Alternatively, if the predicted binding site is scattered all over the genome sequence, this is considered a false positive. As previously described for the template-network based methodologies, the main disadvantage of this method is requiring a high-quality, curated set of TFs and TFBSs for a model organism which in turn should also be phylogenetically close to the organism under study. A quick summary of the steps involved in this methodology is shown in Figure 2.

Genetic Regulatory Modules (GRAM) was the first well documented algorithm designed for reconstructing TRNs using only binding site data [78]. Likewise, *Regulogger* is another example of a computational tool for TRN reconstruction using binding site data [79]. *Regulogger* regulon-based approach is able to find sets of co-regulated genes for which the regulatory machinery has been conserved across multiple organisms. The web resource of *RegPredict* also enables TRN reconstruction with TFBS data, though this tool is offline at the time of writing [80].

In addition, it is possible to reconstruct TRNs from comparative genomics data using methods such as phylogenetic footprinting [81].

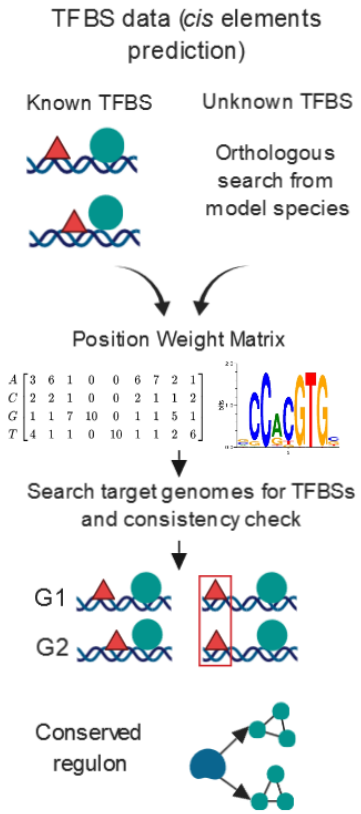


Figure 2.: The different steps involved into the TFBS data-based methodology for TRN reconstruction comprise TF and TFBS data retrieval from the reference organism or phylogenetically close ones, PWM assembly for predicting TFBS on the target genome and consistency check execution (adapted from Faria et al, 2014 [11]).

2.4.3 *Gene expression data approach: de novo reverse engineering*

With the development of high-throughput functional genomics, there has been a significant rise of publicly available gene expression datasets [28]. This data can be used for reconstructing TRNs without the need to use curated regulatory networks or TFBS data. Previous attempts of reconstructing TRNs from gene expression data included various methodologies, such as Bayesian networks, ordinary and partial differential equations, stochastic equations, Boolean networks, qualitative differential equations and rule-based formalisms [82]. These methods led to the development of new methodologies, which can be classified according to three different categories:

- Mathematical modeling approaches [83]
- Module-based or direct inference approaches [84]
- Unsupervised or (semi) supervised approaches [85, 86]

Regarding the mathematical modelling classification, differential equation-based [87], mutual information-based [88] and Bayesian network-based [89] are the most commonly used approaches. Boolean networks [90], neural networks [91], correlation analysis [92] and relevance networks [93] are worth mentioning as well.

Module-based approaches assume that regulatory networks exhibit a modular structure comprised by sets of co-expressed genes [94]. Regulatory modules are commonly inferred using clustering or biclustering algorithms, such as *cMonkey* [95]. On the other hand, direct inference approaches focus on single interactions between targets and their regulators [96]. Previous studies have shown that there is no clear advantage between module-based and direct inference methodologies and results depend on the gene expression data and type of regulators to be predicted [84]. It has been shown that module-based approaches such as *Learning Module Networks (LeMoNe)* [97] provide better results for regulators that interact with many genes, whereas direct inference methods such as *CLR* [96] have better performance for regulators with fewer interactions. Nevertheless, module-based and direct inference approaches must be seen as complementary, as combining these methodologies often leads to improved TRN reconstructions.

TRNs can be reconstructed using either unsupervised or supervised methodologies. Supervised methods, such as *Gene Network Inference with Ensemble of Trees (GENIE3)* [98], require a dataset of previously known regulatory interactions, unlike unsupervised methods such as *Inferelator* [99]. The applicability of either supervised or unsupervised methods depends on the availability of curated regulatory datasets. In fact, if a curated dataset of regulatory interactions is available, supervised methods would eventually yield better results. On the other hand, unsupervised methods are more precise for less documented and studied organisms.

Recently, a new approach has been proposed for validating TRN reconstruction with gene expression data. This approach introduces the concept of *Atomic Regulon (AR)*s, which are sets of co-regulated genes simultaneously underexpressed or overexpressed across a wide range of diverse environmental conditions [33]. Thus, it is assumed that genes in the same AR respond to the same set of stimuli.

Using additional comparative genomics tools, such as *Protein-protein Basic Local Alignment Search Tool (BLASTp)* [100, 101], it might be possible to validate an AR and connect its set of stimuli to TFs. A predicted TRN might incorrectly group a set of co-regulated genes together, in which a particular gene might respond to different stimuli than the others. ARs may provide information for overcoming this inconsistency by grouping sets of co-regulated genes regarding its systematic response to a particular stimulus.

2.5 Integration of regulatory information in *GSM* models

2.5.1 *GSM models: analysis and simulation*

GSM models are *in silico* metabolic representations of an organism, that enable metabolic response analysis and phenotype simulation [102]. These models can be used to predict the phenotypic and metabolic behavior of organisms over multiple environmental and genetic conditions. Assembling metabolic networks is an essential step for reconstructing GSM models. First, a genome wide annotation process is required to find the function of each gene product. Then, using databases, such as *KEGG* [103], one is capable of assigning metabolic reactions to enzymes encoded into the genome of the organism. Furthermore, this data regarding genes, enzymes and metabolic reactions is afterwards compiled into the metabolic network of the organism.

The mathematical representation of a metabolic network describes the stoichiometry of the set of metabolic reactions that occur in a given organism. This information is represented in the model as a stoichiometric matrix S. In this matrix, rows and columns correspond to metabolites and reactions respectively, while the elements stand for the stoichiometry of a given metabolite in the corresponding reaction. The fluxes within the system are defined as a flux vector v.

Different approaches for modelling metabolic networks have been proposed, which require additional varying sets of constraints [104]. Kinetic models predict metabolite concentrations and the fluxes of reactions as a function of time, using ordinary differential equations [105]. However, this approach might be impaired due to insufficient kinetic data, such as the maximal rate of reactions and Michaelis-Menten constant. Kinetic models are usually applied to smaller fractions of metabolic networks, covering few pathways [105].

On the other hand, constraint-based models do not require kinetic information and can be used for genome-scale modelling [106]. These models assume a steady-state within the cell which means that the concentration of the metabolites does not change over time. In detail, the sum of producing and consuming fluxes of each metabolite equals zero. Hence, the system can be seen as a set of linear equations, equal to zero, describing the flux of the metabolites within the cell. Moreover, the steady-state assumption can be represented as a matrix formalism as follows (1):

$$S * v = 0 \quad (1)$$

Where:

- S is the stoichiometric matrix;
- v is the flux vector.

In detail, most metabolic networks can be seen as an under-determined system of linear equations (the number of unknown variables is higher than the number of equations) due to a typically higher number of reactions than metabolites. FBA is a mathematical approach, that can simulate metabolic and phenotypic behavior in GSM models, by applying linear programming [107] to solve the aforementioned under-determined system. The optimal solution to this linear programming problem, requires the maximization or minimization of a certain objective function, such as biomass production or production of a compound of interest. Maximization or minimization of an objective function is subject to stoichiometric constraints. Additional constraints, such as upper and lower bounds of fluxes, which represent reversibility and directionality of reactions, can be added to the equation system as follows (2):

$$\begin{gathered} maximize \rightarrow Z \\ subject\ to \rightarrow S * v = 0 \\ and \rightarrow \alpha_j \leq v_j \leq \beta_j, j = 1, ..., N \end{gathered} \quad (2)$$

Where:

- Z is the linear objective function;
- v is the flux vector;

- S is the stoichiometric matrix;
- α_j and β_j is the lower and upper bound of a flux, respectively.

FBA is commonly used for simulating phenotypic behavior as it relies solely on stoichiometric characteristics and does not require kinetic parameters. On the other hand, this approach does not account for dynamic behavior nor regulatory interactions. It has been shown that by integrating regulatory constraints, the accuracy of the predictions of FBA is improved [7]. Several methodologies have been proposed for overcoming the limitations of FBA, which will be further discussed in chapter 2.5.2.

2.5.2 *Integration of transcriptional regulatory data in GSM models*

Different mathematical formalisms have been proposed for modelling cellular metabolism such as constraint-based models [104]. Information retrieved from the genome annotation and metabolic databases may reveal the stoichiometry of the metabolic network of a cell. Mathematical frameworks such as FBA use the information of constraint-based models for clarifying steady-state flux distributions of a metabolic network. The increasing availability of genome sequences, as well as the efficiency and scalability of this methodology has made constraint-based models a widely used tool for describing phenotypic and metabolic behavior as well as predicting biological capabilities [108]. However, GSM model predictions can be incorrect when regulatory interactions influence metabolic pathways. Hence, integrating transcriptomics data with GSM models may improve the outcome of its predictions [109].

The proposed concept of *Regulatory Flux Balance Analysis (rFBA)* aims at overcoming limitations of the FBA formulation [110]. This methodology integrates transcriptional regulatory information into the FBA methodology, representing regulatory constraints through a Boolean logic formalism. In fact, the integration of regulatory information has proved to be advantageous for quantitative dynamic simulation of substrate uptake, cell growth and by-product secretion. However, this approach can fail to account for complex regulatory interactions as it considers genes to be "off" or "on" disregarding gene expression variance.

Probabilistic Regulation of Metabolism (PROM) is also an example of a methodology for integrating regulatory networks into GSM models [111]. This approach requires a large amount of gene expression data, measured under multiple environmental and genetic conditions, to generate a probabilistic model for the gene regulatory network. Then, the regulatory network is integrated with a constraint-based metabolic model by

changing the flux bounds according to the associated probabilities. This method may allow for direct integration of transcriptomics data into a metabolic model without requiring Boolean rules. However, this approach requires a large amount of experimental gene expression data for accurate data validation.

Transcriptional-controlled Flux Balance Analysis (tFBA) [112] is another approach that uses experimental gene expression data to overcome the rigid binary structure of Boolean formulations, such as the ones used in rFBA. Instead of using "on" and "off" constraints, it relies on flexible "up/down" constraints based on gene expression data. Among different experimental conditions, the level expression of a gene may vary beyond the binary state of "on" and "off", yielding better results than FBA when predicting flux distribution. Regarding the integration of transcriptional regulatory data into GSM models, a universal methodology that systematically outperforms others does not emerge, as certain approaches might yield better results, in specific conditions.

2.6 DATA WAREHOUSING

Building a repository of data retrieved from different sources requires preemptive planning and schematizing. For that, several guidelines, procedures and tools have been created easing such extremely laborious endeavor. Data warehousing is a widely accepted methodology for compiling, processing and merging data from different sources into a unified repository, easing the extraction of relevant information [113].

The contents of a data warehouse must be legible and intuitive both for developers and for users. In turn, data warehousing often resorts to the ETL tools that are capable of extracting data from heterogeneous sources, transform it to appropriate formats and structures, and finally load it into a unified and standardized database. Figure 3 portrays a simplified view of data warehousing and the application of the ETL tools.

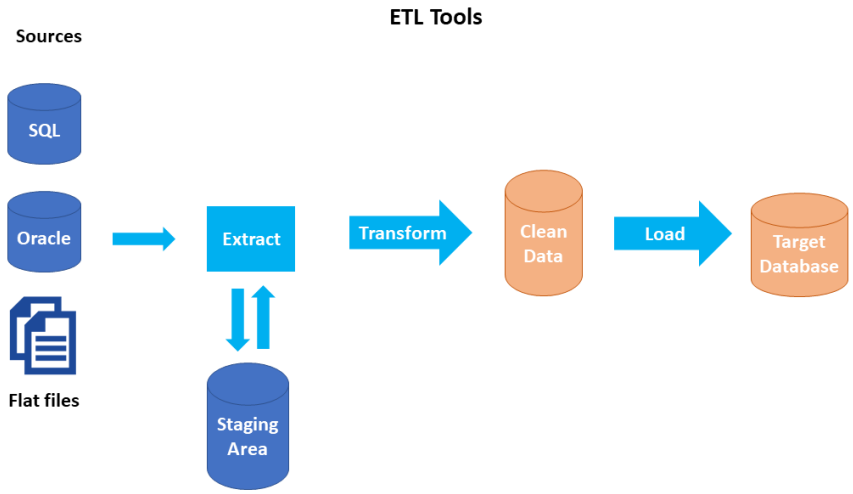


Figure 3.: Data warehousing using the ETL tools; data extraction, transformation and loading.

2.6.1 *Extraction*

The first step of the ETL tools consists in extracting data from external sources such as flat files or databases, without compromising the integrity of the original source. During this task, performance is prioritized over data curation and filtering. Hence, data is transferred directly to the staging area, a parallel database hosting all extracted and unfiltered data. The extraction task of the ETL tools may require simple transformations to convert the extracted data types into suitable ones for the staging area (not to be mistaken with the formal second task of the ETL tools).

2.6.2 *Transformation*

Different data sources often have distinct representations, formats and encodings regarding the same information, which must be reconciled. Several transformation procedures are often used at this task, so that data to be integrated, is consolidated and fits the predefined structure for the unified repository/data warehouse. Thus, there are multiple steps which can be employed to convert extracted data to a general format, with some examples as follows:

- **Cleaning** - standardize certain values (e.g. ”Transcription Factor” to ”TF”);
- **Upper and lower case conversion** - avoid denoting same entities by different names (e.g. ”Zur” to ”zur”);
- **Filtering** - select relevant columns only;
- **Conversion of units of measurement** - numerical conversions, date time conversions and others;
- **Splitting** - split a column in multiple columns;
- **Joining** - merge data from other sources (e.g. merging *UniProt* data with a TRN dataset);
- **Threshold validation** - check that certain values are contained within the established threshold (e.g. gene position in a TRN dataset must be a lower value than the genome size);
- **Mandatory fields validation** - remove table rows which lack data for mandatory columns;
- **Rearranging** - change order of columns or rows;

2.6.3 *Loading*

The final step in the ETL tools, loads the extracted and transformed data to a target database. During this step it is important to verify data integrity, as eventual mistakes committed during this task might lead to complex and overwhelming restoration procedures. Hence, the loading process must guarantee that key fields are not missing and that previously loaded data must not be affected.

2.7 DATABASE MANAGEMENT SYSTEMS

A database can be seen as a manageable structured collection of data. On the other hand, a repository is a set of computational tools (including the database) that can access and manipulate a collection of data. Designing a database involves selecting a database management system. In short, a *Database Management System (DBMS)* is a

computational tool for creating and manipulating databases, in which the data is centralized and organized according to the DBMS predefined structure [114]. Nowadays, *Oracle* [115] and *My Structured Query Language (MySQL)* [116] are the most common DBMSs. One of the main advantages of DBMSs is the possibility of using an API for accessing and manipulating data, regardless of the platform. *Structured Query Language (SQL)* [117] is one of the most common APIs for database management.

DB-Engines [118] is an online platform that ranks DBMSs based on their popularity and usage. Analyzing each popular DBMS is out of the scope of this work, instead it is pertinent to focus on the different database models. Among the top 25 most popular DBMSs at the time of writing, it is possible to identify several database models such as:

- Relational models
- Key-value store models
- Document store models
- Wide column store models
- Graph models

Relational models were proposed in 1969 by Edgar Codd [119] and since then, they have been used by the most popular relational database systems, such as *Oracle* and MySQL. Relational databases can store data into tables which in turn are organized into tuples and columns. Each column belongs to a specific data type and has a certain descriptive name, while the tuples store the actual data and are associated with a unique key. Tables can represent entities, such as genes or metabolic compounds, while the corresponding rows represent an instance of that entity such as "*pkm*" or "pyruvate", respectively.

Relational databases are fast, secure and have been consistently refined throughout the years. Moreover, this type of database often supports SQL which is the most popular domain-specific language for querying in databases. Therefore, besides being widely used, this type of DBMS is also very well-documented. The main disadvantage associated with relational databases is the inefficiency of representing interactions between different entities, namely tables. Tables are connected by foreign keys, which refer to primary key attributes. *JOIN* tables, which demand additional computational power and complex SQL queries, are required to access these connections. Biological

data often required for reconstructing a TRN is based on interactions between different entities. Thus, appropriate DBMSs should be sought to meet these specific criteria.

With the rise of big data and increasing amount of complex information, NoSQL databases [13] have gained increasing attention, as their counterparts lack the flexibility to handle such complex data. The term NoSQL refers to "non-SQL" in the way that NoSQL databases are not relational nor store data in a table structure. Hence, most NoSQL databases do not support the utilization of the popular query domain-specific language, SQL. Nevertheless, there are some exceptions as for instance, non-relational databases capable of using SQL only for querying purposes, which led to the commonly accepted definition of the term NoSQL as "not only SQL" [120].

Key-value store databases can store and manage data using dictionaries [120]. Dictionaries contain a collection of records in which different fields can be available, each containing distinct data. Data is retrieved using the unique key of a record. Key-value databases are more flexible than traditional relational databases having for instance excellent performance when dealing with unrelated data. *Redis* [121], *Aerospike* [120] and *DynamoDB* [122] are some of the most popular key-value databases.

Document-store are NoSQL databases where each record and associated data are stored as a single document. Different document formats are supported, including XML or *JavaScript Object Notation (JSON)* files [123]. One of the main advantages of document-stores is the lack of a predefined schema, unlike relational databases. Different documents of the same information type, such as a user registration file, can contain different attributes without being responsible for database inconsistencies. In a simple example, while filling a user registration document, one can choose to fill its phone number, and another might refuse it, thus the phone number would not be an attribute on the respective document and both files would be registered in the database. In a classical relational database, the phone number would be an attribute for every user registration file, but in some cases, it would not contain a value.

Document stores have many useful applications but are not the most prevalent databases for handling data relationships, due to the nature of storing data in single isolated documents. *MongoDB* [124] and *Couchbase* [125] are popular examples of document-oriented databases.

Wide-column store databases are classified as NoSQL data management systems that share the common core characteristic of relational databases: a table-based structure. The main difference relies on the fact that in wide-column databases, the number and names of the columns may vary from row to row within the same table [13]. Hence, tables regularly employed in this type of databases typically have a large num-

ber of columns. Wide-column stores are commonly used with large-scale web applications, and some of the most popular DBMSs are *Accumulo* [126] and *Cassandra* [127]. As this type of database model relies on a table-based structure, it shares the same critical disadvantage as relational models, providing less efficiency when handling data with many relationships.

Finally, graph-based databases do not support SQL for querying data nor rely in table-based structure. This type of databases represents data through graphs using nodes and edges. In detail, the data items are stored in nodes having as many attributes as required, while the relationships between nodes are defined as edges. Furthermore, the relationships can also hold data and different attributes. In short, a graph database is focused on the relationships present in the data, making this database model an ideal candidate for storing and maintaining biological data such as the one addressed in this work. Tables 5 and 6 summarize the NoSQL database models that have been analyzed so far including the main features, applications and DBMS examples.

Table 5.: Compilation of the most relevant NoSQL database models.

NoSQL database models	**Features**
Key-value store	Uses dictionaries that pair unique keys with its respective records
Document store	Stores data in flexible documents in formats such as *XML* or *JSON*
Wide column store	Stores data in flexible tables with a large number of columns
Graph	Uses graphs to store data in nodes and edges, focusing on the relationships present in the data

Table 6.: Compilation of the most relevant NoSQL database models, examples of its applications and its most popular database management systems.

NoSQL database models	**Possible applications**	**Most popular *DBMS*s**
Key-value store	Application logs and sets of unrelated data	*Redis* [121] and *Aerospike* [120]
Document store	Content management and web monitoring	*MongoDB* [124] and *Couchbase* [125]
Wide column store	Internet search and large-scale web applications	*Accumulo* [126] and *Cassandra* [127]
Graph	Social networks, recommendation systems and biological networks	*Neo4j* [12] and *AllegroGraph* [128]

2.7.1 *Graph databases*

Among several NoSQL database models, graph databases have shown to be the most appropriate for developing repositories of TRNs. The rationale of graph databases is that data processing is as important as the inherited relationships. Thus, the flexibility and robustness of graph databases ease the process of uploading and manipulating data. In contrast, relational databases require that tables, including attributes and data types, should be set in advance before loading the data items.

As discussed before, while analyzing online repositories in chapter 2.2, graphical visualization is important for studying relationships between data. Graph databases allow data filtering and automatic retrieval with a programming querying language. For instance, in biological networks where the volume of highly connected data is considerable, organizing the data into a graph-based structure and using graph theory algorithms for node and edge traversal will yield better performance, while decreasing scalability hurdles of a large biological network.

Graphs are represented by nodes and edges. The latter represent relationships between the former. Nodes are the various data entities, holding an indefinite num-

ber of attributes (which are key-value pairs of data like row-columns in a relational database) and labels that distinguish the different types of nodes. Relationships, on the other hand, are direct connections between nodes and can hold several properties. This allows for a flexible and robust storage of data as well as the respective relationships.

The difference of performance between searching for interconnected data in relational databases or graph databases is staggering. Jonas Partner and coworkers [129] designed a social media database with data for 1 000 000 users and implemented a query for finding the list of friends of each user. For simply finding friends of each user, the graph database *Neo4j* was 60% faster than MySQL. Nevertheless, this example does not even fully explore the potential of a graph database as the depth value of interconnected data is two, meaning that only two nodes are being explored at a time (repetitions of finding one user and its friend). If one considers an extreme example, namely a depth value of four, one would be searching for friends of friends of friends of friends of a certain user. This complex interaction would take in this example 1543.505 seconds with a MySQL query and just 1.359 seconds for a *Neo4j* query (per 1000 users). Hence, a graph database is more than 1000 times faster in this scenario. The higher the depth of interconnectivity, the larger the difference of performance between relational and graph databases.

When compared to relational databases, graph databases are more efficient with highly interconnected data and ease data querying. Considering a simple example of a mock-up database, which stores data about regulators, regulated operons and associated genes, it might be observed that there is a close connection between different data entities. Namely, each regulator controls an operon (which comprises several genes), various operons, or even a single gene (which can be associated with a single-gene operon). In this example, it is necessary to set five different tables when using a relational database. Regulators, operons and genes are stored in three unique tables and connected by two intermediate tables.

In a real case example, finding the set of genes regulated by the TF (regulator) *AraC*, requires joining both the regulator and the operon tables. The primary keys of the regulators and associated operons are foreign keys in the intermediate table that links both entities. Additionally, the primary keys of the operons and associated genes are foreign keys in a final intermediate table that joins both entities. This example of data retrieval requires two join operations which link three different tables (possibly with hundreds of thousands of records).

This process is computationally expensive as it requires complex queries and a rigid schema planning. As shown in Figure 4, this exact same example is simpler and more efficient in a graph database. In fact, there is no need for *JOIN* operations as regulator entities and genes have direct connections, thus making this operation a trivial node traversal problem. These connections are represented by edges which may have several attributes. It is possible to represent operons as the attribute of an edge attribute between regulators and genes, thus dividing regulated genes in different subsets.

Additionally, graph databases have its own query languages. In the case of *Neo4j*, it is possible to query data in the database using *Cypher* [130]. As shown in Figure 5, there is no drawback when using *Cypher* for querying data.

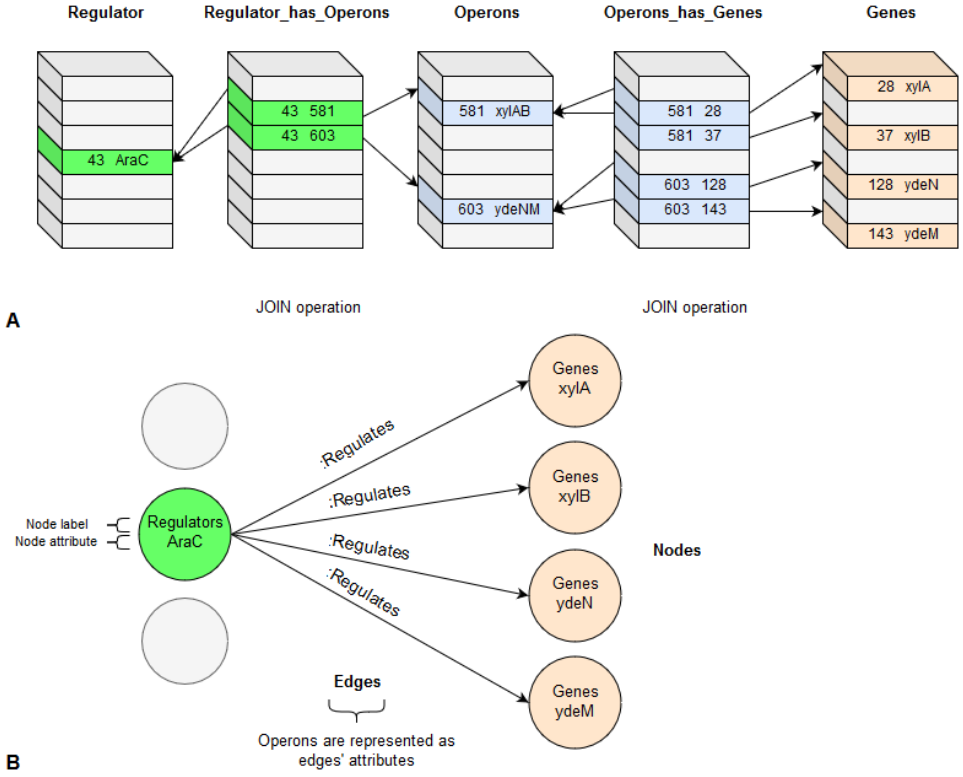


Figure 4.: Example case of data retrieval from a relational database (A) and a graph database (B). It represents a query for finding all genes regulated by the transcription factor *AraC*. The process is computationally expensive in relational databases (A) as the execution of *JOIN* operations is required, while in graph databases (B), the process is simpler and more efficient, due to the fact that data is represented as edges connecting nodes.

A

```sql
select genes_id from regulator inner join regulator_has_operons on
regulator.idregulator = regulator_has_operons.regulator_idregulator inner join
operons_has_genes on regulator_has_operons.operons_idoperons=
operons_has_genes.operons_idoperons where
regulator.regulator_name="AraC";
```

B

```
MATCH (r:regulator)-[:REGULATES]->(g:genes)
WHERE r.regulator_name="AraC"
RETURN g.genes_id
```

Figure 5.: A query statement in two different query languages for finding the same information, which is the set of genes regulated by the transcription factor *AraC*. The first query (A) is written in SQL that is the most popular query language for interacting with relational databases while the second query (B) is written in *Cypher* which is the query language of *Neo4j*.

2.7.2 *Neo4j graph database management system*

According to the *DB-Engine* ranking of DBMSs, *Neo4j* is by far the most mentioned and relevant graph database. *Neo4j* was implemented in *Java* [131] though there is driver API support for several other programming languages such as *C#* [132], *GO* [133], *JavaScript* [134] and *Python*. *Neo4j* relies on *Cypher* which is a declarative graph query language for defining, manipulating and retrieving data. Using the corresponding drivers makes it possible to integrate *Cypher* queries with applications developed with the aforementioned programming languages. Hence, it is trivial to integrate data obtained from custom developed computational tools with a *Neo4j* database.

Furthermore, *Neo4j* also supports graphical visualization of data as shown in Figure 6, which may be very useful for observing and designing the inner structure of the database. This DBMS has a built-in web interface that enables query tests and comprehensive tutorials for both learning *Cypher* commands and good practices for database development. For highly interconnected data, graph databases are the most appropriate DBMSs and *Neo4j* might be the best solution for this model of databases.

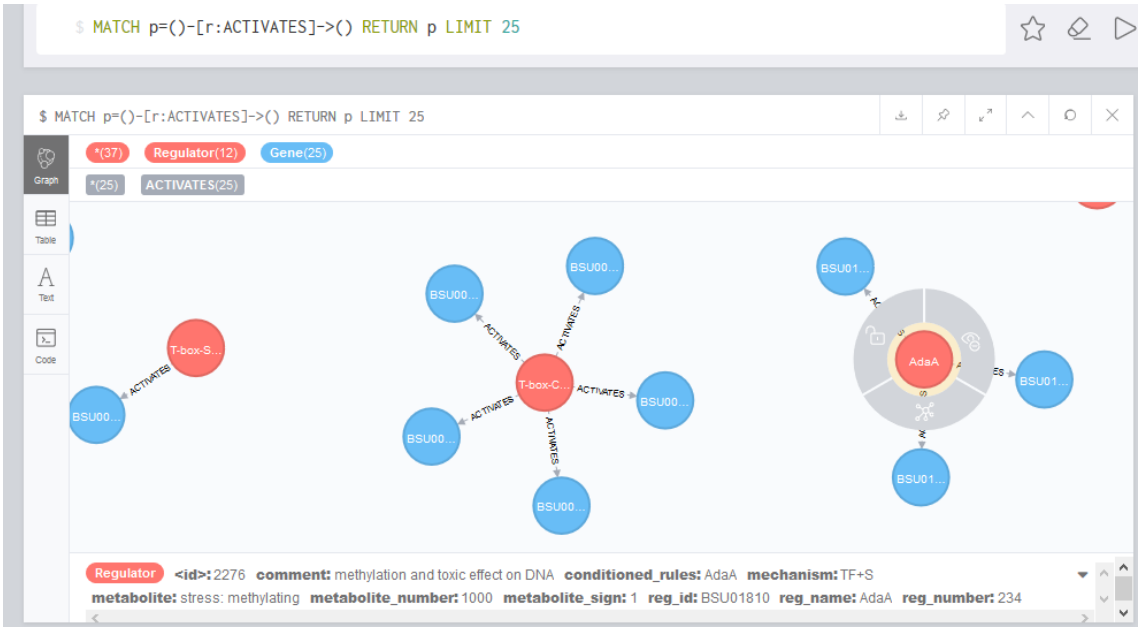


Figure 6.: A simple query showing activation relationships from a prototype regulatory database implemented with the *Neo4j* DBMS. Transcription factors (the regulator entities) are portrayed in red, while the regulated genes are portrayed in blue. The edges connecting nodes represent an activation type of regulation. Each node has a set of attributes that further detail its respective entity.

3

MATERIALS AND METHODS

3.1 REPOSITORY ARCHITECTURE

A system based on the data warehousing process and ETL tools was designed for assembling a TRN repository using *Neo4j*, a graph-based DBMS. The architecture of the repository comprises several subsystems as illustrated in Figure 7:

- Data retrieval subsystem
- Knowledge expansion subsystem
- Integration subsystem
- Request subsystem
- Reporting subsystem

The aforementioned subsystems are tools developed in *Python* responsible for creating or updating the *TRN* database, as well as retrieving and analyzing its contents. These subsystems are available at `https://gitlab.bio.di.uminho.pt/DiogoLima/blueprint_repository.git`.

The database developed in this work is divided in two sections: (1) the *Staging Area*, a compilation of all extracted data converted into graph structures, but without prior data transformations; (2) the *TRN Collection*, an organized collection of filtered, processed and transformed prokaryotic TRNs. As both subunits are available in the same database (although separated by different namespaces), the union of these two subgraphs forms the *Central Data Storage (CDS)*.

Furthermore, a web interface was developed to bridge the gap between users and the CDS. Both the CDS and the web interface are installed in mutually communicating

Docker containers in a server. Thus, any alterations made in the CDS are automatically emulated in the web interface in real time.

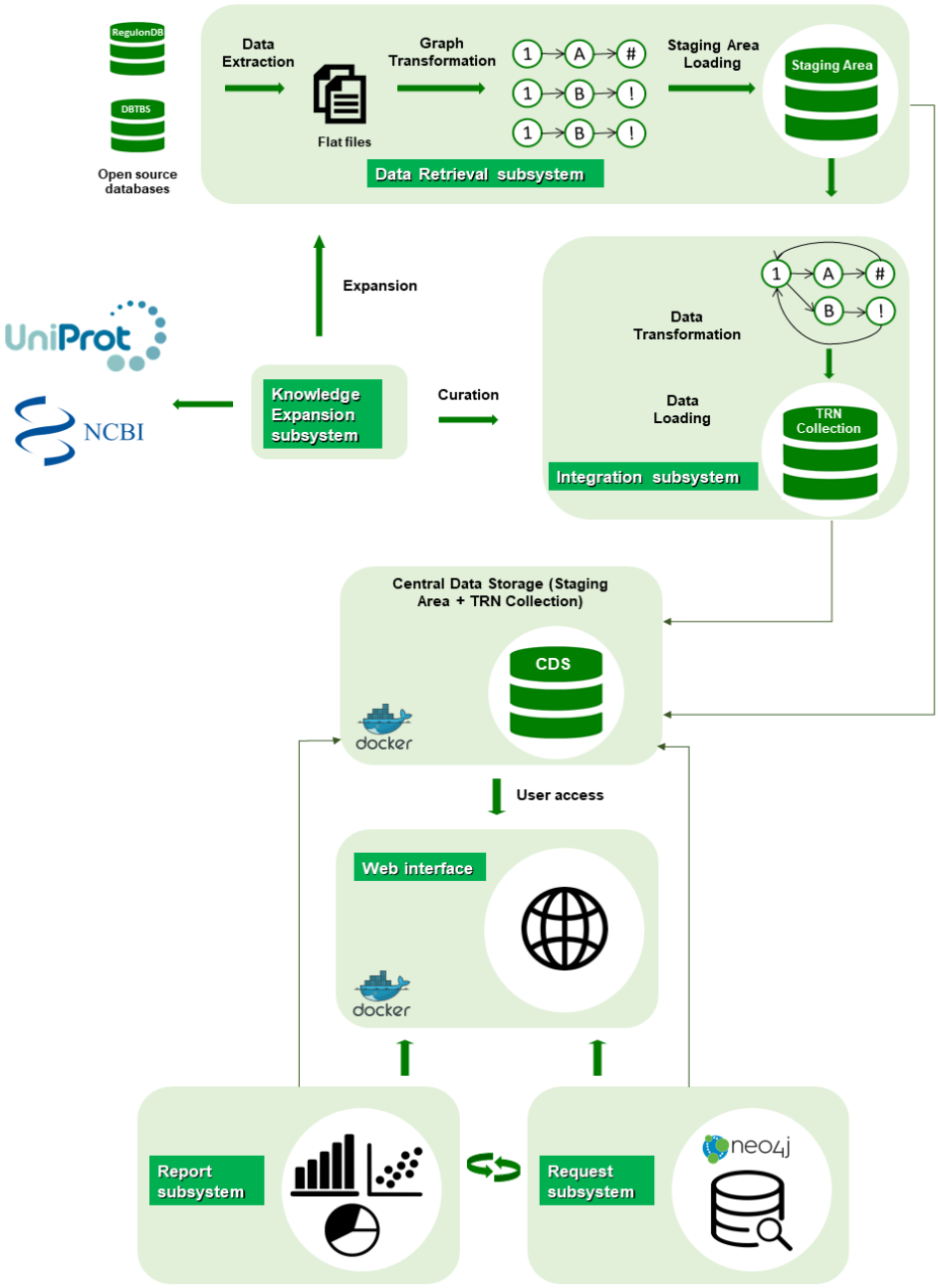


Figure 7.: Architecture of the TRN repository.

3.2 DATA RETRIEVAL SUBSYSTEM

Data extracted from distinct sources was found to be of heterogeneous nature, thus it was necessary to use ETL tools to assemble a *Staging Area* and perform other tasks to ease further data processing. The *Data retrieval* subsystem, based on the *Extraction* task of the ETL tools, can be divided in three different phases:

- Data extraction
- Graph transformation
- Staging Area loading

The extraction phase encompasses retrieving data from different sources, generally in different formats. The transformation and loading phases rely on the amount and quality of the retrieved data. Therefore the extraction task is a core phase in the development of the repository.

3.2.1 *Data extraction*

Multiple sources were surveyed to achieve the goal of assembling a comprehensive database of transcriptional regulatory data. As a result, ten different but complementary sources of regulatory information were collected as shown in Table 7. A different module was developed for each unique source and the methodologies employed for extracting these heterogeneous sources of information will be discussed in the following subsections.

Table 7.: Sources of transcriptional regulatory information collected for the development of the repository.

Source	Database	Published TRN
RegulonDB [23]	✓	-
DBTBS [24]	✓	-
Collectf [26]	✓	-
RegPrecise [27]	✓	-
CoryneRegNet [25]	✓	-
Fang et al. [52]	-	✓
Faria et al. [33]	-	✓
Arrieta-Ortiz et al. [53]	-	✓
Turkarslan et al. [54]	-	✓
Galán-Vásquez et al. [55]	-	✓

3.2.1.1 *RegulonDB*

Although *RegulonDB* hosts downloadable text files with regulatory data of *Escherichia coli* K12, comprising experimentally verified and *in silico* data, only the former was extracted. A tab delimited text file was retrieved from *RegulonDB*, containing data regarding regulatory interactions between TFs and each target *Transcription Unit (TU)*, as well as the binding sites of these regulatory mechanisms. Table 8 summarizes the content of the retrieved text file.

Table 8.: Content of a tab delimited text file retrieved from *RegulonDB*, containing data regarding TF-TU regulatory interactions and the associated binding sites.

Data	**Example**
TF *RegulonDB* identifier	ECK120012595
TF name	Ada
TFBS *RegulonDB* identifier	ECK120012597
TFBS genome left-end position	2147589
TFBS genome right-end position	2147613
TFBS DNA strand	reverse
TF-Gene *RegulonDB* identifier	ECK120031098
Regulated TU	alkA
Regulatory effect	+
Promoter name	alkAp
TFBS center position relative to transcription start site	-41.5
TFBS sequence	ttcagacgctGCGCTTTGCTTTC ATATTCCGGTTgtcgcgacgg
TFBS evidence	[GEA|W|Gene expression analysis]
TFBS evidence confidence level	Strong

Moreover, this dataset was further complemented with two additional tab delimited text files including a document detailing TUs and another document describing target genes. The former provides information for retrieving all gene names associated with

a unique TU and the latter contains additional data for each gene as summarized in Table 9.

Table 9.: Gene data of a tab delimited file retrieved from *RegulonDB*. Examples were selected from several genes.

Data	**Example**
Gene *RegulonDB* identifier	ECK120000988
Gene name and locus tag	thrB (b0003)
Genome left-end position	2801
Genome right-end position	3733
Gene DNA strand	forward
Gene product name	homoserine kinase
Start codon sequence	ATG
Stop codon sequence	TAA
Gene nucleotide sequence	ATGGTTAAAGTTT...

The extraction process for *RegulonDB* and all other sources was performed through the development of parsers in *Python* making use of the *NumPy* and *Pandas* packages. All given examples were transcribed directly from the original data.

3.2.1.2 *DBTBS*

DBTBS is a website that hosts experimentally validated gene regulatory interactions and binding sites of *Bacillus subtilis*. Despite being one of the most important sources of regulatory information on this gram-positive bacterium, this database does not provide downloadable files nor programmatic access. Hence, it was necessary to contact the contributors of the platform, which kindly provided XML and tab-separated files containing the contents of the database. Both files are equivalent content-wise, thus

data was extracted from the tab-separated file. Table 10 summarizes the data extracted from *DBTBS*.

Table 10.: Transcriptional regulatory data available on a tab delimited file retrieved from *DBTBS*. Examples were selected from several TF-gene interactions.

Data	**Example**
Regulator name	AbrB
Factor type	LacI family
Swissprot identifier	Q04385
Subtilist identifier	BG00054
Consensus sequence	TGTCGAA
Comment	Antiterminator protein
Operon	dppABCDE
Regulated Gene	dppA
Gene Strand	+
Sigma Factor	SigA
Regulatory Effect	Negative
TFBS absolute position	1360349..1360382
TFBS location	-16:+18
TFBS sequence	ATGTGATGGGTT AAAAGGATCTCT
References	Fisher SH, et al. 1994

3.2.1.3 *Collectf*

Collectf is a database of regulatory interactions and TFBSs for several bacteria that enables filtering search results by taxonomy (data obtained from NCBI). It was possible to download regulatory data for several *Bacteria* phyla by analysing combined motif reports and using the export data feature of *Collectf*. Data was available in several formats, including TSV files. Furthermore, it was necessary to merge the data of each phyla (individual files) to a single *Comma Separated Values (CSV)* file, containing transcriptional regulatory data for several bacteria (including strain differentiation). Table 11 describes all data extracted from *Collectf*.

Table 11.: Transcriptional regulatory data in the CSV file compiled from the extracted data of *Collectf*.

Data	**Example**
Organism name	*Acidobacterium capsulatum* ATCC 51196
Regulator name	LexA
Regulator *UniProt* accession	C1F978
Genome accession	NC_012483.1
TFBS starting position	284802
TFBS ending position	284820
TFBS strand	1
TFBS sequence	GCGAACAAAAACCGAACAG
Regulatory effect	REP
Experimental evidence	qRT-PCR [RNA]
Regulated genes (locus tags)	ACP_0231

3.2.1.4 *RegPrecise*

RegPrecise is a major resource in prokaryotic regulatory information. Although it is possible to access and browse *RegPrecise* through its website, the platform does not provide downloadable files nor programmatic access via API. Hence, a simple web-scraping solution was implemented using the native *Requests Python* package, to access *RegPrecise* via *Hyper Text Transfer Protocol (HTTP)* requests.

Every organism described in *RegPrecise* is associated with an internal identifier, which is always present in a static *Uniform Resource Locator (URL)*. The proposed web-scraping solution creates a flat text file with TF and TFBS information for each organism based on the corresponding *RegPrecise* entry. However, this solution does not retrieve data regarding the regulatory effect of the TF over its targets. Therefore, the web-scraping algorithm was used to retrieve additional data for each regulator from a different set of web pages in *RegPrecise* containing the same URL pattern.

Contrariwise, the contents of regulator web pages in *RegPrecise* are stored in XML format. Hence, an additional parser for obtaining relevant regulatory data on the effectors of the regulators as well as involved biological processes was developed. All generated text files were merged into a single CSV file with regulatory data for all organisms described in *Regprecise*. Additionally, all data extracted in XML file format was processed and merged into a single JSON file. Figure 8 describes the structure of the extracted data for both scenarios: text files with transcriptional regulatory data for each organism; XML files with additional data for each regulator.

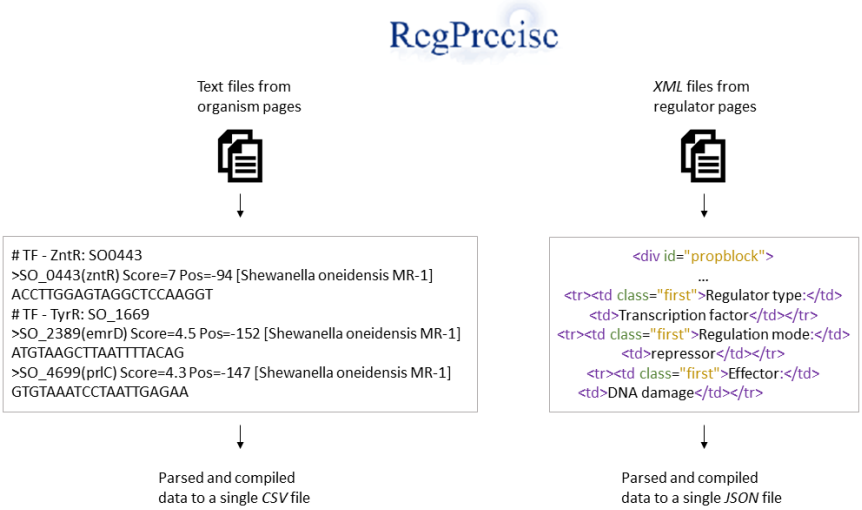


Figure 8.: Structure of the extracted files from *RegPrecise*. Text files retrieved from the organism pages of *RegPrecise* are simple to parse as data is strictly organized; regulators are identified with a "#" symbol, regulated genes and respective data are identified with a ">" symbol. Finally, binding site sequences are represented with the "ATGC" nucleotide alphabet lacking an initiator character in its line. XML files retrieved from the regulator pages of *RegPrecise* consist of scraped *Hyper Text Markup Language (HTML)* [135] code, being complicated to parse and requiring the use of regular expressions or dedicated HTML parsers.

Tables 12 and 13 summarize the data extracted from *RegPrecise* organism and regulator pages, respectively.

Table 12.: Transcriptional regulatory data available in the CSV file compiled from the extracted data of *RegPrecise*.

Data	**Example**
Organism name	*Shewanella oneidensis* MR-1
Regulator name and locus tag	TyrR: SO_1669
Gene locus tag and name	SO_2389(emrD)

(To be continued)

Data	Example
Position relative to transcription start site	-152
TFBS sequence	ATGTAAGCTTAATTTTACAG

Table 13.: Regulator data available in the JSON file compiled from the extracted data of *RegPrecise*.

Data	Example
Organism name	*Ralstonia pickettii* 12J
Regulator name	PsrA
Regulator type	Transcription factor
Regulatory effect	repressor
Biological process	Fatty acid degradation
Effector	Oleate

3.2.1.5 *CoryneRegNet*

CoryneRegNet is a compendium of experimentally validated and computationally predicted regulatory data for *Corynebacteria* with downloadable flat text files. It was possible to retrieve experimentally validated data for four *Corynebacterium* species, namely *Corynebacterium diphteriae*, *Corynebacterium efficiens*, *Corynebacterium glutamicum* and *Corynebacterium jeikeium*. Extracted data is homogeneous across the four species as summarized in Table 14.

Table 14.: Transcriptional regulatory data obtained from *CoryneRegNet* in flat text files.

Data	**Example**
Organism name	*Corynebacterium diphteriae*
Regulator name and locus tag	mntR DIP0619
Regulator module	Macroelement and Metal Homeostasis
Regulator type	Diphteria toxin repressor
Regulator Mutant	disruption
Autoregulator	+
Regulatory Effect	R
Regulated gene name and locus tag	mntA DIP0615
Regulated gene module	Macroelement and Metal Homeostasis
Evidence	experimental
Pubmed Identifiers	12446639
TFBS sequence	aaaggtgttaccgtccttg

3.2.1.6 *Fang et al.*

The study performed by *Fang et al.* [52] (discussed in chapter 2.2.11) provides a complete TRN for *Escherichia coli* K12. The data was obtained in *Excel Microsoft Office Open XML Format Spreadsheet File (XLSX)* format, thus the conversion into a *Pandas* dataframe was straightforward. Table 15 summarizes the contents of the extracted data.

Table 15.: Transcriptional regulatory data obtained from a XLSX file in the supplementary material of *Fang et al* [52].

Data	Example
TF name	nhaR
TF locus tag	b0020
Regulated gene name	pgaD
Regulated gene locus tag	b1021
Regulatory effect	+
Evidence	[BPP, GEA, HIBSCS]
Evidence level	2

3.2.1.7 *Faria et al.*

The work performed by *Faria et al.* [33] (discussed in chapter 2.2.11) provides a complete TRN for *Bacillus subtilis* 168. This TRN can be obtained from the supplementary material in XLSX format [33]. Information retrieved from the complete TRN of *Bacillus subtilis* 168 is summarized in Table 16.

Table 16.: Transcriptional regulatory data available in the XLSX file retrieved from *Faria et al* [33].

Data	Example
Regulated gene name and locus tag	abnA BSU28810
Operon	abnA
Sigma factor	sigA

(To be continued)

Data	Example
Regulator name	CcpA|AraR
Regulatory effect	-1|-1
Involved metabolite(s)	D-fructose-1,6-bisphosphate /Glucose-6-Phosphate|arabinose
Metabolite effect	1/1|-1
Conditioned rules	CcpA+Hpr15P|AraR
Annotation	arabinan-endo 1,5-alpha-L-arabinase

3.2.1.8 *Arrieta-Ortiz et al.*

The study performed by *Arrieta-Ortiz et al.* [53] (discussed in chapter 2.2.11) further complements our repository with transcriptional regulatory data for *Bacillus subtilis* 168. The TRN for the gram-positive bacterium was obtained in XLSX format from the supplementary material of the referred study [53]. Table 17 summarizes the contents of the extracted data.

Table 17.: Transcriptional regulatory data on a XLSX file retrieved from the work of *Arrieta-Ortiz et al* [53].

Data	Example
Regulator name and locus tag	abh BSU14480
Regulated gene name and locus tag	ybfQ BSU02330
β score	0,175
Prior interaction	0

3.2.1.9 *Turkarslan et al.*

The work performed by *Turkarslan et al.* [54] (discussed in chapter 2.2.11) provides a complete TRN for *Mycobacterium tuberculosis* as well as gene expression data. The network was obtained in CSV format from the supplementary material [54]. Based on the expression values of each TF, the dataset was filtered to only contain regulators which have shown differential expression. Table 18 describes the contents of the extracted data.

Table 18.: Transcriptional regulatory data in the CSV file retrieved from *Turkarslan et al.* [54].

Data	**Example**
TF locus tag	Rv0022c
Regulated gene locus tag	Rv1469
Chip center	1656844
Distance	-80
Regulated gene strand	+
Genomic position	1656924
Expression	-1.82
Expression p-value	1.68646e-12
Regulatory effect	REP

3.2.1.10 *Galán-Vásquez et al.*

The study performed by *Galán-Vásquez et al.* [55] (discussed in chapter 2.2.11) provides a complete TRN for several *Pseudomonas aeruginosa* strains. The data was retrieved in XLSX format from the supplementary material. Furthermore, it was possible to acquire data for four *Pseudomonas aeruginosa* strains, namely *Pseudomonas aeruginosa* PA14, *Pseudomonas aeruginosa* PA103, *Pseudomonas aeruginosa* PAK and the most widely

studied *Pseudomonas aeruginosa* PAO1. The acquired data follows the same structure across the four strains as summarized in Table 19.

Table 19.: Transcriptional regulatory data available in the XLSX file retrieved from the publication of *Galán-Vásquez et al.* [55].

Data	**Example**
Organism name	*P. aeruginosa* PAO1
Regulator name	anr
Operon	cox
Regulated gene name	coxA
Regulatory effect	-
Experimental evidence	Microarray and Gene Chip data analysis
PubMed reference	19930444

3.2.2 *Graph transformation*

As previously shown in Figure 7, the repository developed within the scope of this work stores all data in the CDS, which is composed by both the *Staging Area* and *TRN Collection* (Definition 1). The *Staging Area* can be thought of as the namespace for all raw data extracted in sections 3.2.1.1 to 3.2.1.10. The lack of processing implies that the *Staging Area* may contain duplicate, incomplete or erroneous data. However, this subunit allows further processing and integration of multiple sources of data efficiently without the need to manipulate additional files, easing subsequent data manipulations, such as data filtering and transformation, directly on the repository.

Definition 1. *Central Data Storage*

Let G_{sa} be the Staging Area subgraph, G_{tc} the TRN Collection subgraph, the Central Data Storage graph G is defined as $G = G_{sa} \cup G_{tc}$

By separating the *Staging Area* and the *TRN Collection*, it becomes clear that two different phases of data transformation are required: a first phase for transforming

raw data into graph structures and a second phase for processing and manipulating data. Therefore, the first phase of data transformation aims to convert raw data in nodes, edges and properties of a graph. This was performed simultaneously with the *Staging Area loading* phase (discussed in section 3.2.3), using the *Neo4j Python Driver* to integrate *Python* code with *Cypher* queries for generating graph structures. All extracted data was associated to a given entity as shown in Figure 9.

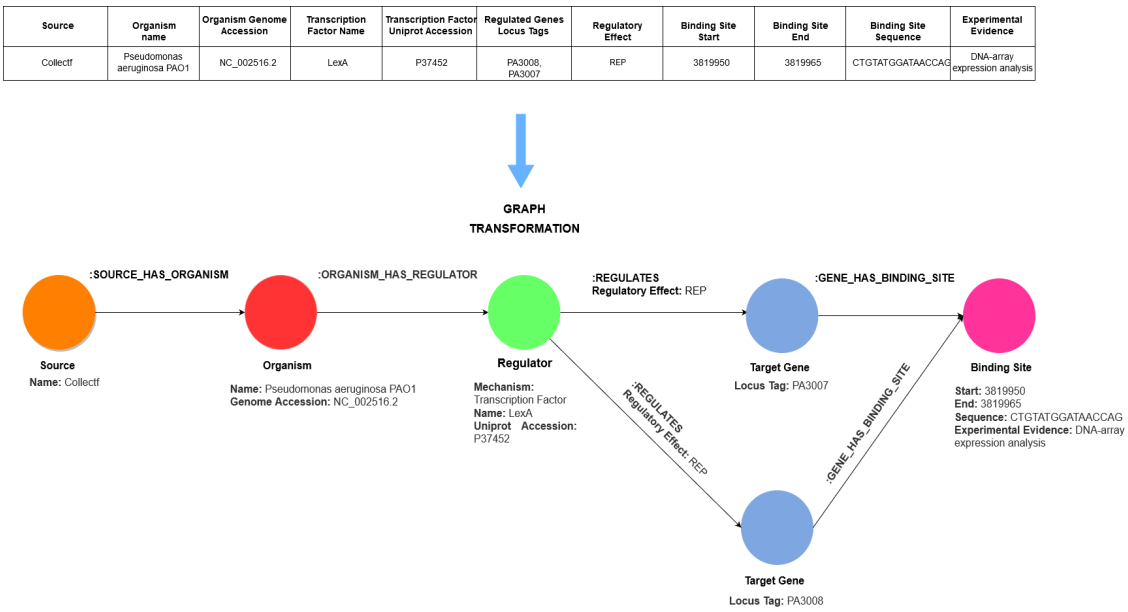


Figure 9.: *Graph transformation* of an entry of the *Collectf* extracted data in the *Staging Area* (adapted for illustration purposes). The extracted data is organized into node labels, namely *Source*, *Organism*, *Regulator*, *Target Gene* and *Binding Site*, node properties, such as *Experimental Evidence* and edges which may or not contain properties.

The resulting *Staging Area* is in fact a conversion of the contents of each file extracted from external sources (Definition 2) to a set of directed acyclic graphs (Definition 3).

Definition 2. *Staging Area*

Let G_σ be the subgraph of data extracted from a given source σ, the Staging Area graph G_{sa} is defined as $G_{sa} = \{G_{\sigma 1}, G_{\sigma 2}, ..., G_{\sigma_n}\}$

Definition 3. *Directed acyclic graphs*

$G_\sigma = (\mathcal{N}_\sigma, \mathcal{E}_\sigma)$ and $G_\sigma \subset G_{sa}$; such that G_σ is the subgraph of data extracted from a given source σ and a subset of the Staging Area graph G_{sa}. $\mathcal{N}_\sigma$ is the complete set of nodes of G_σ and $\mathcal{E}_\sigma$ is the complete set of edges of G_σ that are oriented forming a given path from an origin node to a compulsorily different destination node.

3.2.3 *Staging Area loading*

The *Staging Area loading* phase and the *Graph transformation* phase operations were implemented in the same *Python* modules. The pipeline for loading data is iterative; each *Python* module reads the extracted files, parses the contents, transforms and loads a TF-gene-TFBS interaction at a time.

The loading procedure is generic for any source. However, it was necessary to develop separate projects to process and parse the heterogeneous data retrieved from multiple sources. All nodes in the database are organized by label, thus a given node belongs to one and only one label (Definition 4). The same principle is valid for edges connecting nodes (Definition 5).

Definition 4. *Node Labels*

Any given node is tagged with one and only one label λ_ν, from a label set $\Lambda_\nu = \{$"SA Source", "SA Organism", "Organism", "SA Regulator", "Regulator", "SA Gene", "Gene", "SA Binding Site", "Binding Site"$\}$, in which each label is associated with the data type.

Definition 5. *Edge Labels*

Any given edge is tagged with one and only one label λ_ϵ, from a label set $\Lambda_\epsilon = \{$"SA Source has Organism", "SA Organism has Regulator", "Regulator in Organism", "SA Regulator binds to Binding Site", "Regulator binds to Binding Site", "Regulator forms Protein Complex", "SA Binding Site in Organism", "Binding Site in Organism", "SA Gene in Organism", "Gene in Organism", "SA Gene has Binding Site", "Gene has Binding Site", "SA Regulator regulates Gene", "Regulator regulates Gene"$\}$, in which each label is associated with the relationship between node entities.

It is worth noting that the labels created for the *Staging Area* are separated from the labels associated with the *TRN Collection*. This eases data manipulation while maintaining the two modules apart. Although, labels created for the *Staging Area* and the *TRN Collection* do not overlap with each other, these label abstractions might have the same meaning in both (Example 1).

Example 1. *Staging Area and TRN Collection label demonstration*

Any given node or edge belonging to the Staging Area has a label λ from a label set Λ. However the label value is preceded by a "SA" identifier:

λ_1 = *"Regulator"*, $\lambda_1 \in \Lambda_v \subset \Lambda$;

λ_2 = *"SA Regulator"*, $\lambda_2 \in \Lambda_v \subset \Lambda$;

Both λ_1 and λ_2 represent a label associated with nodes containing regulator information. Nevertheless, the former corresponds to processed regulators in the TRN Collection, while the latter corresponds to raw regulator data in the Staging Area.

In addition, instances of nodes associated with a given label λ will be referred to as objects (Definition 6).

Definition 6. *Objects*

An object is a tuple $O = \{O_{id}, O_{\lambda}, O_{att}\}$, such that O_{id} is an internal identifier of a node, $O_{\lambda} \in \Lambda_v$ and O_{att} is a set of attributes of O corresponding to the properties of a node.

3.3 KNOWLEDGE EXPANSION SUBSYSTEM

The development of the knowledge expansion subsystem aims to complement the data extracted from external sources. The analysis of the extracted data revealed that a given organism may have different nomenclature or internal identifiers across multiple sources, thus generating data inconsistencies. This issue was recurrent with other data entities, namely genes and regulators. Furthermore, it was concluded that it was necessary to standardize data to enable comparisons of information from a variety of sources.

A solution for standardizing organism nomenclature was implemented using the *NCBI Taxonomy* database. Therefore, the taxonomy identifier of each organism was retrieved and loaded into the CDS, both into the *Staging Area* and the *TRN Collection*. The proposed algorithm uses the *Biopython* and *Entrez* packages for remotely accessing the *NCBI Taxonomy* database and obtaining information regarding the organisms obtained in the *Data retrieval* subsystem.

Certain organism names, retrieved from external sources, were mistyped or reflected the common name of the species, thus returning empty results from *NCBI Taxonomy*. Such cases were manually curated to maintain data consistency.

Information, concerning the regulator and corresponding gene, retrieved from different sources was standardized. A solution was implemented by querying UniProt and retrieving data using the *Bioservices* package, namely the UniProt module. Obtain-

ing the UniProt accession of each regulator and gene, allowed avoiding duplicates and merging data from different sources regarding the same entities. Furthermore, it was not necessary to rely on regulator or gene names and locus tags exclusively, which are case-sensitive and sometimes ambiguous. Finally, protein sequences were retrieved from UniProt for every regulator or gene entity, using the aforementioned solution.

Additional data obtained for genes and regulators during this process was stored in the *Staging Area* as specific objects, and later loaded into the *TRN Collection* as properties of either nodes and edges. As *Gene* and *Regulator* objects have unique UniProt accessions and sequences, it was not necessary to create additional objects to accommodate this information.

3.4 INTEGRATION SUBSYSTEM

The purpose of the code developed within the *Integration* subsystem is to clean and process data from the *Staging Area* and load the filtered data into the *TRN Collection*. This module encompasses two different tasks, namely the *Transformation* and *Loading*. It is worth noting that the *Integration* subsystem is also based in the *ETL* tools. However, the *Extraction* task was performed in the *Data retrieval* subsystem.

3.4.1 *Data transformation*

During this task, the disconnected data available in the *Staging Area* was transformed according to the *TRN Collection* structure. The following subsections describe several tasks performed during the *Data transformation* operation, to assemble a database of standardized TRNs.

3.4.1.1 *Data Source*

The *Staging Area* has a defined *SA Source* object that identifies the source of loaded regulatory data. Nevertheless, this is not the case in the *TRN Collection*, as it is possible to represent this information as a relationship (Definition 7) between regulatory entities, such as regulators or binding sites and the respective organism. Hence, the source of a given regulatory interaction was transformed into the property of an edge between such regulatory entities and the associated organism (Example 2), easing the process of filtering the database contents by source.

Definition 7. *Relationships*

A relationship is an edge connecting two Objects such that, $R = O_a - [R_{id}, R_\lambda, \{R_{att}\}] - > O_b$*,* $R \in \mathcal{E}$ *and* O_a*,* $O_b \in \mathcal{N}$ *where* R_{id} *is an internal identifier of an edge,* $R_\lambda \in \Lambda_e$*,* R_{att} *is a set of attributes of R corresponding to its properties and* O_a *is the tail of the edge whereas* O_b *is the head.*

Example 2. *Data source*

Given an Organism object:

$O_a = \{3,$ *Organism,* $\{$*Taxonomy ID = "511145", Name = "Escherichia coli K12 MG1655"*$\}\}$;

and two associated Regulator objects with several attributes:

$O_b = \{245,$ *Regulator,* $\{$*Name = "zur", ...,* $O_{b_{att}}\}\}$;

$O_c = \{246,$ *Regulator,* $\{$*Name = "agar", ...,* $O_{c_{att}}\}\}$;

where the former was retrieved from RegulonDB and the latter from Collectf, data can be organized as a relationship between objects as follows:

$R_x = O_b - [28,$ *Regulator in Organism* $\{$*Source: "RegulonDB"*$\}] - > O_a$;

$R_y = O_c - [29,$ *Regulator in Organism* $\{$*Source: "Collectf"*$\}] - > O_a$

3.4.1.2 *Organism uniqueness*

The addition of the organisms Taxonomy ID to the repository was crucial for integrating data in the *TRN Collection*, as well as to avoid duplicates and to merge data from different sources regarding the same organism (Definition 8 and Example 3).

Definition 8. *Organism uniqueness*

O_a*,* $O_b \in \mathcal{N}$*, such that,* $O_a = \{O_{id_a}, \lambda, \{O_{tax_a}, O_{att_a}\}\}$*,* $O_b = \{O_{id_b}, \lambda, \{O_{tax_b}, O_{att_b}\}\}$ *and* $\lambda =$ *"Organism", then* $O_a = O_b$ *if and only if,* $O_{tax_a} = O_{tax_b}$*, where* O_{tax} *is the Taxonomy ID of the Organism object and* O_{att} *is the remainder set of attributes.*

Example 3. *Matching organisms*

Given two Organism objects retrieved from different external sources:

$O_a = \{1,$ *Organism,* $\{$*Taxonomy ID = "210", Name = "Helicobacter pylori"*$\}\}$;

$O_b = \{2,$ *Organism,* $\{$*Taxonomy ID = "210", Name = "Campylobacter pylori"*$\}\}$;

Despite having different names, both objects are considered to represent the same organism due to a common taxonomy identifier. As such, it is possible to merge these two objects into a single Organism object.

3.4.1.3 *Regulator uniqueness*

A regulator entity, such as a TF, was considered to be the same across different organisms when it shared a significant proportion of the gene sequence as well as the same regulatory function in the biochemical machinery of such organisms (Example 4), thereby usually taking the same common gene name. Based on this rationale, BLASTp searches were performed using common bacterial TFs to ascertain that regulators belonging to different organisms, but sharing a common name have extremely high similarity, thus sharing a similar protein sequence and biological function. The results of the BLASTp searches are available in Table S1 of the supplementary material.

Example 4. *Common regulators across different organisms*

Given two TFs and corresponding UniProt Accessions, namely "P0A9A9" and "Q03456", it is verifiable by accessing UniProt, that both regulators share a common name ("fur") and are involved in ferric uptake regulation. The former TF is present in Escherichia coli *K12 and the latter belongs to* Pseudomonas aeruginosa *ATCC 15692 (and equivalent strains).*

To avoid repeating data in the *TRN Collection*, *Regulator* objects sharing a common name or UniProt accession were grouped into a hub object without compromising data integrity. A hub *Regulator* object may have multiple relationships, with multiple edges connecting this hub to several *Organism* objects. These edges contain the properties that differentiate the shared *Regulator* object among different organisms, including the respective UniProt accession and protein sequence as seen in Figure 10.

Merging *Regulator* objects sharing a common name into a unique hub object eases the search for shared regulatory mechanisms across distinct organisms.

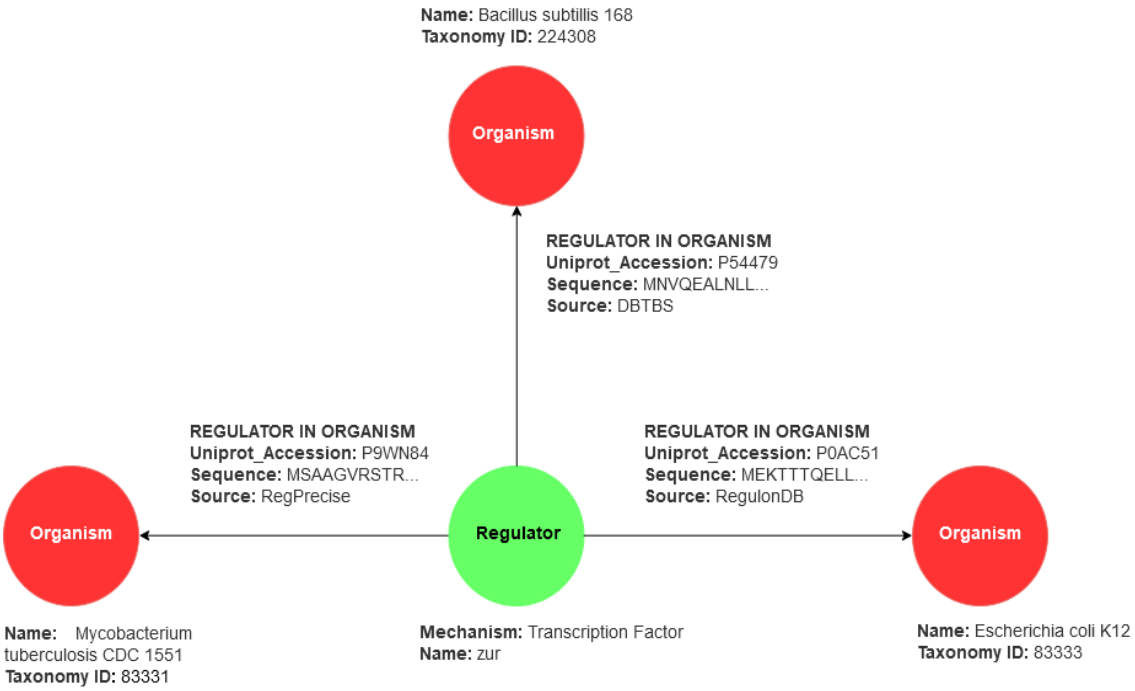


Figure 10.: An example of a hub *Regulator* object in the *TRN Collection*. The TF "zur" is present in three different organisms, each having a unique UniProt accession and protein sequence.

3.4.1.4 *Target Gene uniqueness*

A given gene name may be available in the genome of several organisms as long as these share significant gene sequence similarity. Hence, genes with the same name in different organisms, are likely associated with the same biological function. On the other hand, duplicated *Gene* objects should not be present within an organism TRN. Thus, two genes cannot share the same UniProt accession. According to these rules, the UniProt accession property was used to merge repeated gene data into unique *Gene* objects (Definition 9).

Definition 9. *Gene uniqueness*

$O_a = \{O_{id_a}, \lambda_a, \{O_{UniProt_a}, O_{att_a}\}\}$, $O_b = \{O_{id_b}, \lambda_b, \{O_{UniProt_b}, O_{att_b}\}\}$ and $\lambda_a = \lambda_b =$ "Gene", such that $O_a, O_b \in \mathcal{N}$, then $O_a = O_b$ if and only if, $O_{UniProt_a} = O_{UniProt_b}$, where $O_{UniProt}$ is the UniProt accession of the Gene object and O_{att} is the remainder set of attributes.

To process extracted genes that lacked UniProt data, the uniqueness verification was performed using the gene name or locus tag when available.

3.4.1.5 *Binding Site uniqueness*

Throughout the development of the *Integration* subsystem it was found that binding site data retrieved from distinct sources regarding the same entities may overlap. Thus, a solution was proposed to merge repeated binding sites. Binding site data is considered duplicated, when structural information, such as the binding site nucleotide sequence, starting and ending position in the genome, and respective DNA strand is repeated. However, this is only valid if the duplicated binding site data is associated with the same *Organism* object (Example 5).

Example 5. *Overlapping binding sites*

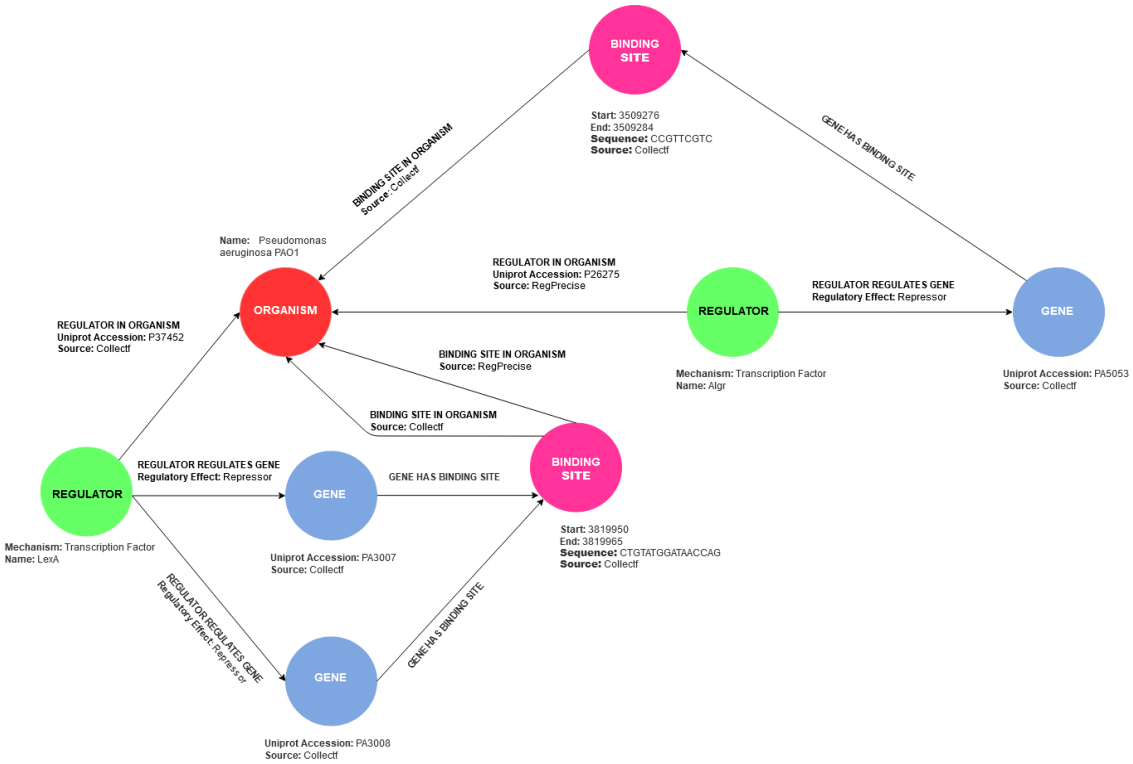
Given three Binding Site objects connected to the respective Organism objects:
$O_a = \{58,$ *Binding Site,* $\{$*Start = "93450", End = "93470", Strand = "forward" , Sequence = "ATCGA..."*$\}\}$ $-[50,$ *Binding Site in Organism,* $\{...\}]-> O_x = \{1,$ *Organism,* $\{...\}\}$ *;*
$O_b = \{63,$ *Binding Site,* $\{$*Start = "93450", End = "93470", Strand = "forward" , Sequence = "ATCGA..."*$\}\}$ $-[51,$ *Binding Site in Organism,* $\{...\}]-> O_x = \{1,$ *Organism,* $\{...\}\}$ *;*
$O_c = \{72,$ *Binding Site,* $\{$*Start = "93450", End = "93470", Strand = "forward" , Sequence = "ATCGA..."*$\}\}$ $-[52,$ *Binding Site in Organism,* $\{...\}]-> O_y = \{2,$ *Organism,* $\{...\}\}$ *;*

Then, $O_a = O_b$ *and* $O_a \neq O_c \neq O_b$ *; two Binding Site objects are deemed as duplicated, when its structural data is the same and these objects are associated to the same Organism object.*

To merge binding site data in the *TRN Collection*, overlapping entries from different sources are represented as additional relationships. These relationships connect a given *Binding Site* object to the associated *Organism* object, portraying the distinct sources as a property of the edge, as shown in Figure 11. Nonetheless, duplicated overlapping entries from the same source are not loaded.

Organism name	Source	Transcription Factor Name	Transcription Factor Uniprot Accession	Regulated Genes Uniprot Accession	Regulatory Effect	Binding Site Start	Binding Site End	Binding Site Sequence
Pseudomonas aeruginosa PAO1	Collectf	LexA	P37452	PA3008	Repressor	3819950	3819965	CTGTATGGATAACCAG
Pseudomonas aeruginosa PAO1	Collectf	LexA	P37452	PA3007	Repressor	3819950	3819965	CTGTATGGATAACCAG
Pseudomonas aeruginosa PAO1	Collectf	AlgR	P26275	PA5053	Repressor	3509276	3509284	CCGTTCGTC
Pseudomonas aeruginosa PAO1	RegPrecise	LexA	P37452	PA3008	Repressor	3819950	3819965	CTGTATGGATAACCAG
Pseudomonas aeruginosa PAO1	RegPrecise	LexA	P37452	PA3008	Repressor	3819950	3819965	CTGTATGGATAACCAG

(a) Overlapping data from distinct sources



(b) Handling overlapping binding site data in the *TRN Collection*

Figure 11.: Compilation of data from different sources with overlapping entities (a) and its integration in the *TRN Collection* (b). Duplicated binding sites overlapping in different sources are integrated once in the *TRN Collection*, and its sources are portrayed as properties of edges connecting the binding sites to the associated organisms. Duplicated binding site data from the same sources is filtered out in the *TRN Collection*.

3.4.1.6 *UniProt curation*

As discussed in chapter 3.3, a solution was implemented to expand extracted data by accessing UniProt and retrieving information, such as the protein sequence of TFs. The utility of this process is not exclusive to increase the amount of information over each TF, as it was necessary to curate data assembled in the *Staging Area*. For instance, it was found during the extraction phase that some regulatory databases provide external data, such as UniProt accessions, which contradict their own internal data. Certain *Regulator* or *Gene* objects may have UniProt accessions assigned to the wrong organism or protein, thus providing erroneous data (Example 6).

Example 6. *Fixing consistency errors with UniProt*

Collectf is one of the multiple extracted sources and it provides information regarding the prokaryotic organism Helicobacter pylori *26695. In addition, the Collectf entry for this microorganism, further details a TF named "Fur" with a UniProt accession of "Q1CU85". Upon further inspection at UniProt, it is possible to verify that this accession matches a protein from a different strain,* Helicobacter pylori *HPAG1. There is no evidence at UniProt, nor at NCBI Taxonomy proving that both strains are equivalent. In conclusion, this Collectf entry failed the proposed UniProt curation filter and was not integrated in the TRN Collection.*

A solution based on the Taxonomy ID of each organism and provided regulatory data, such as locus tag or UniProt accession numbers of genes and regulators, was implemented to curate and merge data in the *TRN Collection* in the absence of consistency errors. This was possible by curating the results of UniProt queries with web scraping tools using the *UniProt* module from the *Bioservices* package.

3.4.1.7 *Other operations*

After merging duplicated *Organism* objects by Taxonomy ID, as discussed in chapter 3.3, a reverse search was performed to retrieve the scientific names of each organism in the *NCBI Taxonomy* database This data was then used to update the name of all organisms in the *TRN Collection* improving consistency.

Overlapping regulatory data, namely regulatory interactions between regulator-gene pairs, was found across all data sources. To include all overlapping data, these duplicated or contradictory regulatory interactions (with different sources or data properties) were stored as multiple relationships between a given regulator-gene pair (Example 7).

Example 7. *Contradictory regulatory interactions*

Given an Organism object:

$O_a = \{5, Organism, \{Taxonomy\ ID = \text{“}224308\text{”}, Name = \text{”}Bacillus\ subtilis\ 168\text{”}\}\}$;

and the Regulator and Gene objects:

$O_b = \{245, Regulator, \{Name = \text{“}abrb\text{”}, ...\}\}$;

$O_c = \{1025, Gene, \{Name = \text{“}aprE\text{”}, ...\}\}$;

it was found that this regulator-gene pair has contradictory data in DBTBS and RegPrecise. The former provides evidence that the ”abrb” regulator represses the ”aprE” gene, however the latter presents evidence for an activation regulatory effect. Conflicting data is represented as multiple edges connecting a regulator-gene pair:

$R_x = O_b - [54, Regulator\ Regulates\ Gene\ \{Source: \text{”}DBTBS\text{”},$
$Regulatory\ Effect: \text{”}Repression\text{”}\}]- > O_c$;

$R_y = O_b - [55, Regulator\ Regulates\ Gene\ \{Source: \text{”}RegPrecise\text{”},$
$Regulatory\ Effect: \text{”}Activation\text{”}\}]- > O_c$

Moreover, extracted data from *DBTBS* was not assigned to a specific *Bacillus subtilis* strain. To better assert data consistency with other sources, the most recently published *Bacillus subtilis* 168 genome (genome accession NC_000964.3) was downloaded from NCBI. All binding site sequences extracted from *DBTBS* were aligned against this genome, which is the most studied strain of *Bacillus subtilis*. Thus, it was possible to assemble two TRNs for *Bacillus subtilis* using *DBTBS* data: one for the strain 168 and another for a generic non specified strain.

Additionally, it is worth noting that the work developed by *Turkarslan et al.* [54] provided differential expression values of genes after induction of the corresponding regulators. This data was proved to follow a normal distribution, using the *Stats* module of the *SciPy* package. Afterwards, the retrieved data was divided by quartiles to further classify regulatory interactions by relative expression.

Thus, it was possible to assign ”Weak” or ”Strong” degrees of repressions and activations, based on the quartile distribution. Gene expression values lower than the first quartile of the dataset (lower than bottom 25% of the dataset) were classified as ”Weak” regulations. On the other hand, values higher than the third quartile (higher than top 75% of the dataset) were classified as ”Strong” regulations.

3.4.2 *Data loading*

Several tools were implemented to transform and load data from the *Staging Area*, or backup flat files, to the *TRN Collection*. All processed data is labelled with the source and loading date in the repository. While the loading procedure is mostly generic for data of any source, it was necessary to implement different modules to deal with heterogeneous information across all sources and optimize the organization of the repository.

In detail, all sources provided regulatory data, such as regulators and target genes, for one or more prokaryotic organisms. Nonetheless, only *Collectf*, *CoryneRegNet*, *DBTBS*, *RegPrecise* and *RegulonDB* provided binding site data. The proposed loading procedure prioritizes certain data properties, such as the Taxonomy ID of organisms, UniProt accession of both regulators and target genes as well as binding site sequences. As described in the previous section, these data properties are extremely important during the data transformation.

The proposed solution is iterative, analyzing every TRN retrieved from the extracted sources as well as data obtained through the *Knowledge Expansion* subsystem. At each iteration, new organisms are loaded into the *TRN Collection*, while duplicates are ignored. The same procedure is applied to regulators, target genes and binding sites. However, different sources may be responsible for updating pre-existing objects with new information (Example 8).

Example 8. *Updating entries in the TRN Collection*

Given three Escherichia coli *K12 Gene objects processed in the following order:*
$O_a = \{95, Gene, \{Source = \text{“}RegulonDB\text{”}, Name = \text{”}acrA\text{”}, Locus\ Tag = \text{”}b0463\text{”}\}\}$;
$O_b = \{96, Gene, \{Source = \text{“}Collectf\text{”}, Name = \text{”}acrA\text{”}, UniProt\ Accession = \text{”}P0AE06\text{”}\}\}$;
$O_c = \{97, Gene, \{Source = \text{“}RegPrecise\text{”}, Name = \text{”}acrA\text{”}, Sinonym = \text{”}mtcA\text{”}\}\}$;

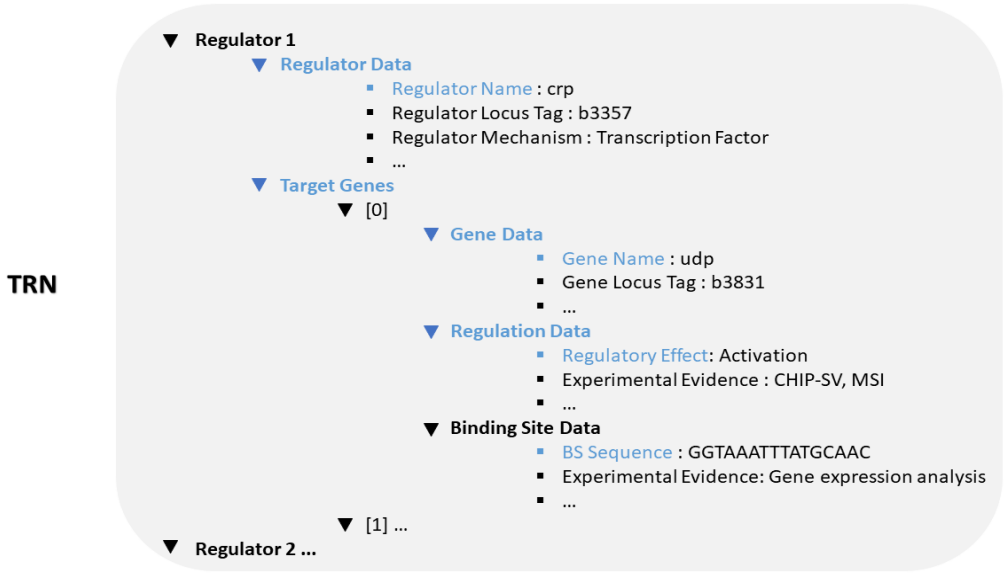
where the data was loaded into the TRN Collection as follows:
Step 1 - $O_a = \{95, Gene, \{Source = \text{“}RegulonDB\text{”}, Name = \text{”}acrA\text{”}, Locus\ Tag = \text{”}b0463\text{”}\}\}$;
Step 2 - $O_a = \{95, Gene, \{Source = \text{“}RegulonDB\text{”}, Name = \text{”}acrA\text{”}, Locus\ Tag = \text{”}b0463\text{”}, UniProt\ Accession = \text{”}P0AE06\text{”}\}\}$;
Step 3 - $O_a = \{95, Gene, \{Source = \text{“}RegulonDB\text{”}, Name = \text{”}acrA\text{”}, Locus\ Tag = \text{”}b0463\text{”}, UniProt\ Accession = \text{”}P0AE06\text{”}, Sinonym = \text{”}mtcA\text{”}\ ;\}\}$

the gene "acrA" found in Escherichia coli *K12 was first loaded in the TRN Collection while analyzing the TRN of RegulonDB. The* Escherichia coli *K12 regulatory data from Collectf also had an entry describing the "acrA" gene. As this gene was already loaded, it was not necessary*

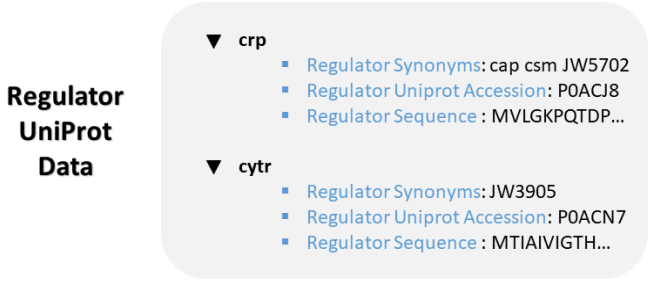
to duplicate the gene in the repository, although it was possible to update the pre-existing entity with the associated UniProt accession. The same process happened while analyzing RegPrecise data. In this case, the Gene object was updated with a gene synonym instead. As this gene was identified in three sources, three edges were connected from the gene to the associated regulator portraying the respective data source.

An additional *Python* tool was developed to automatize loading, maintenance and updates of the repository. The proposed ATRNI tool receives a TRN in JSON format as input and loads the contents directly in the *TRN Collection* by performing all required transformation and verification steps. This tool is not meant to replace the loaders implemented in the *Integration* subsystem, as it follows rigid rules and thus failing, for instance, to allocate data attributes in the optimal node or edge entities. Nonetheless, this limitation can be overcome through manual curation efforts.

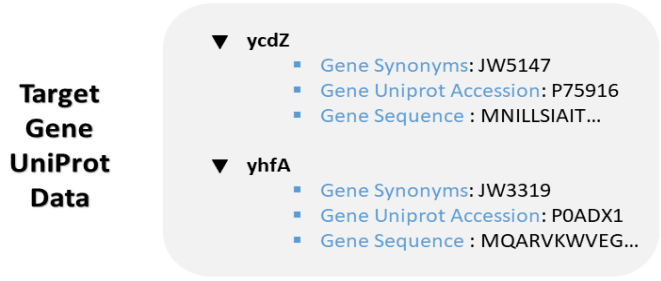
The ATRNI tool requires multiple input arguments, namely the data source, organism name and Taxonomy ID. Also, three JSON files containing the TRN in addition to UniProt information on the regulators and target genes must be provided. The structure of the required input files is shown in Figure 12. The main algorithm of the tool is described in Algorithm 1. Additional algorithms were implemented with the *Neo4j Python Driver* to select, create and update both objects and relationships as shown in Algorithm 2 and further elucidated in Example 9. The ATRNI tool can be used to integrate new TRNs or update previously loaded ones.



(a) Example of a JSON file containing a TRN



(b) Example of a JSON file containing UniProt data for regulators in the TRN



(c) Example of a JSON file containing UniProt data for target genes in the TRN

Figure 12.: Structure of the JSON files for the ATRNI tool. Fields in blue are mandatory, as the tool requires these data properties to perform verifications and merge data in the *TRN Collection*. *Gene Data* and *Regulation Data* are mandatory fields, whereas binding site data is optional. However, if binding site data is available, the binding sequence is mandatory. Fields in black are optional and there is no limit for the number of additional data properties to be included.

Algorithm 1 Automatic TRN Integrator - integrate(arg1, arg2, arg3, arg4, arg5, arg6)

```
1: Input: source
2:        organismTaxonomyID
3:        organismName
4:        TRN
5:        regulatorUniProt
6:        geneUniProt
7:
8: currentTime ← initialize current timestamp
9: driver ← connect to database
10: if organismTaxonomyID not in database then
11:     loadOrganism(driver, organismName, organismTaxonomyID, source, currentTime)
12: end if
13: for each regulator in TRN do
14:     regulatorData ← get regulator data from TRN
15:     if regulatorData in (regulatorUniProt) then
16:         add UniProt data to regulatorData
17:     end if
18:     if regulatorData in database then
19:         updateRegulator(driver, regulatorData, source, organismTaxonomyID, currentTime)
20:     else
21:         createRegulator(driver, regulatorData, source, organismTaxonomyID, currentTime)
22:     end if
23:     for each gene in TRN do
24:         geneData ← get gene data from TRN
25:         regulationData ← get regulation data from TRN
26:         bindingSiteData ← get binding site data from TRN
27:         if geneData in (geneUniProt) then
28:             add UniProt data to geneData
29:         end if
30:         integrateGene(driver, geneData, source, organismTaxonomyID, currentTime)
31:         if bindingSiteData then
32:             if bindingSiteData in database then
33:                 updateBindingSite(driver, geneData, regulatorData, bindingSiteData,
                    source, organismTaxonomyID, currentTime)
34:             else
35:                 createBindingSite(driver, geneData, regulatorData, bindingSiteData,
                    source, organismTaxonomyID, currentTime)
36:       end for
37: end for
```

Algorithm 2 Create or update Gene - integrateGene (arg1, arg2, arg3, arg4, arg5)

Input: *driver*
    geneData
    source
    organismTaxonomyID
    currentTime

if *geneData* in database **then**
    updateGene(*driver*, *geneData*, *source*, *organismTaxonomyID*, *currentTime*)
else
    createGene(*driver*, *geneData*, *source*, *organismTaxonomyID*, *currentTime*)
end if
if *GeneUniProtAccession* in *geneData* **then**
    match ← run *Neo4j* query
     (”MATCH (r:Regulator)-[rel:REGULATOR_IN_ORGANISM]− >(o:Organism)
     WHERE rel.UniprotAccession={*GeneUniProtAccession*}
     RETURN rel”)
    if *match* **then**
        run *Neo4j* query
        (”MATCH (g:Gene)
        WHERE where g.GeneUniprotAccession={*GeneUniProtAccession*}
        SET g.IsRegulator = True”)
    end if
end if

Example 9. *Loading Gene objects by UniProt accession*

Several Neo4j queries were formulated in the Python tools that constitute the Integration subsystem and the ATRNI. The following example uses the UniProt accession of a given gene to load new objects, avoiding duplicates:

”MATCH (o:Organism) where o.TaxonomyID = {tax}
MERGE (g:Gene {GeneUniprotAccession:{uniprot}})-[:GENE_IN_ORGANISM]->(o)
ON CREATE SET g.GeneSequence = {sequence},
g.GeneSynonyms = {synonyms}, g.GeneName = {name},
g.Source = {source}, g.timestamp = {time} ”;

where {tax, uniprot, sequence, synonyms, name, source, time} is a set of data provided to the loading tools or Automatic TRN Integrator algorithm. The MERGE command verifies

if there is a gene in the TRN Collection having the input UniProt accession as a property, connected to the given Organism object. If there is no match, the aforementioned object and relationship is created by the MERGE command. Additionally, ON CREATE SET is the clause that is activated when the MERGE command does not match any objects. ON MATCH SET (not disclosed in the example) is the clause that is activated when the "MERGE" command matches at least one object, with an exact pattern. Both clauses update the properties of the created/matched Gene object.

3.5 REQUEST SUBSYSTEM

After finishing the assembly of the CDS of the repository, a set of tools was implemented to retrieve regulatory information of interest. Thus, the *Request* subsystem is based on computational tools that combine *Cypher* query language with *Python*-based data structures adapted to the characteristics of the repository developed herein. It is worth noting that this subsystem was designed for obtaining data only available in the *TRN Collection*.

Additionally, a command line interface was designed within the scope of the *Request* subsystem. The goal of the development of this tool was to provide developers and users with a non-intrusive interface to access the contents of the repository, without requiring expertise with *Cypher* queries and knowledge regarding the *TRN Collection* structure.

One of the main features of the *Request* subsystem generates complete prokaryotic TRNs in XLSX format with several degrees of detail. Upon selecting the TRN of a given organism, users have the freedom to choose from three tiers of complexity. The highest complexity tier retrieves all data in the repository, including details such as protein sequences. On the other hand, the lowest complexity tier is focused at the regulatory interactions across an organism, returning a more compact version of the TRN.

3.6 REPORTING SUBSYSTEM

To grasp the topology of the *TRN Collection*, a *Report* subsystem has been developed. A *Python* module containing several functions was implemented to gather statistics on all objects and relationships. Thus, it was possible to generate several JSON files re-

vealing some insights on the contents of the *TRN Collection*, such as organism rankings by regulator count as well as the contribution of each external source to the repository.

Furthermore, it was necessary to graphically represent the reported statistics. This was possible, by using the *Matplotlib* and *Seaborn* packages to automate the process of generating graphics based on the aforementioned JSON files. A more insightful structural and graphical analysis of the repository will be performed in section 4.2.

3.7 WEB INTERFACE DEVELOPMENT

A web interface was developed to access the contents of the *TRN Collection*. This interface allows sharing data compiled in the repository to users without programming knowledge nor the burden to install additional software. Thus, a web interface was developed using the *Django* [136] framework in *Python*.

Using *Django*, it was not necessary to handle model databases within this framework, as the web application was in constant communication with the repository. In detail, the web application was deployed using both *Docker*, a container platform, and *NGINX* [137], a HTTP web server. For that, three docker containers were created using the *Docker Compose* tool for multi-container *Docker* applications. This has allowed the communication between the web application, HTTP web server and repository in *Neo4j*. Thus, future updates on the repository will be automatically applied to the web interface.

Finally, the front-end implementation was developed with HTML and *Cascading Style Sheet (CSS)* [138], thus the web application is expected to be compatible with all modern browsers.

4

RESULTS AND DISCUSSION

4.1 *ProTReND* REPOSITORY

The following subsections describe the main outcomes of this project, namely the implementation of an extensive TRN database, to be further referred to as ProTReND; tools to request its contents, including a command line interface; a tool to load and update TRNs in the database and a web interface to explore the contents of ProTReND, available at `http://bit.ly/ProTReND_DB`. The source code for the implementation of the web interface is available at `https://gitlab.bio.di.uminho.pt/DiogoLima/blueprint_repository.git`.

4.2 *ProTReND* DATABASE STRUCTURE

A discussion over the structure and contents of the ProTReND database implemented in this work, will be presented in this section. All further analyses will be aimed at the *TRN Collection* of the CDS, as it contains the processed data intended to be shared with other researchers and developers.

At the time of writing, the ProTReND database contains transcriptional regulatory data for 510 prokaryotic organisms, including two *Archaea* species and 508 *Bacteria* species. Moreover, data was compiled for approximately 1500 unique regulators and over 46000 target genes, across more than 67000 regulatory interactions, as summarized in Table 20.

Table 20.: Summary of the contents in the *TRN Collection* of the ProTReND database. It is worth mentioning that the regulators count was performed under the assumption that a regulator shared across multiple organisms is to be counted only once.

Entity	Count
Organisms	510
Unique regulators	1 490
Target genes	46 504
Binding Sites	62 592
Regulatory interactions	67 268

Figure 13 describes the graph schema designed for handling prokaryotic TRNs in the ProTReND database. Tables S2 and S3 of the supporting material provide a complete compendium of all node and edge properties in the ProTReND database.

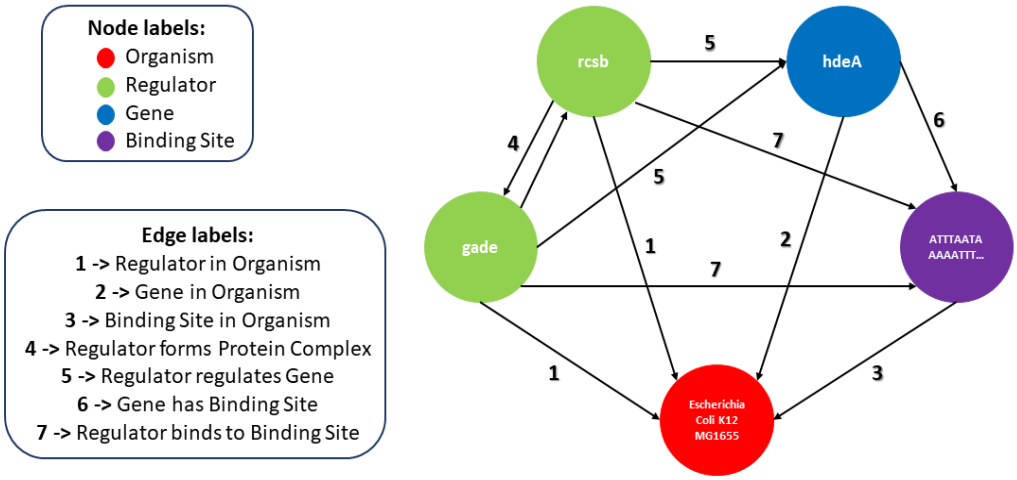


Figure 13.: Illustration of the *TRN Collection* schema in the ProTReND database. Nodes are represented by colored circles while the numbered arrows indicate relationships between entities.

Regulator in Organism relationships connect *Regulator* objects to *Organism* objects. This a *many-to-many* relationship that conveys the protein sequence and UniProt accession of each regulator (if available) in the corresponding organism, among other data. A given regulator may be present in several organisms, and organisms are expected to have multiple regulators in their genome.

Regulator binds to Binding Site relationships connect *Regulator* objects to *Binding Site* objects. This is likewise a *many-to-many* relationship as regulators that affect multiple genes bind to several binding sites, and it is possible for a given binding site to be co-bound by multiple regulatory entities.

Regulator forms Protein Complex relationships connect two *Regulator* objects which may group together and form a protein complex, co-regulating target genes. This is a *one-to-one* relationship as all retrieved protein complexes included two regulators only, and no intersections were found across regulator pairs. *Neo4j* does not support bidirectional edges, thus regulator pairs capable of forming a protein complex were connected by two counter-directional edges.

Binding site in Organism relationships connect *Binding Site* objects to *Organism* objects. This is a *many-to-one* relationship as a given binding site belongs to one and only one organism. Nevertheless, the genome of an organism may have multiple binding sites (*cis*-regulatory elements for multiple genes).

Gene has Binding Site relationships connect *Gene* objects to *Binding Site* objects. The connection is represented by a *many-to-many* relationship as a given gene can be regulated at multiple binding sites, and a binding site may affect the regulation of multiple genes (most likely belonging to the same operon).

Gene in Organism relationships connect *Gene* objects to *Organism* objects in a *many-to-one* relationship. Unlike regulators, genes with the same function and name across different organisms, were treated as different entities, simplifying the assembly of TRNs. herefore, a gene is unique in the genome of a single organism and genomes encompass multiple genes.

Regulator regulates Genes relationships connect *Regulator* objects to *Gene* objects in a *many-to-many* relationship. A given regulator may be responsible for regulating the expression of multiple genes, and a gene may be co-regulated by several regulatory entities. The distribution of each node and edge structure in the ProTReND database can be seen in Figure 14.

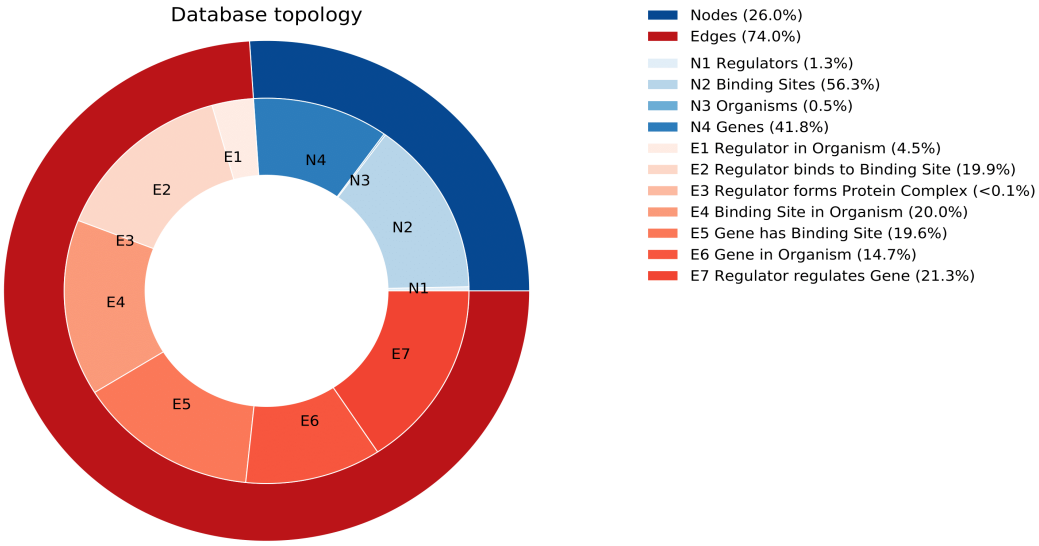


Figure 14.: ProTReND database node and edge distribution. Nodes are represented in shades of blue and edges are represented in shades of red. Percentage values for the different node and edge labels consist on the relative frequency to the total number of nodes and edges, respectively. Absolute frequency: Nodes (111 167); Edges (315 825); *N1* (1 490); *N2* (62 592); *N3* (581); *N4* (46 504); *E1* (14 161); *E2* (62 777); ; *E3* (2); *E4* (63 152); *E5* (61 957); *E6* (46 508); *E7* (67 268).

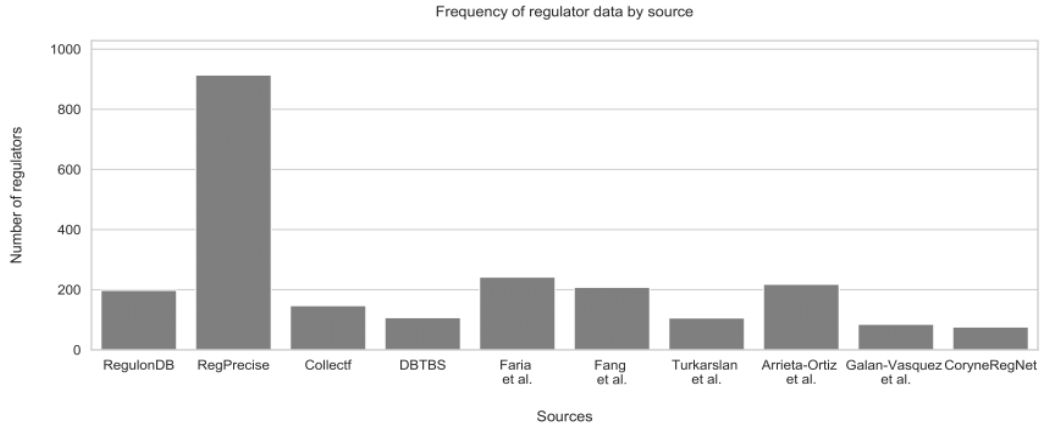
It is noteworthy that the number of relationships in the ProTReND database is approximately three times the number of nodes. This proportion supports the decision to select a graph database, as it better represents highly interconnected regulatory networks. *Regulator* and *Organism* node objects compose a vast minority of the database, as regulatory entities and prokaryotic organisms were stored in high degree central hub nodes, with a large number of relationships.

Regulator forms Protein Complex relationships are scarce as *RegulonDB* was the only source providing this type of information. Relationships involving binding site data are very well represented in the ProTReND database, due to the large contribution of *RegPrecise* in TFBS data, totaling almost 60% of the relationships.

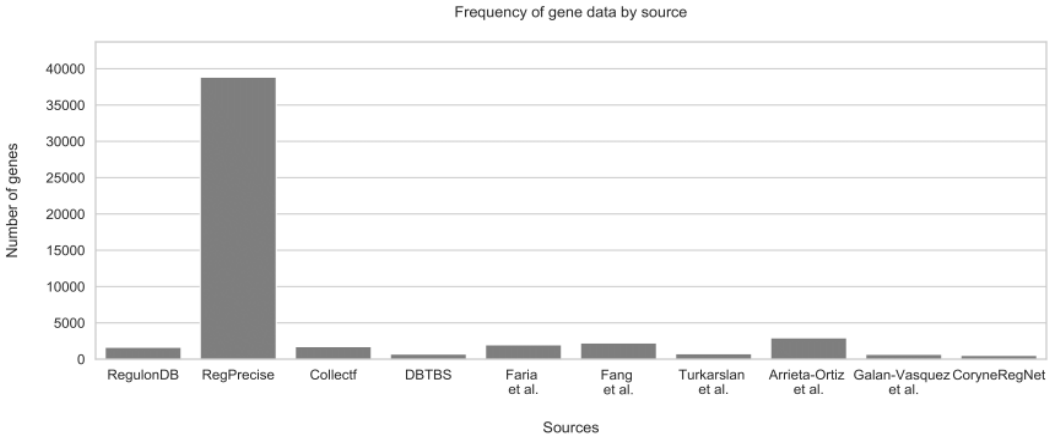
Regulator regulates Gene relationships are arguably the most important connections made throughout the development of this work, summing up to over 21% of all relationships. These relationships were found to be 1.5% more frequent than the average proportion of relationships involving binding sites. This difference is explained by two reasons: not all retrieved regulatory interactions contained binding site data; over-

lapping regulatory data was stored in multiple *Regulator regulates Gene* relationships, providing manifold sources of evidence.

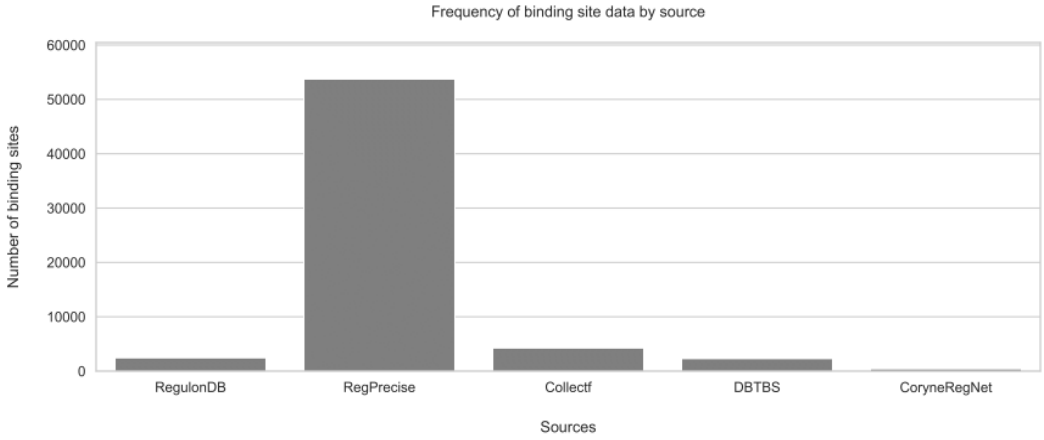
The contribution of each external source for the development of the ProTReND database is shown in Figure 15. *RegPrecise* is the main source of transcriptional regulatory data providing regulatory interactions for 914 unique regulators, targeting 38844 genes at 53733 binding sites. Regarding data contribution, there is no significant difference among organism-specific and non-organism-specific sources. Besides *RegPrecise*, it is possible to find a balance between the data contribution of each analyzed source.



(a) Regulator frequency in the *TRN Collection* by source



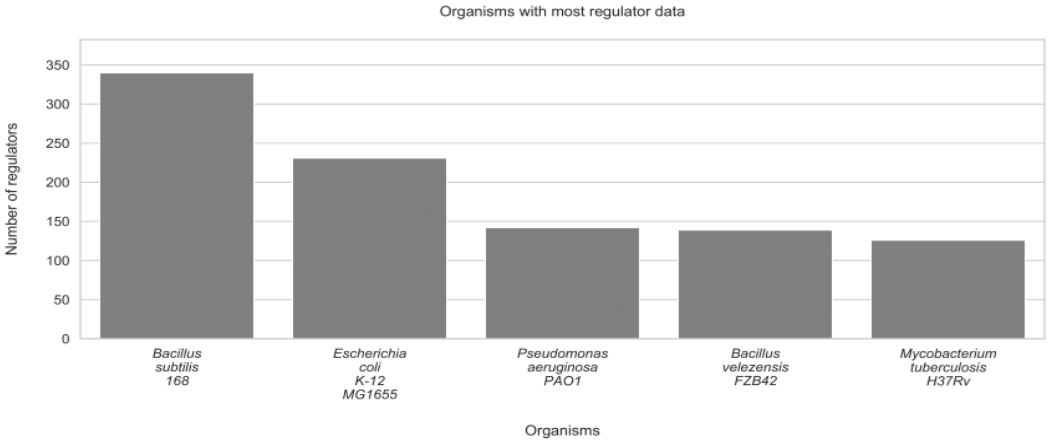
(b) Gene frequency in the *TRN Collection* by source



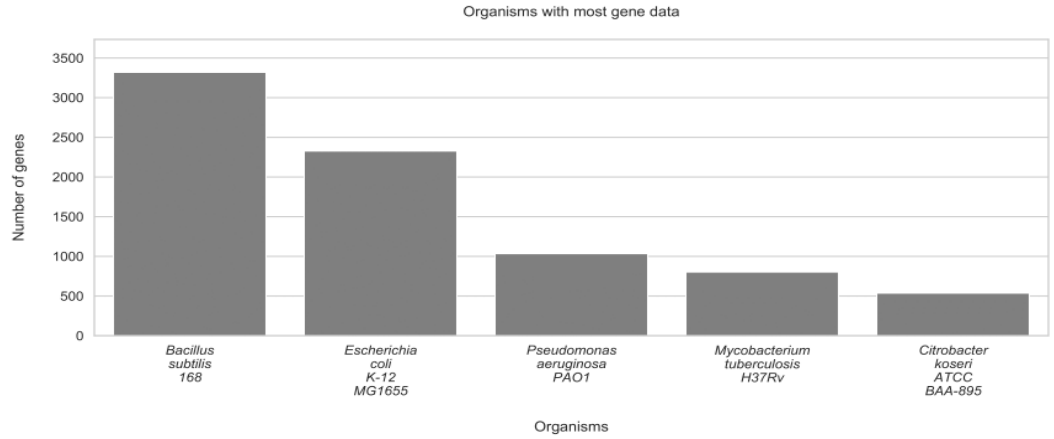
(c) Binding site frequency in the *TRN Collection* by source

Figure 15.: External source contribution to the ProTReND database, regarding number of regulators, genes and binding sites. Data was extracted from *RegulonDB* [23], *DBTBS* [24], *Collectf* [26], *RegPrecise* [27], *CoryneRegNet* [25], *Fang et al.* [52], *Faria et al.* [33], *Arrieta-Ortiz et al.* [53], *Turkarslan et al.* [54] and *Galán-Vásquez et al.* [55].

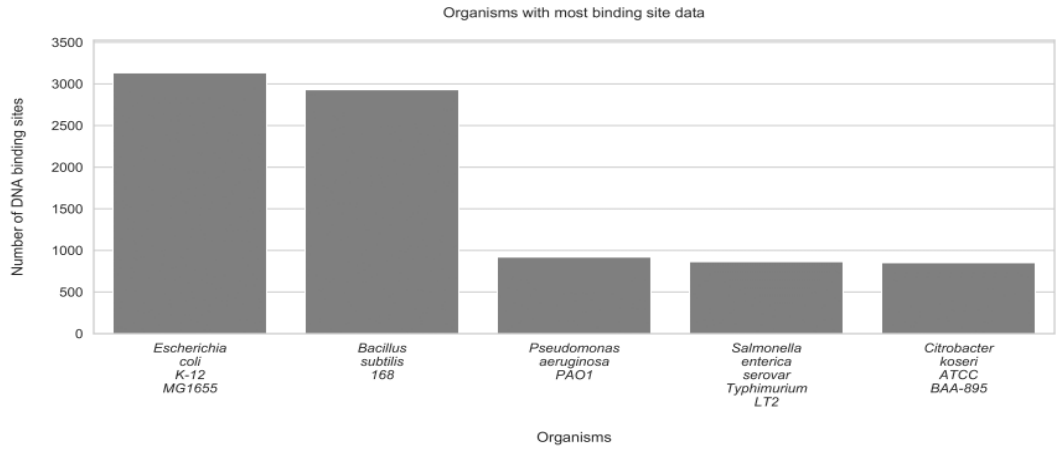
Figure 16 details the prokaryotic organisms with more regulatory entities assigned. *Bacillus subtilis* 168 and *Escherichia coli* K-12 MG1655 are the organisms with the most retrieved regulatory data. As both species are extensively described in the literature, it was possible to analyze data from five sources specifically describing regulatory interactions of these organisms. It was possible to expand the TRN knowledge of both model organisms: 32% increase of target genes when compared with the study developed by *Fang et al.* [52] (from 1764 to 2327 genes), which was the most extensive analyzed source of *Escherichia coli* K-12 MG1655 regulatory information; 8% increase of target genes when compared with the study developed by *Arrieta-Ortiz et al.* [53] (from 3086 to 3319 genes), the most extensive analyzed source of *Bacillus subtilis* 168 regulatory information.



(a) Organisms with the largest regulator set in the *TRN Collection*



(b) Organisms with the largest gene set in the *TRN Collection*



(c) Organisms with the largest binding site set in the *TRN Collection*

Figure 16.: Best represented organisms in the ProTReND database regarding the number of regulators, genes and binding sites.

It was also feasible to improve current knowledge regarding the regulatory machinery of non-model organisms, such as *Pseudomonas aeruginosa* PAO1, with a 35% increase of target genes on the work performed by *Galán-Vásquez et al.* [55] (from 593 to 802 genes). Although the vast majority of retrieved regulators are TFs, other regulatory mechanisms, such as riboswitches and sigma factors, were also collected, as shown in Figure 17.

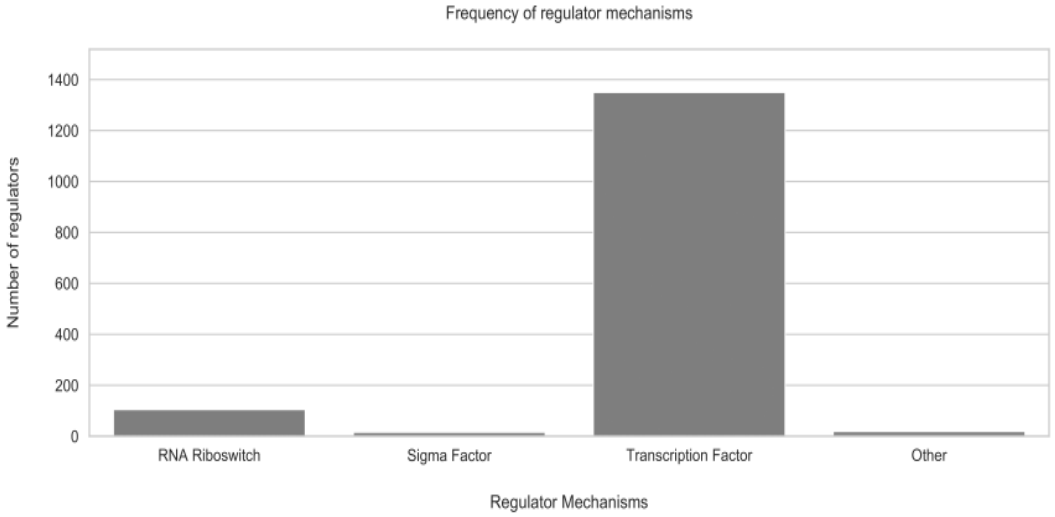


Figure 17.: Summary of regulatory mechanisms in the *TRN Collection* of the ProTReND database.

As regulators were stored in high-degree hub nodes connected to the respective organisms, it was possible to validate the presence of ubiquitous regulators in several species. As shown in Figure 18, certain regulators are part of the TRN of over 200 analyzed organisms, such as:

- the *tpp* riboswitch that responds to the coenzyme thiamine pyrophosphate [139];
- the *nrdr* TF which regulates the expression of ribonucleotide reductase genes [140];
- the *lexa* TF that regulates the expression of genes involved in the response to DNA damage [141];
- the *hrca* TF associated with heat shock response [142];
- the *fmn* riboswitch that controls the expression of flavin mononucleotide biosynthesis and transport genes [143].

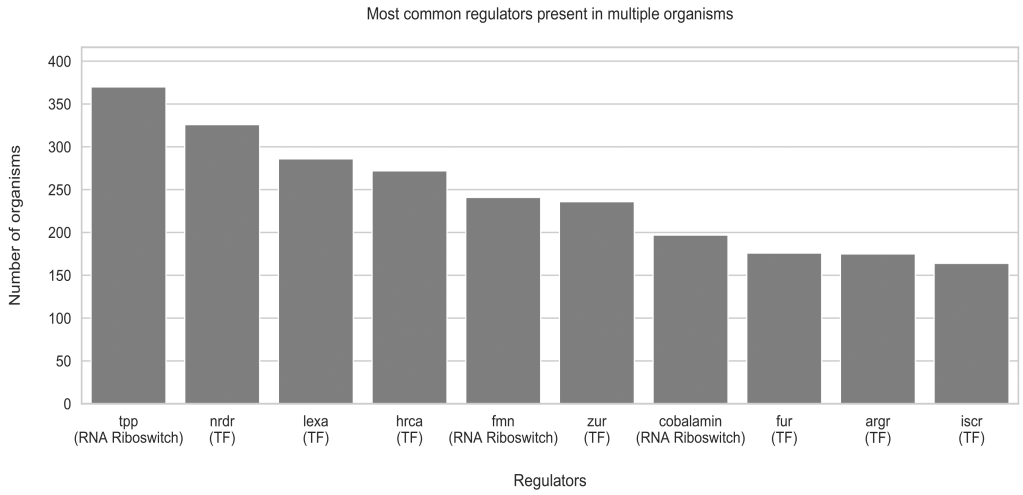


Figure 18.: Regulators most frequently found in the TRN of distinct prokaryotic organisms, identified by name and mechanism.

4.3 REQUEST SUBSYSTEM: COMMAND LINE INTERFACE

A command line interface was developed to ease the access to the contents of ProTReND. This section provides a simple demonstration of the developed software, emulating user experience. After installing the required dependencies it is possible to run the proposed *Python* program and access an interactive menu, as shown in Figures 19 to 22. Data obtained with the command line interface is bound to a given organism by Taxonomy ID, thus users must know the Taxonomy ID of the organism of interest. As shown in Figure 19, a simple search system was implemented to return the organism Taxonomy ID or a list of possible organisms, after typing in the entire or partial organism name, respectively.

After selecting a given organism, users may filter the output TRN by source, as shown in Figure 20. The command line interface is directly connected to the database, thus users can generate a XLSX file with the most recent TRN version in a predefined directory. Table 21 presents execution times to assemble, convert and generate an organism TRN using the request command line interface.

```
Press 0 for help
Press 1 to choose the output directory (currently RequestDemonstration/)
Press 2 to request an organism Taxonomy ID
Press 3 to request a complete TRN
Press 4 to request a compact TRN
Press 5 to request an ultra compact TRN
Press 6 to request an organism name
Press 7 to request an organism set of genes
Press 8 to request an organism set of regulators
Press 9 to request an organism set of binding sites
Press 10 to request a set of organisms that share a common regulator
Enter 'sources' to request all sources in the repository
Enter 'exit' to exit the program

Enter your option: 2
Enter the organism name: Mycobacterium
Result:  The requested organism name returns multiple results,
enter a more specific organism name (following results are some of the matched examples)
[('262316', 'Mycobacterium avium subsp. paratuberculosis K-10'), ('272631', 'Mycobacterium leprae TN'),
('164757', 'Mycobacterium sp. JLS'), ('189918', 'Mycobacterium sp. KMS'), ('83332', 'Mycobacterium tuberculosis H37Rv')]

Enter your option: 2
Enter the organism name: Mycobacterium tuberculosis H37Rv
Result: 83332
```

Figure 19.: Main menu of the request command line interface and a demonstration of partial and perfect hits for organism searches in the ProTReND database.

```
Press 0 for help
Press 1 to choose the output directory (currently RequestDemonstration/)
Press 2 to request an organism Taxonomy ID
Press 3 to request a complete TRN
Press 4 to request a compact TRN
Press 5 to request an ultra compact TRN
Press 6 to request an organism name
Press 7 to request an organism set of genes
Press 8 to request an organism set of regulators
Press 9 to request an organism set of binding sites
Press 10 to request a set of organisms that share a common regulator
Enter 'sources' to request all sources in the repository
Enter 'exit' to exit the program

Enter your option: 3
Enter the organism Taxonomy ID: 83332
Do you want to filter the regulatory network by source ? Enter y/n: n
Process completed

Enter your option: 5
Enter the organism Taxonomy ID: 83332
Do you want to filter the regulatory network by source ? Enter y/n: y
['Turkarslan 2015', 'RegPrecise', 'Collectf']

Enter the desired sources as written above, separated by commas (,): Turkarslan 2015
Process completed
```

Figure 20.: Demonstration of TRN retrieval from the ProTReND database through the request command line interface.

Table 21.: Average execution time of TRN request by network size (number of target genes), using the command line interface. Ten case studies were analysed using complete TRNs, including amino acid and nucleotide sequences, without data/source filtering.

Organism	**Genes count**	**Time (seconds)**
Bacillus subtilis 168	3 319	21
Escherichia coli K-12 MG1655	2 327	37
Pseudomonas aeruginosa PAO1	1 034	3
Mycobacterium tuberculosis H37Rv	802	2
Agrobacterium fabrum C58	204	<1

Gene, regulator or binding site sets of a given organism can also be individually and separately requested in JSON format, as shown in Figure 21. Alternatively, Figure 22 exhibits the organism data retrieval in JSON format, for regulators shared across several species. This information may help to enlighten conservation of regulatory mechanisms in different prokaryotic lineages.

```
Press 0 for help
Press 1 to choose the output directory (currently RequestDemonstration/)
Press 2 to request an organism Taxonomy ID
Press 3 to request a complete TRN
Press 4 to request a compact TRN
Press 5 to request an ultra compact TRN
Press 6 to request an organism name
Press 7 to request an organism set of genes
Press 8 to request an organism set of regulators
Press 9 to request an organism set of binding sites
Press 10 to request a set of organisms that share a common regulator
Enter 'sources' to request all sources in the repository
Enter 'exit' to exit the program

Enter your option: 7
Enter the organism Taxonomy ID: 83332
Successfuly requested 802 genes
Process completed

Enter your option: 8
Enter the organism Taxonomy ID: 83332
Successfuly requested 126 regulators
Process completed

Enter your option: 9
Enter the organism Taxonomy ID: 83332
Successfuly requested 408 binding sites
Process completed
```

Figure 21.: Demonstration of regulatory data retrieval from the ProTReND database through the request command line interface, including the gene, regulator and binding site set of a given organism.

```
Press 0 for help
Press 1 to choose the output directory (currently RequestDemonstration/)
Press 2 to request an organism Taxonomy ID
Press 3 to request a complete TRN
Press 4 to request a compact TRN
Press 5 to request an ultra compact TRN
Press 6 to request an organism name
Press 7 to request an organism set of genes
Press 8 to request an organism set of regulators
Press 9 to request an organism set of binding sites
Press 10 to request a set of organisms that share a common regulator
Enter 'sources' to request all sources in the repository
Enter 'exit' to exit the program

Enter your option: 10
Enter the regulator name: lexa
Successfuly requested 286 organisms where, lexa is present
Process completed
```

Figure 22.: Demonstration of data retrieval from the ProTReND database through the request command line interface, regarding shared regulators across distinct organisms.

4.4 *ATRNI*: showcase

A solution was developed to automatically integrate new prokaryotic TRNs or update previously loaded ones. This section provides a demonstration of the ATRNI tool. A test case based on a fraction of the *Bdellovibrio bacteriovorus HD100* TRN is used here for demonstration purposes. The test case TRN was retrieved from the *TRN Collection*. The network depicted in Figure 23 consists on a TF named *lexa*, with the UniProt accession *Q6MHM9*, that co-represses two genes, namely *recA* (*Q6MQS4*) and *dnaE2* (*Q6MQS6*), and binds at the binding site sequence shown in the Figure (data extracted from *Collectf*). As shown in Table 22, a mock-up regulatory dataset was designed to showcase the ATRNI tool.

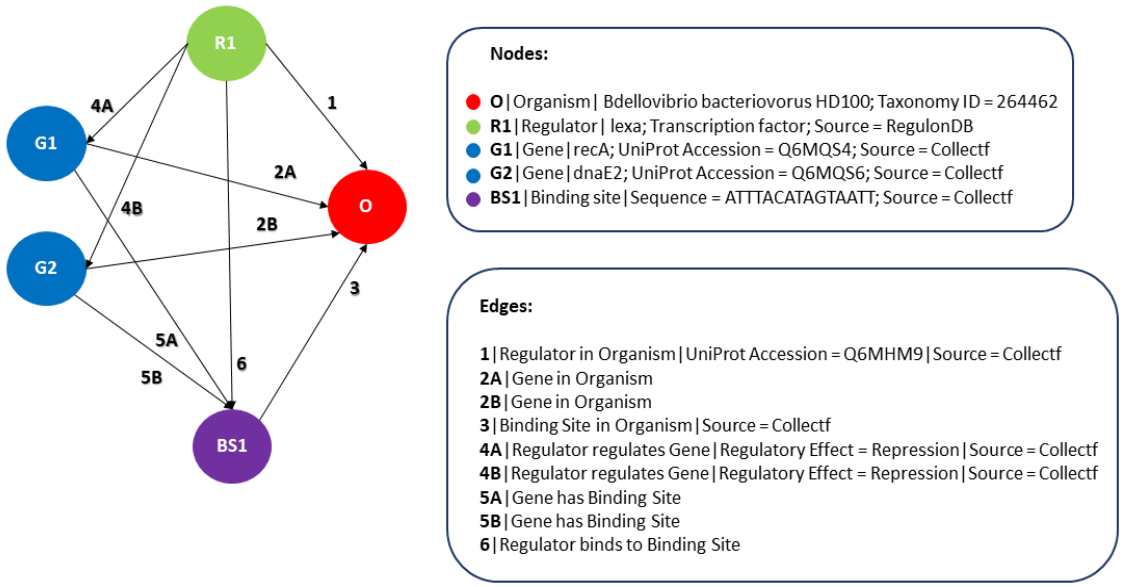


Figure 23.: Test case for the ATRNI tool based on a fraction of the *Bdellovibrio bacteriovorus HD100* TRN simplified for illustration purposes and before using the tool. The letter notation after a given edge number distinguishes multiple instances of the same edge label.

Table 22.: Input data for testing the ATRNI tool. This is part of a mock-up dataset, therefore it has no biological context.

Regulator UniProt accession	**Regulator name**	**Gene UniProt accession**	**Gene name**	**Binding Site Sequence**	**Regulatory Effect**
Q6MHM9	lexa	Q6MQS4	recA	ATTTACAT AGTAATT	Repression
Q6MHM9	lexa	Q6MQS4	recA	ATTTACAT AGTAATT	Activation
Q6MHM9	lexa	Q6MQS9	tpbA	AGGCATGA GCAATTA	Repression
Q6SPP1	dpr	Q6SMH8	xlaB	ATGAAGGA CCCAGTG	Activation
Q6SPP1	dpr	Q6SMH8	xlaB	ATGAAGGA CCCAGTG	Repression

The mock-up dataset contains five rows, each to simulate different scenarios:

- Data already loaded in the database, yet available in a different source.
- Entry with a new regulatory effect (possible evidence of dual regulation) for preexisting *Regulator-Gene-Binding Site* data.
- A new target gene with a new binding site sequence, associated with a preexisting regulator.
- New *Regulator-Gene-Binding Site* interaction.
- Entry with a new regulatory effect (possible evidence of dual regulation) for the previous *Regulator-Gene-Binding Site* interaction.

Figure 24 describes the logic employed in the main algorithm of this tool. Firstly, since the organism of the mock-up dataset was already loaded in the database, the organism object is not duplicated. The first row of the dataset provided another source for the data already loaded in the database. Thus, the *Regulator* and *Gene* data matched preexisting data by UniProt accession, the binding site sequence is considered to be

duplicate (preexisting and connected to the same organism), and the regulatory effect was already accounted for. Nevertheless, it was necessary to register that this preexisting data is also in the currently analyzed source (ATRNI demonstration). Hence, a new *Regulator regulates Gene* relationship was created, with a different source attribute (and other non portrayed attributes, such as timestamps).

The second row provided a new regulatory effect for the data in the current state of the database. Hence, another *Regulator regulates Gene* relationship between the *lexa Regulator* and the *recA* gene was added.

Alternatively, the third row provided data for a new target gene and associated binding site sequence. This results into new *Gene* and *Bindind Site* objects, as well as the corresponding *Gene in Organism*, *Binding Site in Organism*, *Gene has Binding Site*, *Regulator regulates Gene* and *Regulator binds to Binding Site* relationships.

The fourth row presented a new regulatory interaction without duplicated entities, thus it was necessary to load all associated objects and relationships.

Finally, the fifth row provided evidence for an alternate regulatory effect for preexisting regulatory entities (loaded in the previous iteration), thus creating a second *Regulator regulates Gene* relationship. Figure 25 illustrates the TRN of *Bdellovibrio bacteriovorus HD100* after using the ATRNI tool with the mock-up dataset.

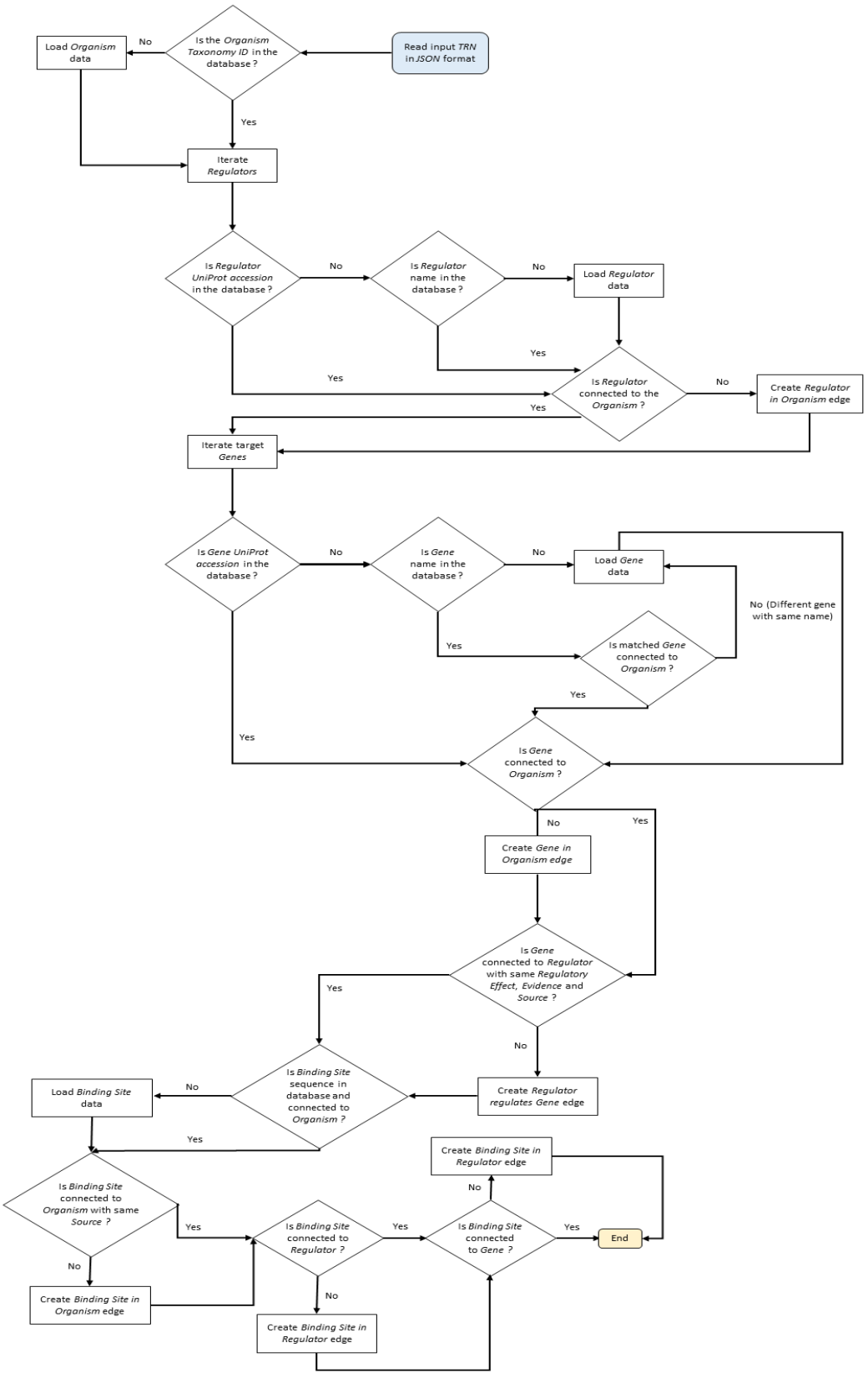


Figure 24.: Decision flowchart of the ATRNI tool. The blue rounded rectangle represents the input of the ATRNI tool, and the golden rounded rectangle represents the finishing point of the flowchart. Squared rectangles and diamonds depict operations and decisions, respectively.

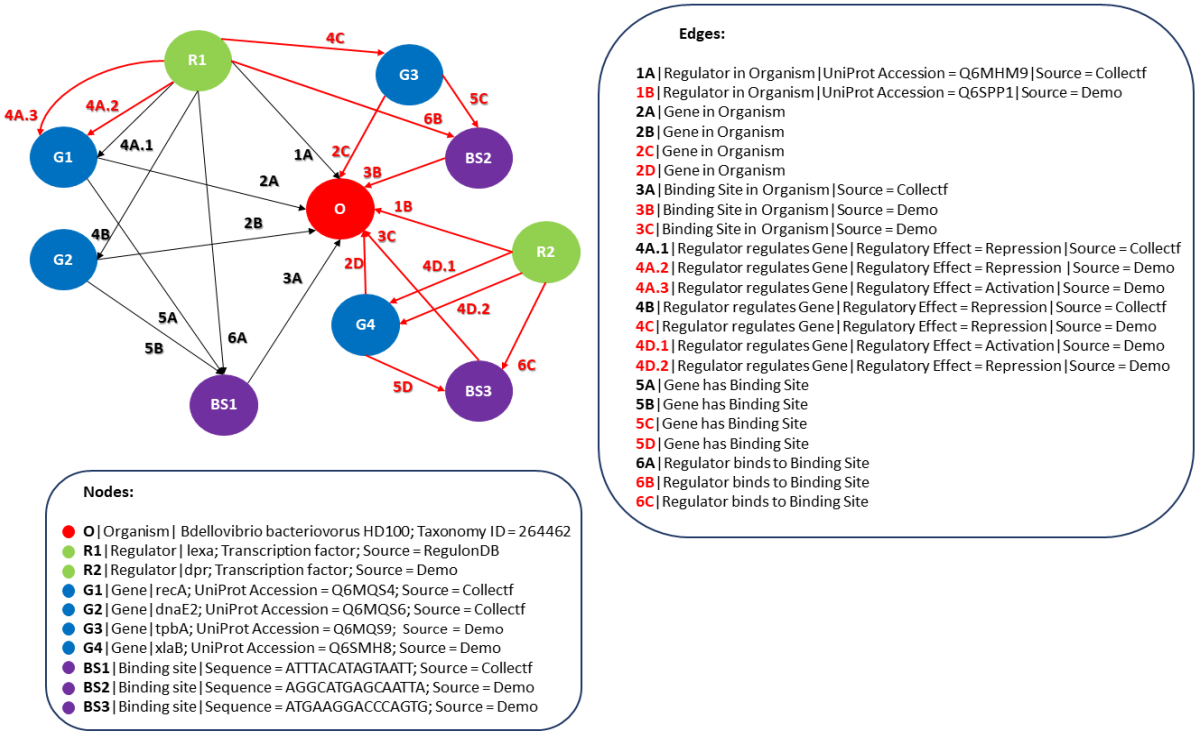


Figure 25.: Test case for the ATRNI tool based on a fraction of the *Bdellovibrio bacteriovorus HD100* TRN simplified for illustration purposes and after using the tool. Newly loaded edges are highlighted in red. The letter notation after a given edge number distinguishes multiple instances of the same edge label. The *.1* and *.2* notation distinguishes multiple instances of the same edge label between the same pair of nodes.

4.5 *ProTReND* web interface

ProTReND is a web application designed to make the contents of the repository available to users without programming knowledge or command line interface experience. The goal of this section is to demonstrate and explore some of the most relevant features of ProTReND.

The web interface was designed to be as user-friendly as possible. For that, two effective views were implemented for accessing the regulatory data. A simple browsing system was designed for both organisms and regulators allowing users to select the desired option among a list of entities, separated by alphabetical groups. Moreover, besides these browsing systems, a robust search engine was implemented in the navigation bar of the website as well as in the homepage, as shown in Figure 26.

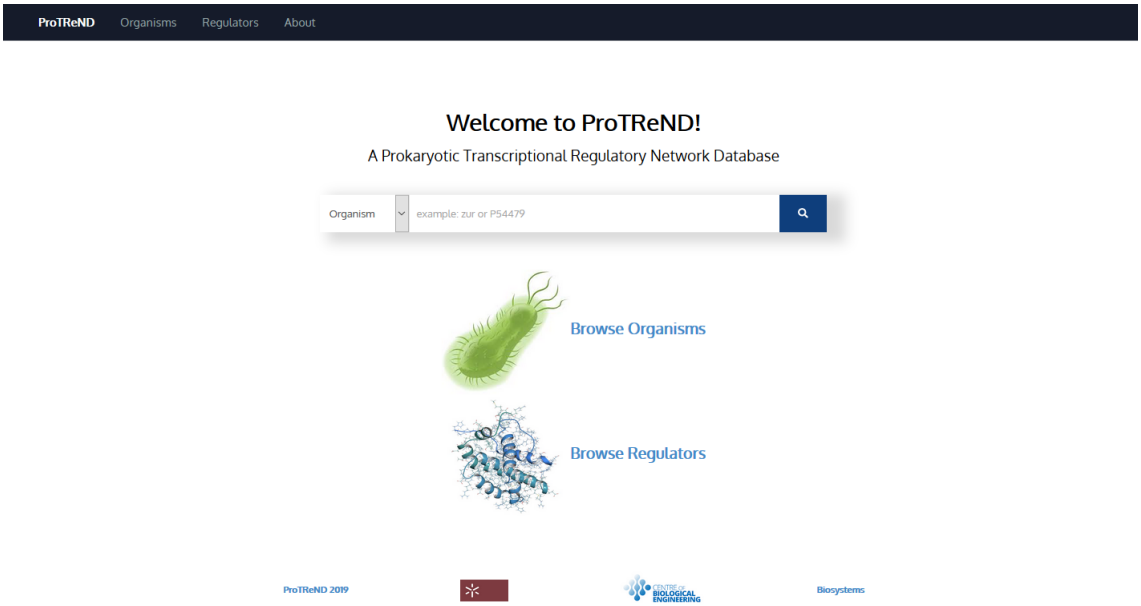


Figure 26.: ProTReND home page. One may access the contents of the repository with the web application by browsing a set of regulators or organisms, as well as entering a query input in the integrated search engine.

The search engine is divided in three options: organisms, regulators and genes. Queries are performed dynamically to the *TRN Collection*, thus users always have access to the most recent version of the ProTReND database. Searches can be performed for any descriptive identifier, such as common names, Taxonomy IDs, UniProt accessions or gene locus tags. If the query returns a perfect hit, the user is redirected to a new page containing the requested data. On the other hand, if the query returns a partial hit, therefore matching several entities, a table is displayed summarizing all search hits. The latter outcome is shown in Figure 27.

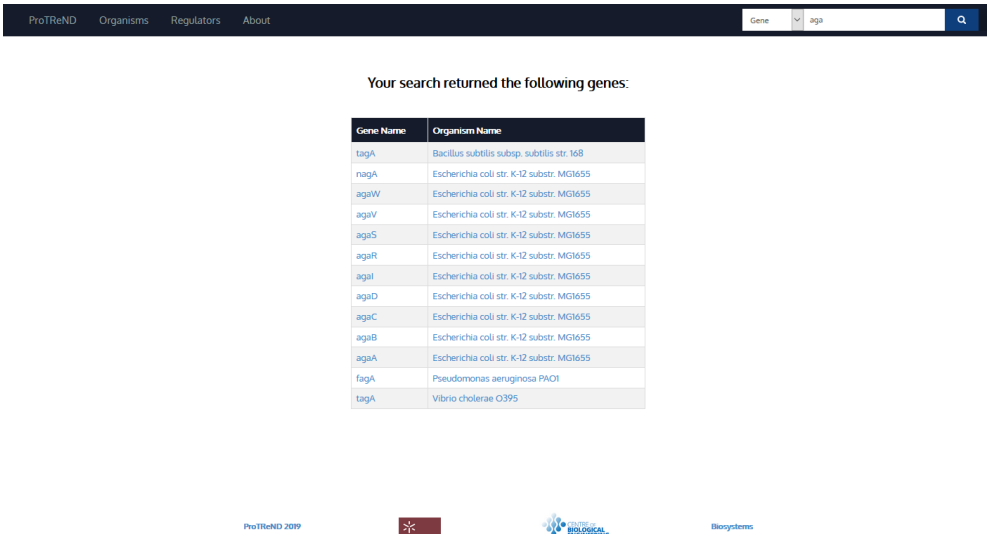


Figure 27.: Demonstration of the ProTReND gene search engine. User input search for "aga" in the gene section returns multiple hits, thus a table is generated with the results.

Upon selecting an organism, one may access external sources for supplementary data, such as the *NCBI Taxonomy* or genome sequence pages. Additionally, it is possible to download the complete TRN of the selected organism, in XLSX format, as shown in Figure 28. The corresponding set of regulators is listed in the organism detail page and can be accessed individually.

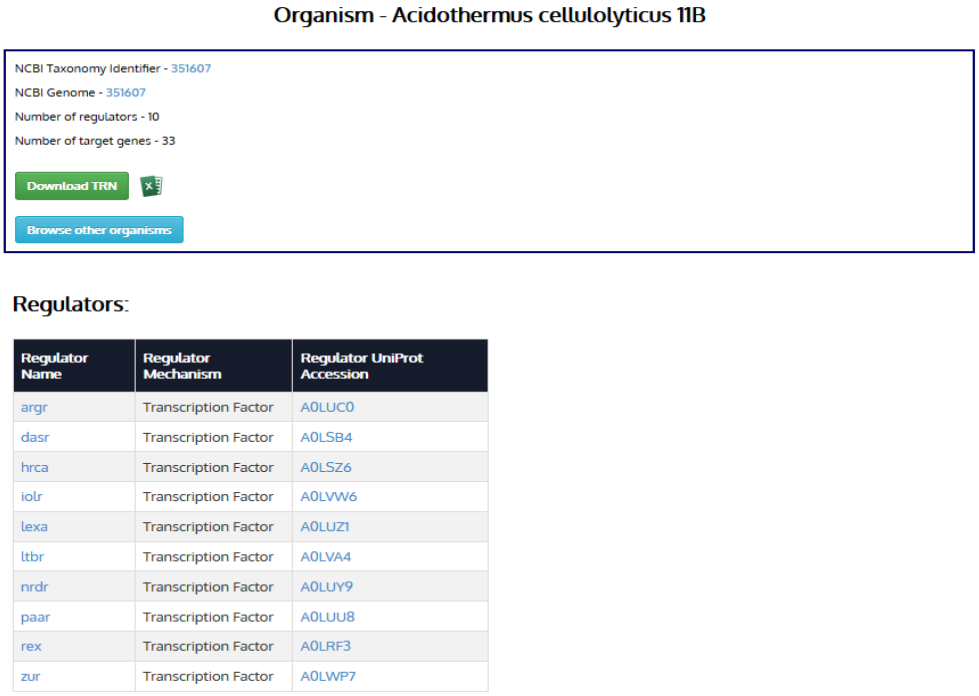


Figure 28.: Example of a given organism detail web page. Besides organism data and external links, this page also contains a brief summary of all identified regulators for this species.

The regulator detail page contains all regulatory data concerning the TF or another regulatory mechanism, including an external link to its UniProt record, and the respective FASTA sequence (when available), as shown in Figure 29. Moreover, it is possible to browse the associated set of regulated genes.

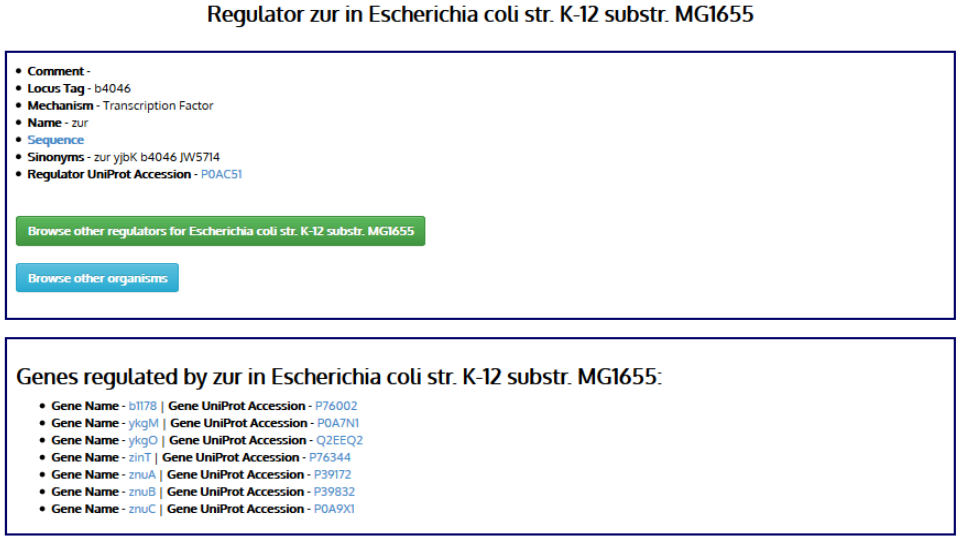


Figure 29.: Example of a given regulator detail web page. This page contains all compiled data for a given regulator and the respective set of regulated genes.

Likewise, the gene detail page contains compiled data, including both gene nucleotide and amino acid FASTA sequences, and an external link for the associated UniProt record. As detailed in Figure 30, a compilation of all regulatory interactions involving a given gene is also available in this detail page. Results may show some redundancy, however there are no duplicate regulatory interactions. Portrayed data includes binding site sequences, retrieved evidence, effectors and biological processes associated with the regulators and corresponding target genes.



Figure 30.: Example of a given gene detail web page. This page contains all retrieved gene data and associated regulatory interactions. In this example, the expression of the *tagA* gene of *Vibrio cholerae o395* was found to be activated by the *toxt* TF at two different binding site sequences (in proximal genome regions). *PubMed* evidence is available for further clarification.

5

CONCLUSION

5.1 CONCLUDING REMARKS

The goal of the work developed in this dissertation was to implement a repository of TRNs for prokaryotic organisms. The development of a state-of-the-art repository with regulatory data regarding approximately 1500 regulators controlling expression levels of over 46000 genes across 510 organisms, allowed bridging the gap between organism-specific and non-organism-specific databases.

Data was extracted and processed from five databases: *RegulonDB* [23] , *DBTBS* [24], *Collectf* [26], *RegPrecise* [27] and *CoryneRegNet* [25]. Besides data extracted from public databases, the repository was extended with data from the works of *Fang et al.* [52], *Faria et al.* [33], *Arrieta-Ortiz et al.* [53], *Turkarslan et al.* [54] and *Galán-Vásquez et al.* [55]. Combining these sources of regulatory information resulted, non-exclusively, in the assembly of TRNs with a 35%, 32% and 8% increase of target genes when compared with the state-of-the-art TRNs of *Pseudomonas aeruginosa* PAO1 [55], *Escherichia coli* K-12 MG1655 [52] and *Bacillus subtilis* 168 [53], respectively.

The repository was implemented using *Neo4j* [12], a graph database management system. The vast connectivity of biological regulatory networks supported the decision of representing the acquired information in a graph-based structure. The fluid schema of graph databases proved to be particularly useful throughout the development of this work as extracted data proved to be of heterogeneous nature.

After accomplishing the proposed objective of implementing an extensive compendium of standardized TRNs, focus was redirected on developing additional software for accessing and manipulating the contents of the repository. Thus, a set of *Python* tools was implemented to ease the maintenance of the repository by future developers. Additionally, a command line interface was developed for users with less programming expertise. A tool for automating the data loading procedure and cus-

tom parsers for processing data extracted from new sources were created to ease the process of updating the database.

Finally, a web application that can be accessed at `http://bit.ly/ProTReND_DB` was developed to make the data retrieved in this work available to all interested parties, regardless of programming knowledge. The source code for the web application is available at `https://gitlab.bio.di.uminho.pt/DiogoLima/blueprint_repository.git` as well as all code developed for the implementation and manipulation of the repository.

5.2 FUTURE DIRECTIONS

It is expected that the repository, developed during this work, will be updated as new relevant regulatory data is available, as already explored datasources will most likely be updated and new sources might arise, further complementing ProTReND. Moreover, as discussed in section 1.1, the proposed repository may contribute to the development of a TRN inferring software, capable of integrating regulatory data into GSM models. This software is expected to be developed within the *The Department of Energy Systems Biology Knowledgebase (KBase)* platform [144], a software and data platform for systems biology projects. Therefore, future work may involve the integration of the repository and developed tools in KBase.

5.3 OUTCOMES

The following outcome was achieved throughout the development of this dissertation:

- Cruz, F.; Lima, D.; Faria, J.P.; Rocha, M.; Dias, O. Towards the reconstruction of integrated genome-scale models of metabolism and gene expression. Advances in Intelligent Systems and Computing, 1005 (PACBB 2019), 173-181, 2020

BIBLIOGRAPHY

[1] Michael L Metzker. Sequencing technologies — the next generation. *Nature Reviews Genetics*, 11(1):31–46, Aug 2009.

[2] Hiroaki Kitano. Computational systems biology. *Nature*, 420(6912):206, 2002.

[3] Tae Yong Kim, Seung Bum Sohn, Yu Bin Kim, Won Jun Kim, and Sang Yup Lee. Recent advances in reconstruction and applications of genome-scale metabolic models. *Current opinion in biotechnology*, 23(4):617–623, 2012.

[4] Jeffrey D Orth, Ines Thiele, and Bernhard Ø Palsson. What is flux balance analysis? *Nature biotechnology*, 28(3):245–248, 2010.

[5] Christof Francke, Roland J Siezen, and Bas Teusink. Reconstructing the metabolic network of a bacterium from its genome. *Trends in microbiology*, 13(11):550–558, 2005.

[6] A Ivanova and A Lykidis. *Encyclopedia of Microbiology: Metabolic Reconstruction*. Elsevier Inc, 2009. SBN 9780123739445.

[7] Markus W Covert and Bernhard Ø Palsson. Transcriptional regulation in constraints-based metabolic models of escherichia coli. *Journal of Biological Chemistry*, 277(31):28058–28064, 2002.

[8] Kevin Struhl. Fundamentally different logic of gene regulation in eukaryotes and prokaryotes. *Cell*, 98(1):1–4, 1999.

[9] Oscar Dias, Miguel Rocha, Eugénio C Ferreira, and Isabel Rocha. Reconstructing genome-scale metabolic models with merlin. *Nucleic acids research*, page gkv294, 2015.

[10] Rachel Heery and Sheila Anderson. Digital repositories review. 2005.

[11] José P Faria, Ross Overbeek, Fangfang Xia, Miguel Rocha, Isabel Rocha, and Christopher S Henry. Genome-scale bacterial transcriptional regulatory networks: reconstruction and integrated analysis with metabolic models. *Briefings in Bioinformatics*, 15(4):592–611, 2014.

[12] Justin J Miller. Graph database applications and concepts with neo4j. In *Proceedings of the Southern Association for Information Systems Conference, Atlanta, GA, USA*, volume 2324, page 36, 2013.

[13] ABM Moniruzzaman and Syed Akhter Hossain. Nosql database: New era of databases for big data analytics-classification, characteristics and comparison. *arXiv preprint arXiv:1307.0191*, 2013.

[14] Dirk Merkel. Docker: lightweight linux containers for consistent development and deployment. *Linux Journal*, 2014(239):2, 2014.

[15] Bruce Alberts, Alexander Johnson, Julian Lewis, Martin Raff, Keith Roberts, and Peter Walter. Cell movements and the shaping of the vertebrate body. 2002.

[16] Amy Ralston and Kenna Shaw. Gene expression regulates cell differentiation. *Nature education*, 1(1):127, 2008.

[17] Jonathan MW Slack. Metaplasia and transdifferentiation: from pure biology to the clinic. *Nature reviews Molecular cell biology*, 8(5):369, 2007.

[18] Suzanne Clancy and William Brown. Translation: Dna to mrna to protein. *Nature Education*, 1(1):101, 2008.

[19] Jacques Monod and François Jacob. *Journal of Molecular biology: Genetic regulatory mechanisms in the synthesis of proteins*. Elsevier Ltd, 1961.

[20] Rye et al. *Biology: Prokaryotic Gene Regulation*. 2017.

[21] Theresa Phillips. Regulation of transcription and gene expression in eukaryotes. *Nature Education*, 1(1):199, 2008.

[22] Semen A Leyn, Marat D Kazanov, Natalia V Sernova, Ekaterina O Ermakova, Pavel S Novichkov, and Dmitry A Rodionov. Genomic reconstruction of transcriptional regulatory network in bacillus subtilis. *Journal of bacteriology*, pages JB–00140, 2013.

[23] Socorro Gama-Castro, Heladia Salgado, Alberto Santos-Zavaleta, Daniela Ledezma-Tejeida, Luis Muñiz-Rascado, Jair Santiago García-Sotelo, Kevin Alquicira-Hernández, Irma Martínez-Flores, Lucia Pannier, Jaime Abraham Castro-Mondragón, et al. Regulondb version 9.0: high-level integration of gene regulation, coexpression, motif clustering and beyond. *Nucleic acids research*, 44(D1):D133–D143, 2015.

[24] Nicolas Sierro, Yuko Makita, Michiel de Hoon, and Kenta Nakai. Dbtbs: a database of transcriptional regulation in bacillus subtilis containing upstream intergenic conservation information. *Nucleic acids research*, 36(suppl_1):D93–D96, 2007.

[25] Josch Pauling, Richard Röttger, Andreas Tauch, Vasco Azevedo, and Jan Baumbach. Coryneregnet 6.0—updated database content, new analysis methods and novel features focusing on community demands. *Nucleic acids research*, 40(D1):D610–D614, 2011.

[26] Sefa Kılıç, Elliot R. White, Dinara M. Sagitova, Joseph P. Cornish, and Ivan Erill. CollecTF: a database of experimentally validated transcription factor-binding sites in Bacteria. *Nucleic Acids Research*, 42(D1):D156–D160, jan 2014.

[27] Pavel S Novichkov, Alexey E Kazakov, Dmitry A Ravcheev, Semen A Leyn, Galina Y Kovaleva, Roman A Sutormin, Marat D Kazanov, William Riehl, Adam P Arkin, Inna Dubchak, et al. Regprecise 3.0–a resource for genome-scale exploration of transcriptional regulation in bacteria. *BMC genomics*, 14(1):745, 2013.

[28] Tanya Barrett, Stephen E Wilhite, Pierre Ledoux, Carlos Evangelista, Irene F Kim, Maxim Tomashevsky, Kimberly A Marshall, Katherine H Phillippy, Patti M Sherman, Michelle Holko, et al. Ncbi geo: archive for functional genomics data sets—update. *Nucleic acids research*, 41(D1):D991–D995, 2012.

[29] Helen Parkinson, Misha Kapushesky, Nikolay Kolesnikov, Gabriella Rustici, Mohammad Shojatalab, Niran Abeygunawardena, Hugo Berube, Miroslaw Dylag, Ibrahim Emam, Anna Farne, et al. Arrayexpress update—from an archive of functional genomics experiments to the atlas of gene expression. *Nucleic acids research*, 37(suppl_1):D868–D872, 2008.

[30] Jeremiah J Faith, Michael E Driscoll, Vincent A Fusaro, Elissa J Cosgrove, Boris Hayete, Frank S Juhn, Stephen J Schneider, and Timothy S Gardner. Many microbe microarrays database: uniformly normalized affymetrix compendia with structured experimental metadata. *Nucleic acids research*, 36(suppl_1):D866–D870, 2007.

[31] Marco Moretto, Paolo Sonego, Nicolas Dierckxsens, Matteo Brilli, Luca Bianco, Daniela Ledezma-Tejeida, Socorro Gama-Castro, Marco Galardini, Chiara Ro-

mualdi, Kris Laukens, et al. Colombos v3. 0: leveraging gene expression compendia for cross-species analyses. *Nucleic acids research*, 44(D1):D620–D623, 2015.

[32] Socorro Gama-Castro, Heladia Salgado, Martin Peralta-Gil, Alberto Santos-Zavaleta, Luis Muniz-Rascado, Hilda Solano-Lira, Veronica Jimenez-Jacinto, Verena Weiss, Jair S Garcia-Sotelo, Alejandra Lopez-Fuentes, et al. Regulondb version 7.0: transcriptional regulation of escherichia coli k-12 integrated within genetic sensory response units (gensor units). *Nucleic acids research*, 39(suppl_1):D98–D105, 2010.

[33] José P Faria, Ross Overbeek, Ronald C Taylor, Neal Conrad, Veronika Vonstein, Anne Goelzer, Vincent Fromion, Miguel Rocha, Isabel Rocha, and Christopher S Henry. Reconstruction of the regulatory network for bacillus subtilis and reconciliation with gene expression data. *Frontiers in microbiology*, 7:275, 2016.

[34] Lewis Y Geer, Aron Marchler-Bauer, Renata C Geer, Lianyi Han, Jane He, Siqian He, Chunlei Liu, Wenyao Shi, and Stephen H Bryant. The ncbi biosystems database. *Nucleic acids research*, 38(suppl_1):D492–D496, 2009.

[35] Eric Sayers. The e-utilities in-depth: parameters, syntax and more. *Entrez Programming Utilities Help [Internet]*, 2009.

[36] A Griffin and H Griffin. The european bioinformatics institute. *Molecular Biotechnology*, 1995.

[37] Christine Durinx, Jo McEntyre, Ron Appel, Rolf Apweiler, Mary Barlow, Niklas Blomberg, Chuck Cook, Elisabeth Gasteiger, Jee-Hyub Kim, Rodrigo Lopez, et al. Identifying elixir core data resources. *F1000Research*, 5, 2016.

[38] R D. C Team and R Development Core Team. R: A language and environment for statistical computing. *R Foundation for Statistical Computing*, 2016.

[39] Robert C Gentleman, Vincent J Carey, Douglas M Bates, Ben Bolstad, Marcel Dettling, Sandrine Dudoit, Byron Ellis, Laurent Gautier, Yongchao Ge, Jeff Gentry, et al. Bioconductor: open software development for computational biology and bioinformatics. *Genome biology*, 5(10):R80, 2004.

[40] Alex Rodriguez. Restful web services: The basics. *IBM developerWorks*, 33:18, 2008.

[41] Jinyu Wu, Fangqing Zhao, Shengqin Wang, Gang Deng, Junrong Wang, Jie Bai, Jianxin Lu, Jia Qu, and Qiyu Bao. ctfbase: a database for comparative genomics of transcription factors in cyanobacteria. *BMC genomics*, 8(1):104, 2007.

[42] Abel D Gonzalez, Vladimir Espinosa, Ana T Vasconcelos, Ernesto Perez-Rueda, and Julio Collado-Vides. Tractor_db: a database of regulatory networks in gamma-proteobacterial genomes. *Nucleic acids research*, 33(suppl_1):D98–D102, 2005.

[43] Sarah K Kummerfeld and Sarah A Teichmann. Dbd: a transcription factor prediction database. *Nucleic acids research*, 34(suppl_1):D74–D81, 2006.

[44] Shujiro Okuda and Akiyasu C Yoshizawa. Odb: a database for operon organizations, 2011 update. *Nucleic acids research*, 39(suppl_1):D552–D555, 2010.

[45] Denitsa Eckweiler, Christian-Alexander Dudek, Juliane Hartlich, David Brötje, and Dieter Jahn. Prodoric2: the bacterial gene regulation database in 2018. *Nucleic acids research*, 46(D1):D320–D326, 2017.

[46] Gavin Sherlock, Tina Hernandez-Boussard, Andrew Kasarskis, Gail Binkley, John C Matese, Selina S Dwight, Miroslava Kaloper, Shuai Weng, Heng Jin, Catherine A Ball, et al. The stanford microarray database. *Nucleic acids research*, 29(1):152–155, 2001.

[47] K Robison and G Church. Dpinteract: a database on dna–protein interactions, 1994.

[48] Pierre-Étienne Jacques, Alain L Gervais, Mathieu Cantin, Jean-Franc¸ois Lucier, Guillaume Dallaire, Geneviève Drouin, Luc Gaudreau, Jean Goulet, and Ryszard Brzezinski. Mtbreglist, a database dedicated to the analysis of transcriptional regulation in mycobacterium tuberculosis. *Bioinformatics*, 21(10):2563–2565, 2005.

[49] Deepak Sharma and Avadhesha Surolia. Computational tools to study and understand the intricate biology of mycobacteria. *Tuberculosis*, 91(3):273–276, 2011.

[50] Eduardo Pareja, Pablo Pareja-Tobes, Marina Manrique, Eduardo Pareja-Tobes, Javier Bonal, and Raquel Tobes. Extratrain: a database of extragenic regions and transcriptional information in prokaryotic organisms. *BMC microbiology*, 6(1):29, 2006.

[51] Alexei E Kazakov, Michael J Cipriano, Pavel S Novichkov, Simon Minovitsky, Dmitry V Vinogradov, Adam Arkin, Andrey A Mironov, Mikhail S Gelfand, and Inna Dubchak. Regtransbase—a database of regulatory sequences and interactions in a wide range of prokaryotic genomes. *Nucleic acids research*, 35(suppl_1):D407–D412, 2006.

[52] Xin Fang, Anand Sastry, Nathan Mih, Donghyuk Kim, Justin Tan, James T Yurkovich, Colton J Lloyd, Ye Gao, Laurence Yang, and Bernhard O Palsson. Global transcriptional regulatory network for Escherichia coli robustly connects gene expression to transcription factor activities. *Proceedings of the National Academy of Sciences of the United States of America*, 114(38):10286–10291, sep 2017.

[53] M. L. Arrieta-Ortiz, C. Hafemeister, A. R. Bate, T. Chu, A. Greenfield, B. Shuster, S. N. Barry, M. Gallitto, B. Liu, T. Kacmarczyk, F. Santoriello, J. Chen, C. D. Rodrigues, T. Sato, D. Z. Rudner, A. Driks, R. Bonneau, and P. Eichenberger. An experimentally supported model of the Bacillus subtilis global transcriptional regulatory network. *Molecular Systems Biology*, 11(11):839–839, nov 2015.

[54] Serdar Turkarslan, Eliza J R Peterson, Tige R Rustad, Kyle J Minch, David J Reiss, Robert Morrison, Shuyi Ma, Nathan D Price, David R Sherman, and Nitin S Baliga. A comprehensive map of genome-wide gene regulation in Mycobacterium tuberculosis. *Scientific Data*, 2(1):150010, dec 2015.

[55] Edgardo Galán-Vásquez, Beatriz Luna, and Agustino Martínez-Antonio. The Regulatory Network of Pseudomonas aeruginosa. *Microbial Informatics and Experimentation*, 1(1):3, jun 2011.

[56] JD Nelson, O Denisenko, K Bomsztyk Nature Protocols, and undefined 2006. Protocol for the fast chromatin immunoprecipitation (ChIP) method. *nature.com*.

[57] Robyn T. Eijlander, Anne de Jong, Antonina O. Krawczyk, Siger Holsappel, and Oscar P. Kuipers. SporeWeb: an interactive journey through the complete sporulation cycle of Bacillus subtilis. *Nucleic Acids Research*, 42(D1):D685–D691, jan 2014.

[58] Raphael H. Michna, Bingyao Zhu, Ulrike Mäder, and Jörg Stülke. SubtiWiki 2.0—an integrated database for the model organism Bacillus subtilis. *Nucleic Acids Research*, 44(D1):D654–D662, jan 2016.

[59] Richard Bonneau, David J Reiss, Paul Shannon, Marc Facciotti, Leroy Hood, Nitin S Baliga, and Vesteinn Thorsson. The Inferelator: an algorithm for learning parsimonious regulatory networks from systems-biology data sets de novo. *Genome Biology*, 7(5):R36, may 2006.

[60] Kyle J. Minch, Tige R. Rustad, Eliza J. R. Peterson, Jessica Winkler, David J. Reiss, Shuyi Ma, Mark Hickey, William Brabant, Bob Morrison, Serdar Turkarslan, Chris Mawhinney, James E. Galagan, Nathan D. Price, Nitin S. Baliga, and David R. Sherman. The DNA-binding network of Mycobacterium tuberculosi s. *Nature Communications*, 6(1):5829, may 2015.

[61] Tige R Rustad, Kyle J Minch, Shuyi Ma, Jessica K Winkler, Samuel Hobbs, Mark Hickey, William Brabant, Serdar Turkarslan, Nathan D Price, Nitin S Baliga, and David R Sherman. Mapping and manipulating the Mycobacterium tuberculosis transcriptome using a transcription factor overexpression-derived regulatory network. *Genome Biology*, 15(11):502, nov 2014.

[62] C. K. Stover, X. Q. Pham, A. L. Erwin, S. D. Mizoguchi, P. Warrener, M. J. Hickey, F.S. L. Brinkman, W. O. Hufnagle, D. J. Kowalik, M. Lagrou, R. L. Garber, L. Goltry, E. Tolentino, S. Westbrock-Wadman, Y. Yuan, L. L. Brody, S. N. Coulter, K. R. Folger, A. Kas, K. Larbig, R. Lim, K. Smith, D. Spencer, G. K.-S. Wong, Z. Wu, I. T. Paulsen, J. Reizer, M. H. Saier, R. E. W. Hancock, S. Lory, and M. V. Olson. Complete genome sequence of Pseudomonas aeruginosa PAO1, an opportunistic pathogen. *Nature*, 406(6799):959–964, aug 2000.

[63] The UniProt Consortium. UniProt: a worldwide hub of protein knowledge. *Nucleic Acids Research*, 47(D1):D506–D515, 11 2018.

[64] Brigitte Boeckmann, Amos Bairoch, Rolf Apweiler, Marie-Claude Blatter, Anne Estreicher, Elisabeth Gasteiger, Maria J Martin, Karine Michoud, Claire O'donovan, Isabelle Phan, et al. The swiss-prot protein knowledgebase and its supplement trembl in 2003. *Nucleic acids research*, 31(1):365–370, 2003.

[65] Dennis A. Benson, Mark Cavanaugh, Karen Clark, Ilene Karsch-Mizrachi, David J. Lipman, James Ostell, and Eric W. Sayers. GenBank. *Nucleic Acids Research*, 41(D1):D36–D42, nov 2012.

[66] Scott Federhen. The NCBI Taxonomy database. *Nucleic Acids Research*, 40(D1):D136–D143, 12 2011.

[67] Peter D Karp, Richard Billington, Ron Caspi, Carol A Fulcher, Mario Latendresse, Anamika Kothari, Ingrid M Keseler, Markus Krummenacker, Peter E Midford, Quang Ong, et al. The biocyc collection of microbial genomes and metabolic pathways. *Brief. Bioinformatics*, 2017.

[68] I Keseler, A Mackie, A Santos-Zavaleta, et al. The ecocyc database: reflecting new knowledge about escherichia coli k-12 nucleic acids res., 45. *D543–D550*, 11, 2017.

[69] Michael Hucka, Andrew Finney, Herbert M Sauro, Hamid Bolouri, John C Doyle, Hiroaki Kitano, Adam P Arkin, Benjamin J Bornstein, Dennis Bray, Athel Cornish-Bowden, et al. The systems biology markup language (sbml): a medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–531, 2003.

[70] Guido Van Rossum. Python programming language, usenix annual technical conference vol 41, 2007.

[71] Elizabeth Sheldon. The difference between orthologous and paralogous genes. *Sciencing*, 2018.

[72] M Madan Babu, Sarah A Teichmann, and L Aravind. Evolutionary dynamics of prokaryotic transcriptional regulatory networks. *Journal of molecular biology*, 358(2):614–633, 2006.

[73] Ross Overbeek, Michael Fonstein, Mark D'souza, Gordon D Pusch, and Natalia Maltsev. The use of gene clusters to infer functional coupling. *Proceedings of the National Academy of Sciences*, 96(6):2896–2901, 1999.

[74] Markus J Herrgård, Markus W Covert, and Bernhard Ø Palsson. Reconstruction of microbial transcriptional regulatory networks. *Current Opinion in Biotechnology*, 15(1):70 – 77, 2004.

[75] Qing Zhou and Jun S Liu. Modeling within-motif dependence for transcription factor binding site predictions. *Bioinformatics*, 20(6):909–916, 2004.

[76] Xin Chen, Lingqiong Guo, Zhaocheng Fan, and Tao Jiang. W-alignace: an improved gibbs sampling algorithm based on more accurate position weight matrices learned from sequence and gene expression/chip-chip data. *Bioinformatics*, 24(9):1121–1128, 2008.

[77] Kai Tan, Gabriel Moreno-Hagelsieb, Julio Collado-Vides, and Gary D Stormo. A comparative genomics approach to prediction of new members of regulons. *Genome research*, 11(4):566–584, 2001.

[78] Ziv Bar-Joseph, Georg K Gerber, Tong Ihn Lee, Nicola J Rinaldi, Jane Y Yoo, François Robert, D Benjamin Gordon, Ernest Fraenkel, Tommi S Jaakkola, Richard A Young, et al. Computational discovery of gene modules and regulatory networks. *Nature biotechnology*, 21(11):1337, 2003.

[79] Wynand BL Alkema, Boris Lenhard, and Wyeth W Wasserman. Regulog analysis: detection of conserved regulatory networks across bacteria: application to staphylococcus aureus. *Genome research*, 14(7):1362–1373, 2004.

[80] Pavel S Novichkov, Dmitry A Rodionov, Elena D Stavrovskaya, Elena S Novichkova, Alexey E Kazakov, Mikhail S Gelfand, Adam P Arkin, Andrey A Mironov, and Inna Dubchak. Regpredict: an integrated system for regulon inference in prokaryotes by comparative genomics approach. *Nucleic acids research*, 38(suppl_2):W299–W307, 2010.

[81] Lee Ann McCue, William Thompson, C Steven Carmack, Michael P Ryan, Jun S Liu, Victoria Derbyshire, and Charles E Lawrence. Phylogenetic footprinting of transcription factor binding sites in proteobacterial genomes. *Nucleic acids research*, 29(3):774–782, 2001.

[82] Hidde De Jong. Modeling and simulation of genetic regulatory systems: a literature review. *Journal of computational biology*, 9(1):67–103, 2002.

[83] Mukesh Bansal, Vincenzo Belcastro, Alberto Ambesi-Impiombato, and Diego Di Bernardo. How to infer gene networks from expression profiles. *Molecular systems biology*, 3(1):78, 2007.

[84] Tom Michoel, Riet De Smet, Anagha Joshi, Yves Van de Peer, and Kathleen Marchal. Comparative analysis of module-based versus direct methods for reverse-engineering transcriptional regulatory networks. *BMC systems biology*, 3(1):49, 2009.

[85] Mohamed Elati and Céline Rouveirol. Unsupervised learning for gene regulation network inference from expression data: A review. *Algorithms in Computational Molecular Biology*, pages 955–978, 2011.

[86] Lore Cloots and Kathleen Marchal. Network-based functional modeling of genomics, transcriptomics and metabolism in bacteria. *Current opinion in microbiology*, 14(5):599–607, 2011.

[87] Mika Gustafsson, Michael Hörnquist, Jesper Lundström, Johan Björkegren, and Jesper Tegnér. Reverse engineering of gene networks with lasso and nonlinear basis functions. *Annals of the New York Academy of Sciences*, 1158(1):265–275, 2009.

[88] Atul J Butte and Isaac S Kohane. Mutual information relevance networks: functional genomic clustering using pairwise entropy measurements. In *Biocomputing 2000*, pages 418–429. World Scientific, 1999.

[89] Jing Yu, V Anne Smith, Paul P Wang, Alexander J Hartemink, and Erich D Jarvis. Advances to bayesian network inference for generating causal networks from observational biological data. *Bioinformatics*, 20(18):3594–3603, 2004.

[90] Shoudan Liang, Stefanie Fuhrman, and Roland Somogyi. Reveal, a general reverse engineering algorithm for inference of genetic network architectures. 1998.

[91] Hendrik Hache, Christoph Wierling, Hans Lehrach, and Ralf Herwig. Reconstruction and validation of gene regulatory networks with neural networks. In *The 2nd Foundations of Systems Biology in Engineering Conference, FOSBE 2007*. Universität Stuttgart, 2007.

[92] Marco Grimaldi, Giuseppe Jurman, and Roberto Visintainer. Reverse engineering gene networks with ann: variability in network inference algorithms. *arXiv preprint arXiv:1009.4824*, 2010.

[93] Atul J Butte and Isaac S Kohane. Unsupervised knowledge discovery in medical databases using relevance networks. In *Proceedings of the AMIA Symposium*, page 711. American Medical Informatics Association, 1999.

[94] Jan Ihmels, Gilgi Friedlander, Sven Bergmann, Ofer Sarig, Yaniv Ziv, and Naama Barkai. Revealing modular organization in the yeast transcriptional network. *Nature genetics*, 31(4):370, 2002.

[95] David J Reiss, Nitin S Baliga, and Richard Bonneau. Integrated biclustering of heterogeneous genome-wide datasets for the inference of global regulatory networks. *BMC bioinformatics*, 7(1):280, 2006.

[96] Jeremiah J Faith, Boris Hayete, Joshua T Thaden, Ilaria Mogno, Jamey Wierzbowski, Guillaume Cottarel, Simon Kasif, James J Collins, and Timothy S Gardner. Large-scale mapping and validation of escherichia coli transcriptional regulation from a compendium of expression profiles. *PLoS biology*, 5(1):e8, 2007.

[97] Anagha Joshi, Riet De Smet, Kathleen Marchal, Yves Van de Peer, and Tom Michoel. Module networks revisited: computational assessment and prioritization of model predictions. *Bioinformatics*, 25(4):490–496, 2009.

[98] Alexandre Irrthum, Louis Wehenkel, Pierre Geurts, et al. Inferring regulatory networks from expression data using tree-based methods. *PloS one*, 5(9):e12776, 2010.

[99] Richard Bonneau, David J Reiss, Paul Shannon, Marc Facciotti, Leroy Hood, Nitin S Baliga, and Vesteinn Thorsson. The inferelator: an algorithm for learning parsimonious regulatory networks from systems-biology data sets de novo. *Genome biology*, 7(5):R36, 2006.

[100] Stephen F Altschul, Warren Gish, Webb Miller, Eugene W Myers, and David J Lipman. Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410, 1990.

[101] Atabak Mahram and Martin C Herbordt. Fast and accurate ncbi blastp: acceleration with multiphase fpga-based prefiltering. In *Proceedings of the 24th ACM International Conference on Supercomputing*, pages 73–82. ACM, 2010.

[102] Cheng Zhang and Qiang Hua. Applications of genome-scale metabolic models in biotechnology and systems medicine. *Frontiers in physiology*, 6:413, 2016.

[103] Minoru Kanehisa and Susumu Goto. Kegg: kyoto encyclopedia of genes and genomes. *Nucleic acids research*, 28(1):27–30, 2000.

[104] Daniel Machado, Rafael S Costa, Miguel Rocha, Eugénio C Ferreira, Bruce Tidor, and Isabel Rocha. Modeling formalisms in systems biology. *AMB express*, 1(1):45, 2011.

[105] Joachim Almquist, Marija Cvijovic, Vassily Hatzimanikatis, Jens Nielsen, and Mats Jirstrand. Kinetic models in industrial biotechnology–improving cell factory performance. *Metabolic engineering*, 24:38–60, 2014.

[106] Nathan D Price, Jason A Papin, Christophe H Schilling, and Bernhard O Palsson. Genome-scale microbial in silico models: the constraints-based approach. *Trends in biotechnology*, 21(4):162–169, 2003.

[107] George Dantzig. *Linear programming and extensions*. Princeton university press, 2016.

[108] Edward J O'Brien, Jonathan M Monk, and Bernhard O Palsson. Using genome-scale models to predict biological capabilities. *Cell*, 161(5):971–987, 2015.

[109] Edward J O'brien, Joshua A Lerman, Roger L Chang, Daniel R Hyduke, and Bernhard Ø Palsson. Genome-scale models of metabolism and gene expression extend and refine growth phenotype prediction. *Molecular systems biology*, 9(1):693, 2013.

[110] Markus W Covert, Christophe H Schilling, and Bernhard Palsson. Regulation of gene expression in flux balance models of metabolism. *Journal of theoretical biology*, 213(1):73–88, 2001.

[111] Sriram Chandrasekaran and Nathan D Price. Probabilistic integrative modeling of genome-scale metabolic and regulatory networks in escherichia coli and mycobacterium tuberculosis. *Proceedings of the National Academy of Sciences*, 107(41):17845–17850, 2010.

[112] Rogier JP van Berlo, Dick de Ridder, Jean-Marc Daran, Pascale AS Daran-Lapujade, Bas Teusink, and Marcel JT Reinders. Predicting metabolic fluxes using gene expression differences as constraints. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 8(1):206–216, 2011.

[113] Zineb El Akkaoui, Esteban Zimànyi, Jose-Norberto Mazón, and Juan Trujillo. A model-driven framework for ETL process development. In *Proceedings of the ACM 14th international workshop on Data Warehousing and OLAP - DOLAP '11*, page 45, New York, New York, USA, 2011. ACM Press.

[114] Per Christensson. Dbms definition. *Avialable at http://techterms.com/definition/dbms [accessed 01 June 2016]*, 2006.

[115] Kevin Loney. *Oracle Database 11g The Complete Reference*. McGraw-Hill, Inc., 2008.

[116] AB MySQL. Mysql, 2001.

[117] Chris J Date and Hugh Darwen. *A guide to the SQL Standard: a user's guide to the standard relational language SQL*. Addison-Wesley, 1989.

[118] Solid IT DB-Engines. Db-engines ranking, 2018.

[119] Edgar F Codd. Derivability, redundancy, and consistency of relations stored in large data banks. Technical report, IBM Research Report, 1969.

[120] Jing Han, E Haihong, Guan Le, and Jian Du. Survey on nosql database. In *Pervasive computing and applications (ICPCA), 2011 6th international conference on*, pages 363–366. IEEE, 2011.

[121] Josiah L Carlson. *Redis in action*. Manning Publications Co., 2013.

[122] Swaminathan Sivasubramanian. Amazon dynamodb: a seamlessly scalable non-relational database service. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 729–730. ACM, 2012.

[123] Ronald Bourret et al. Xml and databases. 1999.

[124] Kristina Chodorow. *MongoDB: The Definitive Guide: Powerful and Scalable Data Storage*. " O'Reilly Media, Inc.", 2013.

[125] Martin C Brown. *Getting Started with Couchbase Server: Extreme Scalability at Your Fingertips*. " O'Reilly Media, Inc.", 2012.

[126] Jeremy Kepner, Christian Anderson, William Arcand, David Bestor, Bill Bergeron, Chansup Byun, Matthew Hubbell, Peter Michaleas, Julie Mullen, David O'Gwynn, et al. D4m 2.0 schema: A general purpose high performance schema for the accumulo database. In *2013 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–6. IEEE, 2013.

[127] Guoxi Wang and Jianfeng Tang. The nosql principles and basic application of cassandra model. In *Computer Science & Service System (CSSS), 2012 International Conference on*, pages 1332–1335. IEEE, 2012.

[128] Diogo Fernandes and Jorge Bernardino. Graph databases comparison: Allegrograph, arangodb, infinitegraph, neo4j, and orientdb. In *DATA*, pages 373–380, 2018.

[129] Aleksa Vukotic Jonas Partner and Nicki Watt. Chapter 1. a case for a neo4j database, neo4j in action, 2014.

[130] Florian Holzschuher and René Peinl. Performance of graph query languages: comparison of cypher, gremlin and native access in neo4j. In *Proceedings of the Joint EDBT/ICDT 2013 Workshops*, pages 195–204. ACM, 2013.

[131] Ken Arnold, James Gosling, and David Holmes. *The Java programming language*. Addison Wesley Professional, 2005.

[132] Anders Hejlsberg, Scott Wiltamuth, and Peter Golde. *The C# programming language*. Adobe Press, 2006.

[133] Alan AA Donovan and Brian W Kernighan. *The Go programming language*. Addison-Wesley Professional, 2015.

[134] David Flanagan. *JavaScript: the definitive guide*. " O'Reilly Media, Inc.", 2006.

[135] Dave Raggett, Arnaud Le Hors, Ian Jacobs, et al. Html 4.01 specification. *W3C recommendation*, 24, 1999.

[136] Jeff Forcier, Paul Bissex, and Wesley J Chun. *Python web development with Django*. Addison-Wesley Professional, 2008.

[137] Clément Nedelcu. *Nginx HTTP Server: Adopt Nginx for Your Web Applications to Make the Most of Your Infrastructure and Serve Pages Faster Than Ever*. Packt Publishing Ltd, 2010.

[138] Hakon Wium Lie and Bert Bos. *Cascading style sheets: Designing for the web, Portable Documents*. Addison-Wesley Professional, 2005.

[139] Alexander Serganov, Anna Polonskaia, Anh Tuân Phan, Ronald R Breaker, and Dinshaw J Patel. Structural basis for gene regulation by a thiamine pyrophosphate-sensing riboswitch. *Nature*, 441(7097):1167, 2006.

[140] Eduard Torrents, Inna Grinberg, Batia Gorovitz-Harris, Hanna Lundström, Ilya Borovok, Yair Aharonowitz, Britt-Marie Sjöberg, and Gerald Cohen. Nrdr controls differential expression of the escherichia coli ribonucleotide reductase genes. *Journal of bacteriology*, 189(14):5012–5021, 2007.

[141] Matej Butala, Darja Žgur-Bertok, and Steve JW Busby. The bacterial lexa transcriptional repressor. *Cellular and Molecular Life Sciences*, 66(1):82, 2009.

[142] Shigeki Ehira, Haruhiko Teramoto, Masayuki Inui, and Hideaki Yukawa. Regulation of corynebacterium glutamicum heat shock response by the

extracytoplasmic-function sigma factor sigh and transcriptional regulators hspr and hrca. *Journal of bacteriology*, 191(9):2964–2972, 2009.

[143] Wade C Winkler, Smadar Cohen-Chalamish, and Ronald R Breaker. An mrna structure that controls gene expression by binding fmn. *Proceedings of the National Academy of Sciences*, 99(25):15908–15913, 2002.

[144] Robert W Cottingham. The doe systems biology knowledgebase (kbase): progress towards a system for collaborative and reproducible inference and modeling of biological function. In *Proceedings of the 6th ACM Conference on Bioinformatics, Computational Biology and Health Informatics*, pages 510–510. ACM, 2015.

A

SUPPORTING MATERIAL

A.1 *BLASTp* SEARCH RESULTS OF REGULATORS SHARING A COMMON NAME

Table S1.: Results of a BLASTp search for regulators found in several bacterial species. The search was performed using the UniProt BLASTp tool with default parameters against the *Bacteria Protein Knowledgebase*. *Query* stands for the UniProt accession and name of searched regulators. *Hit* represents the same information for found regulators. *E-value* stands for the number of expected hits of similar score that could be found randomly and the *Bit Score* is a metric for sequence similarity.

Query	**Hit**	**E-value**	**Bit Score**
P0ACJ8 (crp)	P0ACK1 (crp)	e-148	1,073
P0ACJ8 (crp)	A0A1E3MXN6 (crp)	e-148	1,073
P0ACJ8 (crp)	Q31VT1 (crp)	e-148	1,073
P0A8D0 (nrdR)	A0A331A9G4 (nrdR)	e-101	755
P0A8D0 (nrdR)	A0A1S9KG61 (nrdR)	e-101	755
P0A8D0 (nrdR)	A0A370V1G1 (nrdR)	e-101	755
P31080 (lexA)	A0A136GEF4 (lexA)	e-142	1 036
P31080 (lexA)	A0A0T8Q203 (lexA)	e-142	1 036
P31080 (lexA)	A0A2X1V0D8 (lexA)	e-142	1 036
P0A6D0 (argR)	Q31WA5 (argR)	e-103	770
P0A6D0 (argR)	A0A0M7P2E9 (argR)	e-103	770
P0A6D0 (argR)	B1EFG1 (argR)	e-103	766

Table S2.: Compendium of all node properties in the *TRN Collection* of ProTReND. Sources: (1) *RegulonDB*; (2) *DBTBS*; (3) *Collectf*; (4) *RegPrecise*; (5) *CoryneRegNet*; (6) *Fang et al.*; (7) *Faria et al.*; (8) *Arrieta-Ortiz et al.*; (9) *Turkarslan et al.*; (10) *Galán-Vásquez et al.*;

Label	**Property**	**Description**	**Example**	**Sources**
Organism	Name	*NCBI Taxonomy* name	Borreliella burgdorferi 297	All
Organism	Taxonomy ID	*NCBI Taxonomy ID*	511145	All
Organism	Source	First analyzed source	Collectf	All
Organism	Timestamp	Time and date of loading	2019-07-02 02:05:14	All
Organism	Genome Accession	First analyzed genome accession	NC_018983.1	(1,3)
Regulator	Name	Standardized name	bet	All
Regulator	Mechanism	Regulatory mechanism	Transcription Factor	All
Regulator	Source	First analyzed source	RegulonDB	All
Regulator	Timestamp	Time and date of loading	2019-07-02 09:03:21	All
Regulator	Comment	Function description	Senses *NADH:NAD* ratio	(7)
Regulator	Protein Complex	Boolean flag	True	(1)

(To be continued)

Label	Property	Description	Example	Sources
Regulator	Module	Biological function	Carbohydrate metabolism	(5)
Gene	Timestamp	Time and date of loading	2019-07-02 02:05:14	All
Gene	Gene Name	Standardized gene name	moeA	All
Gene	Source	First analyzed source	RegPrecise	All
Gene	Timestamp	Time and date of loading	2019-07-05 13:15:19	All
Gene	Is Regulator	Boolean flag	False	All
Gene	Gene locustag	Standardized locustag	BSU10300	(1,3,5,6,7,8)
Gene	Gene UniProt Accession	UniProt data	P12281	All
Gene	Gene Sequence	Protein sequence from UniProt	MRSKKLWIS...	All
Gene	Gene Synonyms	Aliases	narE JW081	All
Gene	Gene Operon	Standardized operon	aprE	(2,7,10)
Gene	Gene Start Position	Starting position in the genome	3145143	(1)
Gene	Gene End Position	Ending position in the genome	3146261	(1)
Gene	Start Codon	Standardized first codon	ATG	(1)

(To be continued)

Label	Property	Description	Example	Sources
Gene	Stop Codon	Standardized last codon	TAA	(1)
Gene	Product Type	Standardized product type	small RNA	(1)
Gene	Gene Strand	Standardized gene strand	forward	(1)
Gene	Product name	Standardized product name	L-lactate degydrogenase	(1)
Gene	Gene nucleotide sequence	Standardized sequence	ATGACTGGAG...	(1)
Gene	Gene module	Biological function	Carbohydrate metabolism	(5)
Binding site	Source	First analyzed source	CoryneRegNet	All
Binding site	Timestamp	Time and date of loading	2019-06-13 19:01:15	All
Binding site	Sequence	Nucleotide sequence	TACATACATTCA...	All
Binding site	Strand	DNA Strand	reverse	(1,2,3,5,6,7,8,9,10)
Binding site	Start	Starting position in genome	3145129	(1,2,3,5,6,7,8,9,10)
Binding site	End	Ending position in the genome	3145143	(1,2,3,5,6,7,8,9,10)

(To be continued)

Label	Property	Description	Example	Sources
Binding site	Evidence	Experimental evidence	[BCE | Binding of cellular extracts]	(1)
Binding site	Evidence Confidence Level	Binary degree of confidence	Strong	(1)
Binding site	Center position	Center position relative to transcription start site	22.5	(1)
Binding site	Protein Complex	Boolean flag	False	(1)

Table S3.: Compendium of all edge properties in the *TRN Collection* of ProTReND. Sources: (1) *RegulonDB*; (2) *DBTBS*; (3) *Collectf*; (4) *RegPrecise*; (5) *CoryneRegNet*; (6) *Fang et al.*; (7) *Faria et al.*; (8) *Arrieta-Ortiz et al.*; (9) *Turkarslan et al.*; (10) *Galán-Vásquez et al.*;

Label	**Property**	**Description**	**Example**	**Sources**
Binding Site in Organism	Source	First analyzed source	RegulonDB	All
Binding Site in Organism	Timestamp	Time and date of loading	2019-06-13 19:59:35	All
Regulator in Organism	Source	First analyzed source	Collectf	All
Regulator in Organism	Timestamp	Time and date of loading	2019-07-02 02:05:14	All
Regulator in Organism	UniProt Accession	UniProt data	P0ACF8	All
Regulator in Organism	Sequence	Protein sequence	MDALSRLLMLN...	All
Regulator in Organism	Synonyms	Aliases	gutR b2707	All
Regulator in Organism	SwissProt	SwissProt identifier	O34723	(2)
Regulator in Organism	Factor type	Regulator ontology	TetR family	(2)
Regulator in Organism	Subtilist	Subtilist identifier	BG14133	(2)
Regulator in Organism	Comment	Biological function	Regulation of manganese transport	(2)

(To be continued)

Label	Property	Description	Example	Sources
Regulator regulates Gene	Source	First analyzed source	DBTBS	All
Regulator regulates Gene	Timestamp	Time and date of loading	2019-06-09 14:26:55	All
Regulator regulates Gene	Regulatory effect	Activation, repression or dual	Repression	All
Regulator regulates Gene	Sigma Factor	Involved sigma factor	sigA	(2,7)
Regulator regulates Gene	Involved Effectors	Metabolites or stimuli	Galacturonate	(7)
Regulator regulates Gene	Effector effect	Effect on regulation	Inactivation	(7)
Regulator regulates Gene	Conditioned rules	Rules for regulation	LexA + RecA	(7)
Regulator regulates Gene	Evidence	Experimental or *in silico*	[GEA, HIBSCS]	(1,5,6,10)
Regulator regulates Gene	Evidence Confidence	Weak, strong or inferred	Strong	(1,6)
Regulator regulates Gene	Expression log2	log2 value of gene expression	-1.63	(9)
Regulator regulates Gene	Relative Regulatory Effect	Relative regulation based on percentile distribution	Strong repression	(9)
Regulator regulates Gene	PubMed reference	Study identifier	19930444	(5,10)