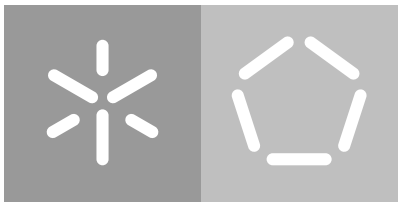


Universidade do Minho
Escola de Engenharia
Departamento de Informática

José Amaro Ferreira Morais

Development of bioinformatics tools to accelerate the reconstruction of genome-scale models

November 2018



Universidade do Minho
Escola de Engenharia
Departamento de Informática

José Amaro Ferreira Morais

Development of bioinformatics tools to accelerate the reconstruction of genome-scale models

Master dissertation
Master Degree in Bioinformatics

Dissertation supervised by
Oscar Manuel Lima Dias

November 2018

AGRADECIMENTOS / ACKNOWLEDGEMENTS

A tese aqui apresentada foi resultado de 1 ano de trabalho, esforço e dedicação. Quero por isso referir que todo este trabalho não seria possível sem todas aquelas pessoas que me apoiaram e me ajudaram em tudo o que precisava, sendo estes agradecimentos dirigidos para elas.

Quero começar por agradecer ao meu orientador, Dr. Oscar Dias , que me proporcionou a oportunidade de realizar este projeto, que me orientou durante todo este percurso, mantendo sempre um acompanhamento próximo e uma disponbilidade excepcional, e que contribui para a minha aprendizagem e para o meu desenvolvimento como estudante e como pessoa.

Dou também um agradecimento especial ao Davide, com quem passei grande parte do tempo deste projeto, por me ter ajudado sempre que precisei, pela companhia e por ter sido um bom parceiro de trabalho e um bom amigo.

Quero agradecer também aos meus pais e à minha irmã, que foram o meu pilar nesta etapa, como têm sido em todas. Obrigado pelo apoio, por me darem motivação, confiança e segurança, que permitiram que tudo isto fosse alcançado.

Agradeço também aos meus amigos por me ajudarem a abstrair quando mais precisei, pelas saídas à noite, pelos quizes, pelos almoços e jantares, pelas tardes muito bem passadas no Pão de Forma.

Agradeço também ao pessoal do Biosystems pelas reuniões, pelos almoços e por estarem disponiveis sempre que precisei.

Acabo por agradecer ao Centro de Engenharia Biológica e à Universidade do Minho, que me permitiram alargar o meu conhecimento e experiência nesta área da Bioinformática, disponibilizando todos os recursos necessários durante os dois anos de mestrado.

ABSTRACT

Title: Development of bioinformatics tools to accelerate the reconstruction of genome-scale models.

Genome-scale metabolic (*GSM*) models are mathematical tools, which can be used to evaluate and manipulate organisms' phenotypes *in silico*, thus predicting their responses to different genetic or environmental conditions. Therefore, *GSM* models can have a leading role in medical research, together with other systems biological approaches, for drug targeting and discovery or vaccine development.

Metabolic Models Reconstruction Using Genome-Scale Information (*merlin*) is an open source framework implemented in JAVA that aims at reconstructing *GSM* models. Even though *merlin* allows the reconstruction of genome-scale metabolic models for any organism, whose genome has been sequenced, as well as the functional annotation of its genome, it has shortcomings. Namely, it lacks a tool able of automatically generating draft reconstructions, from reference well-curated models. Therefore, the main goal of this project was the implementation of new bioinformatics tools in *merlin*, as well as the improvement of other functionalities. This new feature will expedite the reconstruction of genome-scale metabolic models, providing alternatives to the traditional and time-consuming reconstruction process.

Hence, *merlin*'s main upgrades and side improvements are described in detail in this work. It includes the development and implementation of tools that allow importing models in the SBML format and automatically generate draft reconstructions from curated models. Furthermore, a draft reconstruction of a *Treponema pallidum* (str. Nichols) model using *merlin* was performed to assess the new tools. This draft was compared with a recently published draft and an automatically generated KBase's draft model.

Keywords: *merlin*; Genome-Scale Metabolic models, Bioinformatics tools, *Treponema pallidum*;

RESUMO

Título: Desenvolvimento de ferramentas de bioinformáticas para acelerar a reconstrução de modelos à escala genómica

Modelos Metabólicos à Escala Genómica (*MEG*) são ferramentas matemáticas que podem ser usadas para avaliar e manipular fenotipicamente organismos *in silico*, prevendo a sua resposta a diferentes condições genéticas ou ambientais. Portanto, modelos *MEG*, juntamente com outras abordagens biológicas de sistemas, podem desempenhar um papel importante na investigação médica, tanto na descoberta de novos alvos para drogas terapêuticas como na descoberta ou desenvolvimento de vacinas.

O "Metabolic Models Reconstruction Using Genome-Scale Information" (*merlin*) é uma ferramenta "open-source" implementada em JAVA que visa a reconstrução de modelos *MEG*. Apesar do *merlin* permitir a reconstrução de modelos metabólicos em escala genómica para qualquer organismo, cujo genoma tenha sido sequenciado, bem como a anotação funcional de seu genoma, ele apresenta limitações. Mais concretamente, ele não tem ferramentas capazes de gerar automaticamente rascunhos de reconstruções de modelos, a partir de modelos de referência bem curados. Portanto, o principal objetivo deste projeto foi a implementação de novas ferramentas bioinformáticas no *merlin*, bem como o aprimoramento de outras funcionalidades. Estes novos recursos acelerarão a reconstrução de modelos *MEG*, fornecendo assim alternativas ao demorado processo de reconstrução tradicional.

Assim, as principais atualizações e melhorias subjacentes do *merlin* serão descritas em detalhes neste trabalho. Estas incluem o desenvolvimento e a implementação de ferramentas que permitem importar modelos no formato SBML e gerar automaticamente rascunhos de reconstruções de modelos a partir de outros já bem curados. Além disso, uma versão não curada da reconstrução do modelo da *Treponema pallidum* (str. Nichols) foi obtida usando o *merlin* de modo a avaliar as novas ferramentas. Esta reconstrução foi comparada com uma reconstrução curada publicada recentemente e com um rascunho gerado na plataforma online KBase.

Palavras-chave: *merlin*, Modelos Metabólicos à Escala Genómica, Ferramentas bioinformáticas, *Treponema pallidum*;

CONTENTS

1 INTRODUCTION 1
1.1 Context 1
1.2 Objectives 2
1.3 Document Structure 3
2 STATE OF THE ART 4
2.1 Systems Biology 4
2.1.1 *Omics* 5
2.2 Genome-scale Metabolic Models 5
2.2.1 Genome Annotations 6
2.2.2 Draft Network Reconstruction 6
2.2.3 Conversion into Stoichiometric Model 9
2.2.4 Model Evaluation and Validation 11
2.3 GSM models' reconstruction tools 12
2.3.1 Model SEED 12
2.3.2 RAVEN Toolbox 13
2.3.3 SuBliMinal Toolbox 13
2.3.4 Pathway Tools 14
2.3.5 *merlin* 15
2.3.6 Other tools 16
2.4 GSM models' simulation tools 18
2.4.1 COBRA Toolbox 18
2.4.2 OptFlux 19
2.4.3 CellNetAnalyzer 19
2.5 Other Bioinformatics Tools 20
2.5.1 KBase 20
2.6 Java Application Frameworks 20
2.6.1 AIBench 20
2.6.2 Hibernate 21
2.7 *Treponema pallidum* 21
2.7.1 Subspecies *pallidum* 22
2.7.2 Genome sequencing and analysis 22
3 METHODS AND TOOLS 25
3.1 Relational Databases 25
3.1.1 MySQL 25

3.1.2 H2 25
3.1.3 The Java Database Connectivity API 26
3.2 AIBench framework 26
3.2.1 Operations 27
3.2.2 Datatypes 27
3.2.3 Views 27
3.2.4 Graphical User Interface 28
3.3 NCBI Entrez Programming Utilities 28
3.3.1 E-Search 28
3.3.2 E-Summary 28
3.4 Metabolic Engineering Workbench Biocomponents 29
3.5 ModelSEED Database Files 29
3.6 Basic Logical Alignment Search Tool 30
3.6.1 BLAST+ v2.7.1 30
3.6.2 Java Architecture for XML Binding 33
3.7 Systems Biology Markup Language (SBML) Level 3 33
3.7.1 "Identifiers.org" Uniform Resource Identifiers 34
3.7.2 JSBML library 34
3.8 Reference Genome-Scale Metabolic (*GSM*) models, genomes and selection criteria 35
3.9 Clustal Omega 35
3.10 KBase apps 36
3.11 Biomass Assembly 36
3.12 Supplementary files 37
3.13 Third-party tools and licenses 37
4 SOFTWARE DEVELOPMENT 39
4.1 Genome Sequence and Annotation Data Files Automatic Download 39
4.1.1 Searching and retrieving Assembly's records 40
4.1.2 Graphical Component 41
4.1.3 "DownloadNcbiFiles" Operation 42
4.2 SBML Models Importer 43
4.2.1 Reading and processing SBML model data 43
4.2.2 Loading model data into database 46
4.2.3 '*ImportSbmlModel*' Operation 47
4.3 Integrating NCBI BLAST+ into *merlin* 47
4.3.1 NCBI BLAST Parsing 48
4.3.2 BlastAlignment Class 48
4.3.3 merlin's Similarity Searches to use BLAST 50

4.4 Homology-based Draft Models Reconstruction Tool 51
4.4.1 Processing and retrieving subject model's data 52
4.4.2 Implementing homology-based reconstruction pipeline 53
4.5 SBML Level 3 Exporter 53
4.5.1 Retrieving model's data from database 55
4.5.2 Building the SBML level 3 document 55
4.5.3 SBML Level 3 Exporter Operation 59
4.6 Other *merlin*'s features improvement 59
4.6.1 Improving FASTA files processing and uploading 59
4.6.2 Genes view improvement 60
4.6.3 e-Biomass Support of ModelSEED's Compounds 61
5 RESULTS AND DISCUSSION 62
5.1 *merlin*'s improved architecture 62
5.2 Validating SBML model's importer and SBML level 3 exporter 63
5.3 *Treponema pallidum* draft reconstruction 65
5.3.1 Selection of the reference models 65
5.3.2 Draft reconstruction output 66
5.3.3 Biomass Assembly 69
6 CONCLUSIONS 70
6.1 Limitations 70
6.2 Prospect for future work 71

A SUPPORT MATERIAL 83

LIST OF FIGURES

Figure 1	Gene-protein-reaction (GPR) associations.	8
Figure 2	*merlin*'s architecture.	16
Figure 3	XML BLAST Output.	32
Figure 4	"New Project" GUI new layout.	41
Figure 5	"Download NCBI Files" GUI layout.	42
Figure 6	*merlin*'s SBML models' impoter.	44
Figure 7	SBML level 3 structure.	58
Figure 8	Improved "genes" view.	60
Figure 9	*merlin*'s improved architecture.	62
Figure 10	SBML Level 3 "GeneProduct" component.	64
Figure 11	SBML Level 3 "Reaction" component.	65
Figure 12	Clustal Omega Neighbour-joining phylogenetic tree.	66
Figure 13	Pie chart representing the relative frequency, in percentage, of reactions by pathway, in the merlin's draft reconstruction.	68

LIST OF TABLES

Table 1	Genomic-scale metabolic model reconstruction tools overview.	17
Table 2	A small overview over the different BLAST variants.	30
Table 3	*GSM* models and genomes used during this project.	35
Table 4	Third-party tools used in this work.	38
Table 5	Origin of the genes, reactions and gene-reaction associations data incorporated in the *T. pallidum* draft reconstruction.	67
Table 6	Comparison of three *T. pallidum* draft model reconstructions.	67
Table 7	Biomass composition for the draft reconstruction of *T. palliudm*.	69
Table S1	Clustal Omega alignments result identity matrix.	83

ACRONYMS

GSM Genome-Scale Metabolic

MERLIN Metabolic Models Reconstruction Using Genome-Scale Information

AIBENCH Artificial Intelligence workBENCH

API Application Programming Interface

BLAST Basic Logical Alignment Search Tool

COBRA Constraint-Based Reconstruction and Analysis

CORECO Comparative Reconstruction

E-UTILITIES Entrez Programming Utilities

E-VALUE Expected value

EC Enzyme Commission

EFM Elementary Flux Mode

FBA Flux Balance Analysis

FBC Flux Balance Constraints

FVA Flux Variability Analysis

GEMSIRV Genome-scale Metabolic Model Simulation, Reconstruction and Visualization

GFBA geometric FBA

GIMME Gene Inactivity Moderated by Metabolism and Expression

GO gene ontology

GPR Gene-protein-reaction

GUI Graphical User Interface

HSP High Scoring Pair

IMAT integrative Metabolic Analysis Tool

JAXB Java Architecture for XML Binding

JDBC Java Database Connectivity

JPA Java Persistence API

KBASE Department of Energy Systems Biology Knowledgebase

KEGG Kyoto Encyclopedia of Genes and Genomes

MBA Model-Building Algorithm

MEMOSYS Metabolic Model research and development System

MEWORKBENCH Metabolic Engineering Workbench

MFA Metabolic Flux Analysis

MIRIAM Minimal Information Required In the Annotation of Models

MOMA Minimization Of Metabolic Adjustment

MVC Model-View-Controller

NCBI National Center for Biotechnology Information

ORF Open Reading Frame

ORM Object/Relational Mapping

RAST Rapid Annotation using Subsytems Technology

RAVEN Reconstruction, Analysis and Visualization of Metabolic Networks

RDBMS Relational Database Management System

ROOM Regulatory On/Off Minimization

SB Systems Biology

SBML Systems Biology Markup Language

SQL Structured Query Language

TC Transporter Classification

URI Uniform Resource Identifier

XML eXtensible Markup Language

1

INTRODUCTION

1.1 CONTEXT

Systems Biology (SB) has proven to be a rising field in the past few decades in what concerns biological systems study. It provides a systematic study not only of the organism's components individually but also the interactions between them. SB allows connecting biochemical data from different biological areas, such as molecular biology and metabolic engineering, extracting biological meaning from this data. These factors combined, allow comprehending and predicting the phenotypical behaviour on conditions other than ones already characterized [71].

In the past few years, metabolic SB has become a standard field of study with its *GSM* models leading an important role in the advances on metabolic engineering, industrial engineering and biotechnology industry, as well as providing insights on organisms' metabolic mechanisms, multi-species relationships and evolutionary relationships [43, 49, 55, 91]. A *GSM* model is an organism-specific structured collection of biochemical reactions and associated genes obtained from the annotated genome sequence, biochemical information and primary literature [36, 54]. These models are mathematical representations of an organism's metabolism and enable evaluating and manipulating its phenotype *in silico*, thus predicting the organisms' responses to different genetic or environmental conditions [36, 54]. *GSM* models are reconstructed based on the parts of the genome that encode metabolic functions. Hence, a solid and reliable genome functional annotation is mandatory.

Metabolic Models Reconstruction Using Genome-Scale Information (*merlin*) is a JAVA® application that allows reconstructing *GSM* models, for any organism whose genome has been sequenced, including the functional genomic annotation of its entire genome, and exporting of the final reconstruction of the draft model in the SBML standard format [37]. Some prominence can be given to *merlin* as it allows the reconstruction of metabolic models for eukaryotic organisms. Furthermore, it is also recognized for its interactive and intuitive curation panels displayed to perform manual curation at any point of the reconstruction process. However, *merlin* has shortcomings too. It lacks tools to enable an automatic and fast generation of draft genome-scale models. Thus, introducing *merlin* to a more automatic

concept of reconstructing models thus providing alternatives to the traditional and time-consuming reconstruction process, may be a valuable upgrade on this software. It can be accomplished by the implementation of new bioinformatics tools in merlin, as well as the improvement of its current ones. These *merlin* upgrades can then be used to build a draft reconstruction for *Treponema pallidum* (str. Nichols).

Treponema pallidum (str. Nichols) (*T. pallidum*) is a very small motile spirochete that causes the syphilis disease, that can be either sexually transmitted or passed from mother to unborn child. *T. pallidum* are gram-negative bacteria, thus having an inner membrane (a thin peptidoglycan cell wall) and an outer membrane [102, 103]. The complete genome for *T. pallidum* was sequenced in July of 1998. It consists of a single double-stranded circular DNA chromosome 1,138,006 base pairs long, one of the smallest bacterial genomes. It contains approximately 1,090 genes, which encode approximately 1,041 proteins [46]. *T. pallidum* is a pathogenic organism that is responsible for the sexually transmittable syphilis disease, and while a lot of effort has been put into its containment and elimination, there are still occasional reported resurgence of cases, even in developed countries. An unusual feature of this bacterium relies in the extremely low density of proteins in its outer membrane, decreasing the number of immune targets on its outer surface, thus making the discovery of an effective treatment difficult for this type of disease [46, 103]. Due to this fact, a deeper study on this organism metabolic system, its components and the interactions between them, as well as predict its behaviour in response to external changes might be essential to strike this unpleasant organism. As to the best of our knowledge both a proper annotation and a final and curated *GSM* model for *T. pallidum* are still not available.

1.2 OBJECTIVES

The main goal of this work is the development of new *merlin*'s functionalities, as well as the improvement of existing ones, to expedite and accelerate *GSM* models' reconstruction process. These improvements will be used to perform the reconstruction of a draft model for the dairy bacterium *Treponema pallidum* subsp. *pallidum* str. Nichols.

In detail, we aim to perform the following approaches:

- Apply state-of-the-art for Bioinformatics approaches to get insights on the current state of metabolic modelling and its available tools, as well as on the bacterium genome;
- Development and implementation of new tools to expedite the reconstruction of models in *merlin*;
- Generate a draft genome-scale metabolic model for *T. pallidum* str. Nichols.

1.3 DOCUMENT STRUCTURE

This document is organized as follows:

Chapter 2: State of the Art
In the second chapter of this document, a review on the "state of the art" is presented. Starting with a brief introduction to Systems Biology and "its" *omics*, this chapter soon focus on *GSM* models, describing their genesis, characteristics, reconstruction process, importance and possible applications. It follows with an exposition of some *GSM* model reconstruction and simulation tools, as well as application frameworks used in the developing of some of them, or able to be used in the development of new ones. This section ends with a description and characterization of *T. pallidum*.

Chapter 3: Methods and Tools
This chapter presents a description of all approaches, algorithms, bioinformatics tools and Java libraries used in *merlin*'s improvements and new features development.

Chapter 4: Software Development
In this chapter a detailed description of *merlin* improvements and new functionalities, as well as a detailed display of their implementation will be performed.

Chapter 5: Results and Discussion
This chapter comprises the assessment on the implemented tools. It is also presented the draft reconstruction obtained for the bacterium. Critical analysis to the outcome, confronting the results with data present in literature and other drafts of the organism, as well as possible strategies to be applied in a follow up of this work.

Chapter 6: Conclusions
Main conclusions of the work accomplished for this thesis followed by a description of its limitations and possible improvements and future work.

2

STATE OF THE ART

2.1 SYSTEMS BIOLOGY

Complexity and variability are keywords in the characterization of ecosystems and the organisms who live in them. In fact, despite their level of complexity, most living organisms around the planet are complex systems comprising a large set of chemical reactions, genetic material, metabolic pathways and biological compounds interacting with each other with coherent behaviours [69, 71]. Systems Biology (SB) is a biology field that focuses on the systematic study of complex interactions in these biological systems [69]. This is particularly important because, despite the large knowledge on biological facts provided by molecular biology, such as genome sequences and protein properties, the interpretation of biological systems requires understanding the biological interactions at a system-level [71]. This association was firstly preformed in 1915 by Walter Cannon [26], but only in 1950 Ludwig von Bertalanffy proposed that "... organismic conceptions have evolved in all branches of modern biology which assert the necessity of investigating not only parts but also relations of organization resulting from a dynamic interaction and manifesting themselves by the difference in behavior of parts in isolation and in the whole organism." [21]. SB gained particular relevance and emerged in the last decade due the evident progress in molecular biology, particularly in genome sequencing and high-throughput measurements, which provide large collections of comprehensive data sets on system performance and a considerable amount of information on its molecules and mechanisms [71]. Thus, this discipline is based on principle of giving large-scale data sets, analysing the components and the interactions between them, to understand and to predict biological behaviour on conditions other than ones already characterized [36, 69]. Insights into the functioning of biological systems will then rely in a combination of experimental and computational approaches in a computer model, in order to solve these complex problems [70]. This approach is still a major challenge of SB when trying to represent the whole-cell behaviour in a computer model. Despite this challenge, a good model is not just the one who is able to mimic cell comportment but the one who predicts such behaviour. Thus, these models should be able to foretell the phenotypical behaviour of a cell, an organism, or an individual [36].

2.1.1 *Omics*

SB uses technologies such as genomics, transcriptomics, proteomics and metabolomics, to determine the state of the system. These omics generate large amounts of data, which cannot be understood without a network or systems viewpoint and highly sophisticated computational analyses [32].

Genomics is a field of SB dedicated to the study of the genome of organisms, which allows retrieving the case study organism's full genome sequence.

Whereas, in Transcriptomics the focus is on the transcriptome of organisms. Determining a transcriptome involves using techniques, such as next-generation RNA sequencing methods or microarray technology (DNA oligonucleotide and cDNA arrays). Both of which can measure the mRNA expression levels for a given condition of virtually every Open Reading Frame (ORF) in the genome [36, 99].

Proteomics focus on the study of the proteome of organisms. The proteome consists on all proteins in the cell, like enzymes, transport proteins, regulatory proteins, signalling proteins, and others. Although proteins are encoded in the genome, there is no known linear correlation between the quantity of mRNA molecules and the corresponding proteins [36, 52, 53].

Finally, Metabolomics is the study of the metabolome of organisms. The metabolome comprises all available metabolites in the system. The importance in quantifying the metabolites availability resides in the fact these metabolites are responsible for the connectivity of the network [36].

2.2 GENOME-SCALE METABOLIC MODELS

In the past few decades the advances in SB and several *omics* have been considerable. The first *GSM* reconstruction to be published dates back to 1999 with the metabolic model of *Haemophilus influenzabolic* [41, 85], a few years after the publication of its full genome sequencing [44, 91]. Since then large-scale *omics* experiments are now standard in many biological studies. With modern genome-sequencing capabilities, metabolic pathways have been increasingly integrated into *GSM* models [91]. *GSM* models are now valuable SB platforms for model-guided data analysis of large *omics* datasets, because they can offer cellular context to the data [60]. One key aspect on *GSM* models is that they catalog all metabolic reactions in an organism. This allows integrating diverse omics data, as these models can directly associate metabolites with enzymes [94]. Furthermore, these models connect genes to measurable cell phenotypes, thus being able to show how changes in one component affects other pathways and cell phenotypes, such as growth, cell energetics, biosynthesis, etc. Therefore, these models can provide a mechanistic link from genotype to phenotype [94].

The information contained in these models includes details on the biomass composition and energetic requirements, other than network data. These models are currently being used to predict, *in silico*, the response to perturbations of microorganisms and to identify candidate drug targets [36].

Four major steps can be highlighted in a *GSM* model reconstruction [36, 92]. They are the attainment of the annotated genome, the draft network reconstruction, conversion of the reconstruction to mathematical model (stoichiometric model), and the model evaluation and validation [36]. However, these four major steps can be sub-divided in a larger workflow [114]. Nevertheless, the process of reconstructing a model, regarding its main steps division, is not rigid and can be divided in different ways from author to author. These procedures together with appropriate *GSM* models' reconstruction and simulation tools [54], can be applied to obtain a reliable and consistent *GSM* model.

2.2.1 *Genome Annotations*

The first step in this reconstruction is to obtain the genome and annotations, since the *GSM* model reconstruction requires the annotated genome of the organism in study [36]. Many genomes are already annotated. These annotated genomes can be retrieved from public online databases such as the National Center for Biotechnology Information (NCBI) [33], the Kyoto Encyclopedia of Genes and Genomes (KEGG) [64] or the SEED [95] which has stored annotations performed by the Rapid Annotation using Subsytems Technology (RAST) [18]. A curated genome annotation provides a good basis for reconstructing *GSM* models. This is a critical step because it is assumed that the genome annotation is correct, and this step should, ideally, be performed only once. Nevertheless, a reannotation of the genome is mandatory if the existing annotation is not traceable and reliable [36, 114]. The genome annotation process assigns genes with functions, providing unique identifiers, such as the Enzyme Commission (EC) [61] and Transporter Classification (TC) [105] numbers, to the reconstruction. Subunits of protein complexes should also be identified during this process, as more than one gene may be necessary to encode an enzyme. Genes encoding enzymes or transport systems can be labeled as metabolic genes [36]. A curated genome annotation provides a good basis for reconstructing *GSM* models, so its manual curation seems to be mandatory in order to get a reliable and cohesive model.

2.2.2 *Draft Network Reconstruction*

However, metabolic information is also required for developing these models and that is where the assembly of the *GSM* network (second step) begins. This step involves retrieving the reactions promoted by the proteins encoded in the genes (gene-protein-reaction (GPR)

associations) [36]. After identifying the metabolic genes in the genome annotation, using for example enzyme names, EC number or gene ontology (GO), metabolic reactions catalyzed by the products of those genes can be integrated in the draft model [114]. This step involves using biochemical reaction databases, such as KEGG, BRENDA [109], MetaCyc [27], MetRxn [76], BIGG [107] or existing models, available through databases such as BioModels DB [79] or the BioMet Toolbox v2.0 [47]. The *GSM* network should then be manually curated, using literature and databases with reviewed information. This step consists on the evaluation and refinement of the draft model to ensure consistency with physical principles and experimental evidence. The inclusion of each reaction should be carefully scrutinized and GPR associations validated, as not all annotations have a high confidence score and biochemical databases are mostly organism unspecific [114]. The inclusion of other reactions, mostly organism unspecific ones, may affect the predictive behavior of the resulting models [114]. Several steps are applied in this stage to curate the model, such as the verification of metabolic functions supported on experimental data and literature as well as using phylogenetically close organisms (if no organism-specific information can be found in literature). Another step is to exclude certain reactions from the draft model, because some have generic reaction terms such as DNA or protein, as these are not important for these models. Organism-specific reactions, unavailable in the queried databases, should also be included in the model [36], which may be useful for filling gaps in the metabolic network. A gap in a model refers to a missing reaction in a pathway, as this is uncoupled to a gene. Such reaction will not be available in the network, which will lead to accumulation of compounds produced by energetically favored reactions [36]. One approach to find these gaps is to identify metabolic dead ends, metabolites that cannot be produced or consumed. These metabolites can point to gaps, or missing reactions, in the network which may need to be filled [54]. Other alternative is to use KEGG pathways, which is a collection of manually drawn pathway maps representing molecular interaction and reaction networks [36].Since most databases present low organism-specificity, enzymes substrate and cofactor usage have to be carefully reviewed. For example, certain enzymes, used in several reactions, have multiple substrates and cofactors depending on the reaction they catalyze [114]. Charged formula for each metabolite (protonated or deprotonated) as well as metabolite identifiers are also of particular relevance. Metabolites and reactions should be associated with at least one of the following identifiers: ChEBI [24], KEGG and PubChem [33] as these should be identifiable by the SB community software tools. Reactions' stoichiometry, directionality and reversibility are other aspects to take into consideration. Reactions reversibility may be assessed by biochemical studies of enzymes of the target organism or closely related organisms, or from the estimation of the standard Gibbs free energy of formation ($\Delta_f G'^0$) and of reaction ($\Delta_r G'^0$) [36].

One important point in curation is the information for proteins and reactions localization, but this information is not easy to obtain. The use of algorithms such as PSORT [48], LocTree3 [88], WoLF-PSORT [57] and PASUB [83] may solve this problem, as these can predict the cellular localization of proteins based on nucleotide or amino acid sequences [114].

Regarding GPR associations, despite information about these associations being often provided by the genome annotation, some validation is required. Thus, it is mandatory to assess whether functional proteins are a heteromeric enzyme complexes, if enzymes can promote several reactions or if more than one protein can carry out the same function [114]. This complexity, can be visualized in figure 1 with a simple and graphic representation of the associations between genes, proteins and reactions.

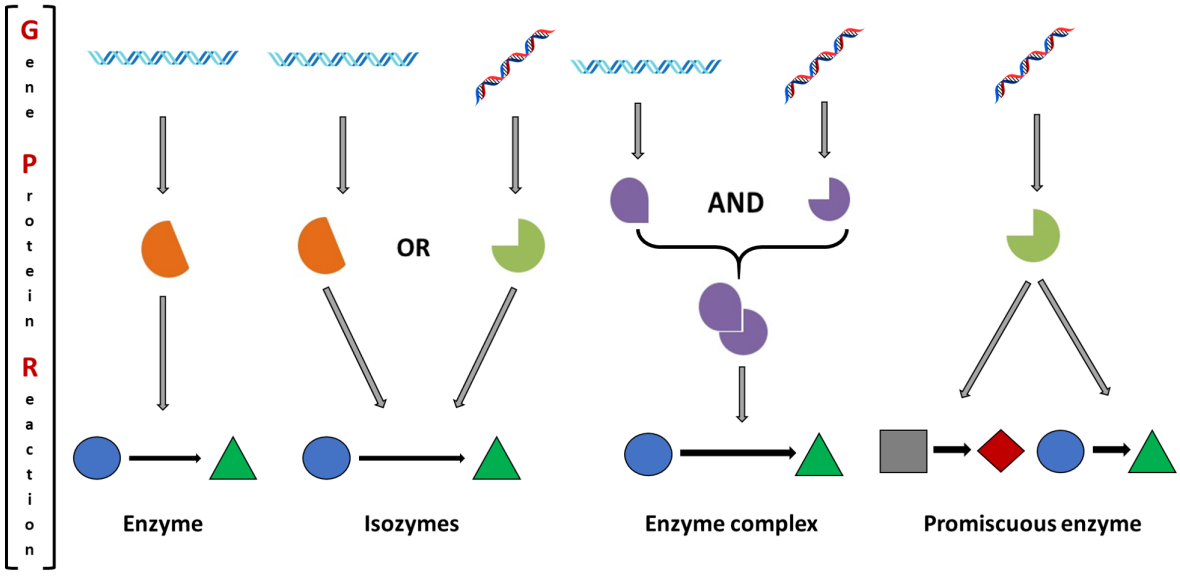


Figure 1.: Simplified schema representing the GPR associations.

Another crucial aspect is the addition of a biomass representative equation to the reconstruction. It consists in a representation of the biomass in the form of a reaction containing the macromolecules and other compounds that make up the dry weight of the cell. This reaction is used to represent cellular growth when performing computational simulations [54]. It is also important to identify the growth requirements of the organism to use in the aforementioned simulations.

There are several bioinformatics tools described in the literature capable of reconstructing GSM models. Model SEED [56], *merlin* [37], Pathway Tools [67], RAVEN Toolbox [14], SuBliMinal Toolbox [113], (MEMOSys) [96], (GEMSiRV) [81] and (CoReCo) [101] are some examples of a wider and continuously growing set of tools. Some of these tools may even perform simulations on the reconstructed models. That is the case of Model SEED, RAVEN, Pathway Tools and GEMSiRV.

2.2.3 *Conversion into Stoichiometric Model*

The next stage is the conversion of the network to a mathematical model. This mathematical model should include the biomass equation, a set of differential equations characterizing the concentration of each metabolite in the model and several other constrains mostly associated with the reversibility of the reactions and the concentration of some metabolites [36]. This model should then be saved in a computational-friendly format such as SBML [30, 59], BioPax [35], Pathway/Genome Database format [27, 28] or an Excell file.

Systems Biology Markup Language

The Systems Biology Markup Language SBML is a machine-readable format used for representing models of biological processes (systems where biological entities are involved) [3, 59]. It can be used to represent many different biological phenomena such as metabolic networks, cell signaling pathways and regulatory networks [59]. This format is defined as neutral regarding the programming languages and software encoding, even it is primarily orientated to be encoded using eXtensible Markup Language (XML) [3]. The use of a representation format as SBML offers several advantages like:

- Model's representations that allow the use multiple tools without model rewriting;
- Model's representations readable by different software environments, thus facilitating models' sharing and publishing;
- Model's endurance beyond the lifetime of the software used to create them.

SBML Level 3

SBML Level 3 is the most recent specification of the SBML format [58]. This SBML specification is a modular language, consisting in a core with a complete format capable to be used alone [58]. Its modular character enables the development and addition of extra Level 3 packages, which will extend the core, offering additional features [58]. Furthermore, this SBML level portray a new modular base for a future and continued expansion of SBML's features and capabilities by presenting several significant changes in constructs and syntax from the previous level.

A key aspect of this SBML format, is that it allows models of arbitrary complexity to be represented [58]. Therefore, each model's component type is represented using a specific type of data object comprising the component's relevant information.Thus, the top level of an SBML model core definition consists of (optional) lists of the components defined below:

- **Function definition:** Represents a mathematical function that may be used throughout the model.

- **Unit definition:** Represents definitions of a units of measurement that can be used in the expression of quantities in a model.
- **Compartment:** Represents a container of finite size, that can be an actual physical structure, and where species may be located.
- **Species:** Represents a group of entities of the same kind that are in a compartment and participating in reactions.
- **Parameter:** Can be defined as global (to the model) or local (to the reactions) and represents a quantity associated with symbolic name.
- **Initial Assignment:** Represents a mathematical expression used to determine the initial conditions of a model.
- **Rule:** This top element represents a mathematical expression, that can only be added to a set of equations constructed from the reactions defined in a model. Rules can be used in variable's values calculation or used to define the rate of change of a variable.
- **Constraint:** Mathematical expression used to detect out-of-bounds conditions during a dynamical simulation by computing a true/false value from model variables, parameters and constants.
- **Reaction:** Represents a statement describes the transformation, transport or binding process that can change the amount of one or more species, such as a metabolic reaction where the reactants are transformed into certain products.
- **Event:** Represents a statement that describes an instantaneous and discontinuous change in one or more variables of any type happening when a condition is satisfied.

As this tool has a modular character, other model's component types and lists of components can be added to the core. An example is the SBML Level 3 Package Flux Balance Constraints (FBC) [93] containing new components for extending the model, such as:

- **GeneProduct:** Represents a single gene product, and may be used to represent the gene product association (GeneProductAssociation component) with the Reaction component;
- **Objective:** Usually represents a linear combination of model variables (ux) and a sense (direction).
- **GeneProductAssociation:** Extends Reaction with one or more genes (GeneProduct component). In case that more than one gene is present in an association they are written as logical expressions using logical 'and' and 'or' operators.

Finally, an extended Model object is also created to serve as a container for the desired core components described above, and for the additional lists of components added to the core and coming from the two above-mentioned packages [58, 93]. The Model object must be located in the "<sbml > ... </sbml>" element and only one instance of it is allowed per instance of an SBML Level 3 Version 1 Core document [58].

2.2.4 *Model Evaluation and Validation*

The final stage is the evaluation and validation of the reconstructed *GSM* model with experimental data. The most common algorithm used for most simulations is the Flux Balance Analysis (FBA) [40], a constraint-based method for predicting the flow of metabolites through a metabolic network. Different physiological analyses, important to validate the model, like predicting growth rates, by-product secretion rates and gene essentiality, as well as calculating theoretical yields, are performed using FBA [54]. Besides FBA, a vast set of algorithms used in model simulations is present in literature. For instance, it is possible to perform prediction analyses on the metabolic state and phenotype of an organism after a gene deletion/knockout using the Minimization Of Metabolic Adjustment (MOMA) [110] algorithm which is similar to FBA in the stoichiometric constraints, but despises the existence of optimal growth flux for gene knockouts. This method bases on the assumption that initial mutant flux values remain close to wild-type ones, to provide a mathematically approximation for this intermediate suboptimal state [110]. Moreover, we can also use Regulatory On/Off Minimization (ROOM) [111] is a constraint-based algorithm that aims to minimize the number of significant flux changes with respect to the wild type, being able to predict steady-state metabolic fluxes and identify the pathways alterations enforced in response to the gene knockouts [111]. These methods are important because they can help understand, through regulation and evolutionary optimization, how a cell adapts to the loss of a gene. Furthermore, there are also algorithms to perform strain design such as OptKnock [25], a computational framework designed for suggesting gene deletion strategies that can lead to an enhanced production of specific biochemical compounds. These strategies focus on the search for reaction-associated genes whose deletion maximize the flux towards a desired product, though maintaining the internal flux distribution track on growth or another biological objective optimization [25]. Another example is the OptKnock similar, OptGene [97], a genetic algorithm that can identify, with relatively less computational time, gene deletion strategies for optimize desired phenotypic objective function, relying on the principle of Darwinian evolution to search for the global optimal solution [97]. Moreover, there are some algorithms/tools that incorporate and remove data from the model, such as Gene Inactivity Moderated by Metabolism and Expression (GIMME) [20] that focus on minimize the usage of low-expression reactions while control objective function values, the

integrative Metabolic Analysis Tool (iMat) [121] that finds the optimal arrangement between the inclusion of high-expression reactions and removal of low-expression reactions, or the Model-Building Algorithm (MBA) [62] that prioritize reactions able to ensure activity in the extract model, keeping them along with some profitable medium-confidence reactions, and randomly discarding reactions that does not support any of the above [94]. Beside this, other analyses such as finding metabolic dead ends, searching reactions for gap filling, finding and eliminating stoichiometrically balanced cycles should also be carried out in this validation stage [114].

Specialized tools for running simulations with GSM models were developed in the past few years, including OptFlux [104], COBRA [108] and CellNetAnalyzer [72].

When the model predictions are not in agreement with experimental data, the last three main stages should be repeated in order to improve and obtain a more efficient, precise and reliable GSM model [36, 114].

These models have been acquiring increasing importance as useful tools [91], for example, in drug-targets discovery in several diseases study.A detailed view on high-quality *GSM* models' reconstruction is available in several protocols referenced in this document [45, 114, 115].

2.3 GSM models' reconstruction tools

In the past decade SB and *GSM* models have gained an increased relevance in the study of biological systems, namely in the study of metabolic pathways and components, their interactions and in the prediction of organisms/biological systems behaviours.

Several tools have been developed to perform *GSM* models' reconstructions as already mentioned above. Here, a more detailed and specific description of some of these tools will be performed.

2.3.1 *Model SEED*

Model SEED is a web-based resource, published in 2010 by Henry *et al.*, developed for reconstructing *GSM* models by automating most of these steps, and it is the first online high-throughput metabolic modeling tool [56]. The Model SEED contains its own curated reactions database, in which the direction of the reactions is associated with the type of organism [54]. The Model SEED requires the annotation of the genome using RAST [18]. Users are allowed to perform manual curation on the obtained annotation using other tools before continuing with the reconstruction pipeline [56]. Several draft reconstructions for a variety of organisms are freely available in the Model SEED [54]. One important consideration regarding Model SEED is its focus on bacterial (and archaeal) reconstructions, thus

only including cytosol and extracellular compartments in the models [56]. The Model SEED has several automatic features such as generation of detailed GPR associations, including enzyme complexes and isozymes, addition of spontaneous reactions based on pathway completeness, with extracellular transport reactions added based on genomic evidence, and the generation of biomass representations based on the organism's phylogeny and genomic content [54]. Furthermore, transport reactions between inner compartments do not exist in this tool, as it only focuses on prokaryotic systems [54, 56].

2.3.2 *RAVEN Toolbox*

Reconstruction, Analysis and Visualization of Metabolic Networks (RAVEN) Toolbox is a software for reconstructing *GSM* models, in a semi-automated way, published in 2013 by Agren Rasmus *et al.* [14]. Besides model reconstruction, it also allows the analysis, simulation, and visualization of *GSM* models, as it is a MATLAB® toolbox [14]. Its main metabolic information reference source while performing the reconstruction is the KEGG database [14]. The user is allowed to import already published models and combine those with KEGG data, process which can be coupled with extensive gap-filling and quality control features [54]. To identify the KEGG Orthology (KO) identifier for each gene, RAVEN uses protein homology, then importing to the reconstruction the reactions and genes that correspond to that KO identifier [54]. A limitation that arises from using KEGG, is that it is not balanced requiring manual balancing. Despite that, RAVEN has a function to identify unbalanced reactions, which simplifies the process [54]. Regarding GPR associations RAVEN can provide these relationships, but also allows users to define them themselves. Another positive aspect on this tool, is the amino acid sequences-based prediction capability of compartments for reactions and genes, using WoLF-PSORT [57] reports [54]. The biomass equation is not generated automatically, being necessary for the users to add it manually. Other non-automatic features are, for instance, the insertion of spontaneous reactions and intra- and extracellular transporters, being that task up to users. This software allows to import and export models in two formats, which are SBML and a Microsoft Excel model representation [14].

2.3.3 *SuBliMinal Toolbox*

The SuBliMinaL Toolbox was published in 2011 by Neil Swainston *et al.*, and quoting the authors it "consists of a number of independent modules that can be used independently or chained together to form a reconstruction workflow allowing the generation of an initial draft of a metabolic reconstruction" [113]. This tool's draft model reconstruction includes data from both KEGG and MetaCyc, in a post-downloading merging process, of data re-

garding pathways and reactions for the organism in study [54, 113]. This may represent a limitation for the users because the reconstruction will be restricted to organisms whose metabolic data is present in such databases. SuBliMinaL Toolbox presents some similarities with RAVEN Toolbox. For instance, regarding GPR associations, SuBliMinaL can provide these associations, and allows users to define them themselves. Other similar aspect is the use of Wolf-PSORT [57] reports in the prediction of compartments for genes and reactions [54, 113]. Despite the fact this tool has some useful features like metabolite charges and formulas update and automatic reactions balance, SuBliMinaL offers no way for the users to perform a manual correction on metabolites and reaction which were not corrected automatically [54]. Resembling Model SEED, SuBliMinaL Toolbox can generate a set of reactions representing the biomass based on the organism's phylogeny and genomic content [54]. Regarding model validation SuBliMinaL does not provide in-software support for simulations. Hence, the reconstruction has to be exported to SBML to allows its upload into simulation tools, for simulations and validation of the model [54, 113].

2.3.4 *Pathway Tools*

Published for the first time in 2002 by Peter Karp *et al.*, Pathway Tools is a multi-component software environment for managing, analysing and visualizing integrated collections of genome, pathways and regulatory data [67, 68]. It handles many types of information besides pathways, and its capabilities are very extensive. In Pathway Tools, the upload of an annotated genome by the users is required [68], and reactions are retrieved from the manually curated MetaCyc database [68]. Curation of annotations is also allowed. Resembling Model SEED, Pathway Tools store several freely available Pathway/Genome Databases (PGDBs) containing the predicted metabolic network for a variety of organisms [27, 28, 54]. Each one of these specific databases, encompass metabolic pathways, enzymes, metabolites and reactions predicted by the Pathway Tools software using MetaCyc data, for the aimed organism [27, 28]. The fact that this software uses a manually curated database is an advantage, as it provides reviewed data on metabolic pathways, enzymes, metabolites and reactions, allowing users to skip manual curation of, for instance, reactions reversibility, metabolites charged state, reactions mass- and charge-balance [54]. This software does not provide reaction's compartmentalization, as it only focuses on bacterial and archaeal reconstructions. Regarding GPR associations, Pathway Tools can automatically generate them in a detailed way, which enzyme complexes and isozymes [54]. This tool also presents a specific curation feature called Evidence Ontology (EO) [66], which can be used to document the reconstructions, presenting justification and evidence for each reaction in the reconstruction [68]. The inclusion of spontaneous reactions, in Pathway Tools models, is based in pathway completeness. Like the RAVEN Toolbox, Pathway Tools does not auto-

matically generate the reaction representing biomass equation, relying on users to do it manually [54]. This tool allows exporting the model in three formats: SBML, BioPax or Pathway/Genome Database format [68]. Pathway Tools provide in-software support for simulations, integrating one of its components, the MetaFlux tool [78], for FBA simulations. This tool presents some considerable support for performing the necessary model evaluation simulations. Its tools allow the users to perform validation and evaluation steps like the identification of dead end metabolites, blocked reactions and network gaps associated with biomass precursors and secretion products [68, 78]. It also offers gap filling supported by reaction additions suggestions, complemented with genomic evidence of different reaction candidates [54, 68].

2.3.5 *merlin*

Metabolic Models Reconstruction Using Genome-Scale Information (*merlin*) is an open source software tool developed at the University of Minho and published in 2015 by Oscar Dias *et al.* [37]. This framework is built in JAVA®, a platform independent and object-oriented programming language. *merlin*, which operates with the support of the AIBench framework, allows the reconstruction of *GSM* models for any organism whose genome has been sequenced, as well as the functional annotation of its genome [37].

In the first module, *merlin* provides several tools to perform the genome functional annotation and curation, as well as the reconstruction of the draft model. This module allows the users to identify and select metabolic proteins, as well as predict the subcellular location wherein these proteins operate [37]. The genome annotation can be performed using different databases (NCBI and UniProt) and different alignment algorithms (BLAST, HMMER) [37]. In this case the homology information is retrieved for every gene and the results are automatically scored [37, 38]. Furthermore, *merlin* has tools to identify and annotate transport protein encoding genes. The transporters annotation is performed with a unique tool developed specifically for this purpose, called *TRIAGE* [37]. The transport reactions for those proteins are generated as well. This module also allows to perform model compartmentalisation, using LocTree3 [88], PSORTb 3.0 [119] or WoLF-PSORT [57] algorithms to retrieve predictions on the organelle localisation of the encoded proteins (enzymes) in the genome and thus the localisation of the metabolites involved in the reactions catalysed by the enzymes [37]. One of the best feature in *merlin*, is the existence of several tools for the curation of the genome annotation and the draft model [37, 38]. This can be a key aspect to support and help model validation and in the reconstruction of an accurate model.

The second module supports the *GSM* model assembling [37, 38]. This module's subcomponents allow loading existing annotations and metabolic data. Moreover, in this module the GPR associations to build the network are automatically generated, including the in-

tegration of genome annotations with the metabolic data, and the e-biomass reaction is included in the model. To facilitate that integration, both modules share the accession to internal databases containing all information, which can be accessed through *merlin*'s interface. Finally, the reconstructed metabolic model can be exported to the SBML standard format [37, 38].

A representative schema of *merlin*'s bimodular architecture and supported databases is displayed in figure 2.

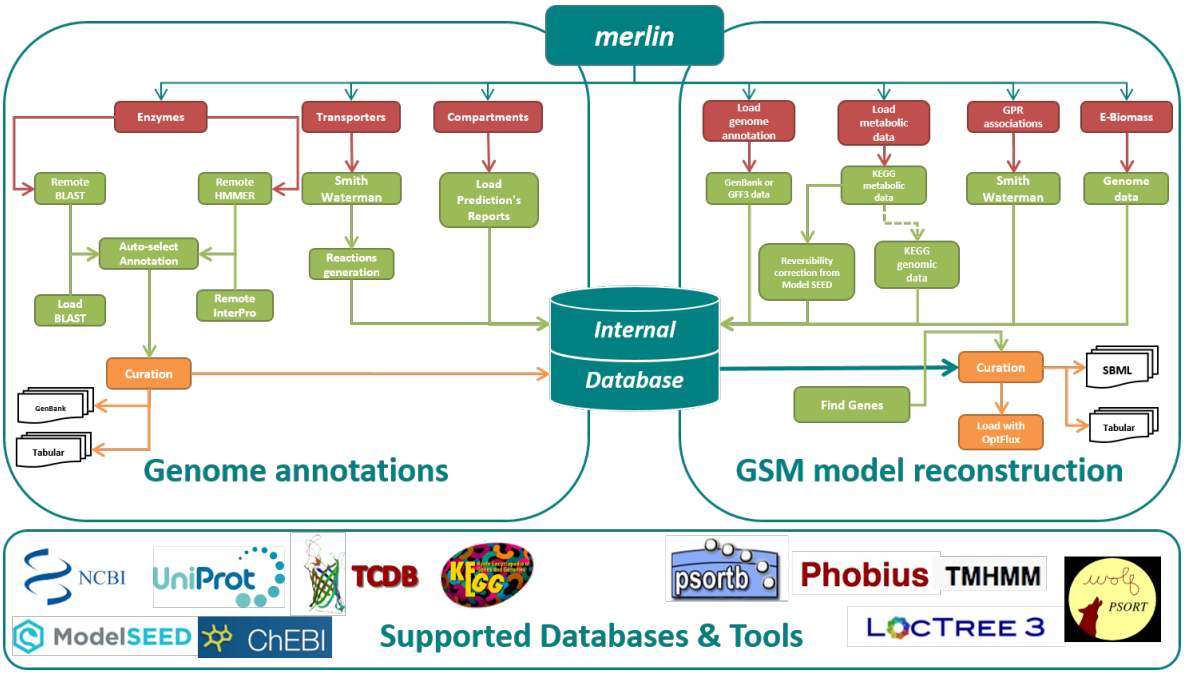


Figure 2.: *merlin*'s bimodular architecture and supported databases, addapted from [38]. *merlin* comprises a module dedicated to genome annotations and another to *GSM* model's reconstruction, as well as solid set of supported databases.

2.3.6 *Other tools*

Nowadays, GSM models' reconstruction is an emerging area in SB. Hence, there is a wider set of tools beyond those mentioned and described above. Metabolic Model research and development System (MEMOSys) [96], a tool that allows the management, storage, and development of metabolic models; Genome-scale Metabolic Model Simulation, Reconstruction and Visualization (GEMSiRV) [81], a software that provides metabolic model reconstruction, simulation and visualization; and Comparative Reconstruction (CoReCo) [101], a software useful for modeling metabolisms of multiple related species; are other examples of published tools to perform metabolic models' reconstructions [117]. A general overview over the reconstruction tools aforementioned is presented in table 1.

Table 1.: Genomic-scale metabolic model reconstruction tools overview.

	merlin	Model SEED	RAVEN	SuBliMinal	Pathway Tools	MEMOSys	GEMSiRV	CoReCo
Input	Species Taxonomy Identifier / genome	Genome annotated in RAST	Annotated genome sequence	Species Name	Annotated genome sequence		Existing models	
Reference Databases	KEGG, NCBI	SEED, RAST	KEGG	KEGG and MetaCyc	MetaCyc		KEGG, MetaCyc, SEED	KEGG, NCBI
Supported Organisms	All	Bacteria and Archaea	All (well suited for eukaryotic organisms)		Bacteria and Archaea			All
Manual Curation Interface	Available	Not Available	Not Available	Not Available	Available	Not Available	Available	Not Available
GPR rules	Available	Available	Not Available	Not Available	Not Available	Not Available	Available	Not Available
E-biomass equation	Manually	Automatic	Manually	Automatic	Manually			Manually
Simulations Support	Not Available	Available	Available	Not Available	Available	Not Available	Available	Not Available
Interface	*merlin* software	Web	Matlab	Command line	Pathway Tools software	Web	GEMSiRV software	CoReCo software
Output	SBML	SBML, Excel	SBML, Excel	SBML	PGDB, SBML, BioPax	SBML	SBML	SBML

2.4 GSM models' simulation tools

As said before, one of the main steps in *GSM* models' reconstruction is model evaluation and validation. This step can be performed by a specific set of applications, developed for that purpose. Although some model reconstruction tools have simulation support incorporated, most of them has no simulation capacities as the proper software reported in literature. OptFlux [104], COBRA Toolbox [19] and CellNetAnalyzer [72] are well established examples of those software.

2.4.1 *COBRA Toolbox*

Published for the first time in 2007 by Becker *et al.*, Constraint-Based Reconstruction and Analysis (COBRA) Toolbox is a MATLAB package containing an ensemble of methods designed to simulate, analyse and predict a variety of metabolic phenotypes using *GSM* models [19, 108]. COBRA Toolbox requires MATLAB® to run the desired simulations [108]. This tool uses a specific set of constrains, such as compartmentalization, mass conservation, molecular crowding and thermodynamic directionality, to enumerate the set of feasible phenotypic states for a reconstructed model network in a given condition [108]. This tool also allows the use of transcriptome data to reduce the size of feasible phenotypic states set [108]. Despite employing these constrains, COBRA may not provide a unique solution. COBRA Toolbox provides access to a wide range of methods, that include, among others, FBA, gene essentiality analysis, geometric FBA (gFBA), loop law, gap-filling and visualization of computational models of metabolism [77]. COBRA Toolbox has already incorporated basic algorithms that offer various functional analysis for mutant phenotype prediction such as MOMA [110] and ROOM [111] [77]. It also provides some strain design algorithms, such as OptKnock [25] and OptGene [97], as well as GIMME [20] and iMAT [121] algorithms to incorporate, for example gene expression data, in the environment- and tissue-specific metabolisms analysis, respectively [77]. It also presents algorithms for gap filling process. Furthermore, this tool helps the user identify the dead-end/blocked metabolites in the model [77].

COBRA Toolbox is in constant evolution, with new methods and resources developed by community members being included as add-ons to the core COBRA Toolbox, or to provide models or other input [108]. Hence, this toolbox provides users with a high-level interface to a variety of COBRA methods. The COBRA Toolbox supports models in the SBML format and MATLAB requires the libSBML and the SBMLToolbox to import models [108].

2.4.2 *OptFlux*

Developed at the University of Minho and published in 2010 by Isabel Rocha *et al.*, OptFlux is a computational application for metabolic engineering [104]. This software was also developed, on top of the AIBench framework [116]. It is an open-source and user-friendly application that can be used by researchers with no/little background in the area. OptFlux is also modular, as it was implemented in a way that new features are easily plugged in [104]. Thus, it can be classified as a plug-in-based application, as it can incorporate plug-ins, each containing a set of functionalities, allowing the reuse and integration of functionalities from past and future developments. OptFlux allows loading models in several formats, namely: SBLM level 2 and 3, Metatool reader specification, CellDesigner specific SBML format and table format [104, 116].

OptFlux uses of Evolutionary Algorithms, Simulated Annealing metaheuristics or the OptKnock [25] algorithm [104]. Optflux has also incorporated basic algorithms that offer various functional analysis for mutant and wild-type phenotype prediction such as MOMA [110] and ROOM [111], as well as the Flux Balance Analysis method [77]. OptFlux has some other main features such as methods for Metabolic Flux Analysis (MFA), which after receiving values for experimentally measured fluxes calculates the effects on the flux space, pathway analysis through the calculation of Elementary Flux Modes (EFMs) and also a tool that provides model visualization which eases the interpretation of results [104].

Furthermore, OptFlux has an internal repository of about 50 different compatible models and different organisms, which makes the access to oftenly used models easier to OptFlux users [104].

2.4.3 *CellNetAnalyzer*

Published in 2007 by Steffen Klamt *et al.*, CellNetAnalyzer is a toolbox-based software that runs over the MATLAB environment [72]. It consists in a graphical user interface for MATLAB®, used for structural and functional analysis of different types of network reconstructions [72]. CellNetAnalyzer allows performing many tasks like FBA, Flux Variability Analysis (FVA) and the simulation of gene deletion mutants. CellNetAnalyzer is, however, a more comprehensive software tool that allows to analyse metabolic, regulatory and signalling networks [72, 77]. To perform its tasks, this toolbox is capable of implementing advanced in silico algorithms including MOMA, ROOM, OptKnock and OptGene. This software allows the users to import not only SBLM files but also Metatool format and text files. Regarding exportation it is able to export to SBLM, Metatool and text format files [72, 77].

2.5 OTHER BIOINFORMATICS TOOLS

2.5.1 *KBase*

The Department of Energy Systems Biology Knowledgebase (KBase) is bioinformatics software tool and data platform that allows the execution of systems biology analysis of microbes, plants and their communities [11, 17]. It is characterized by being an open and collaborative environment that integrates a wide range of tools and data in a unified graphical interface. Users are allowed to create and run large-scale analyses and combine multiple lines of evidence to model plant and microbial physiology and community dynamics, without the exhaustive research work in different sources [11, 17].

Due its open and collaborative character it encourages and empower users to upload their own data, to collaborate on and share the reproducible analyses, workflows and conclusions of their biological data in the context of public data and private collaborator data [11].

Furthermore, to provide the user-platform communication, KBase relies on its main user interface, called Narrative interface [11, 17]. This interface is built with Jupyter Notebook [73], and allows users to create, save and execute dynamic and interactive documents called Narratives, that represent created systems biology analysis workflows. These documents can contain the data, analysis steps, visualizations, custom scripts, images, notes, and commentary of the computational experiments. These also can be kept private, shared with a specific set of people, or be made public for the benefit of the research community [11, 17].

Moreover, KBase offers several categories of applications, such as "Microbial Genome Assembly", "Comparative Genomics", "Annotation" and "Metabolic Modeling" [11, 17], that allow to perform several systems biology and modelling tasks like:

- Quick generation of assembled and annotated genomes and metabolic models using sequence reads;
- Comparison of growth phenotypes under hundreds of conditions;
- Interactive exploration on the comparative genomic organization, phylogeny, and gene content of organisms;
- Identification of enriched species and functions among sets of metagenomes.

2.6 JAVA APPLICATION FRAMEWORKS

2.6.1 *AIBench*

Artificial Intelligence workBENCH (AIBench) is a software development framework published in 2010 by D. Glez-Pea *et al.*, and it was a result of a collaborative project between

researchers from University of Minho and the University of Vigo in Spain [51]. AIBench is a framework that obligate the application of the Model-View-Controller (MVC) [74] software design pattern. Therefore, operations (controller), data types (model), and views are incorporated in the new software [51, 104]. This implementation with three well defined objects types, has some advantages such as the possibility of combining new and existing software components [51]. Furthermore, the fact that AIBench is plug-in based, where applications are developed adding components, called plug-ins, allows reusing and integrating functionality of past and future developments based on AIBench [104]. This structuring of the new software proportionated by AIBench simplifies the implementation work of the developers to the simple task of how to divide and structure the code into objects of three entities referred above [51, 104]. The rest of the work regarding the creation and implementation of a completely runnable final application is carry out by AIBench. In this process this framework performs several tasks such as producing a Graphical User Interface (GUI) , automatically retrieving the user parameters of a given operation, running operations, gathering the results and keeping them available for further use, displaying the results through custom or default views [51].

2.6.2 *Hibernate*

Hibernate is an Object/Relational Mapping (ORM) tool for Java environments [84]. As an a ORM tool, Hibernate not only provides mapping data from an object model representation (Java data type) to a relational data model representation (SQL data type), and vice versa, but also provides data query and retrieval facilities [82]. Being able to significantly reduce development time by eliminating the need for manual, hand-crafted data processing using Structured Query Language (SQL) and Java Database Connectivity (JDBC), Hibernate main goal is to ease and unburden the developer from 95% of common data persistence-related programming tasks [84]. Hibernate is a freeware distributed under the GNU Lesser General Public License. Besides its own API, Hibernate is characterized for implementing the Java Persistence API (JPA). Hence, it can be easily used in any environment supporting JPA, such as Java Standard Edition applications and Java Enterprise Edition application servers [6]. Nevertheless, with great stability and quality as qualities recognized by a wider range of Java developers, Hibernate is also characterized for being highly configurable and extensible [6].

2.7 *Treponema pallidum*

Treponema pallidum is a bacterium species that belongs to domain *Bacteria*, phylum *Spirochaetes*, class *Spirochaetia*, order *Spirochaetales*, family *Spirochaetaceae* and genus *Treponema* [33]. *T. pal-*

lidum is a motile spirochete, a helical to sinusoidal bacterium. Classified as a gram-negative bacterium, *T. pallidum* has outer and cytoplasmic membranes, a thin peptidoglycan layer and flagella. The genus *Treponema* contains both pathogenic and nonpathogenic subspecies [102, 103]. Some of the pathogenic Treponema species, can be classified in four different pathogenic subspecies, based primarily in the clinical manifestations of the diseases caused by them (treponematoses): subsp. *pallidum* (venereal syphilis), subsp. *pertenue* (yaws), subsp. *endemicum* (endemic syphilis), and subsp. *carateum* (pinta) [29, 102, 103]. Subspecies *pallidum* is the only transmitted by sexual contact. These pathogenic subspecies cannot be distinguished by morphologic, antigenic, biochemical, or genetic criteria. Their differentiation is based on geographic location, modes of transmission, and clinical manifestations [102, 103]. *T. pallidum* is an obligate parasite of humans, only surviving in association with a host, and so it cannot be cultured continuously in vitro [46, 103, 118]. It has one of the smallest prokaryotic genomes, represented by a circular chromosome of about 1000 kilobase pairs [29, 46].

2.7.1 *Subspecies pallidum*

The case study, used in this work, will be the subspecies *pallidum*, known as the causative agent of venereal syphilis. Venereal syphilis was first reported in Europe in the late 1400's and it is thought that it could have been brought from the New World [46]. Venereal syphilis is a well-known sexually transmitted disease dispersed worldwide, and is a significant public health problem, mostly in many underdeveloped countries, with the infectivity rates corresponding to the most sexually active age groups. This disease is caused by *T. pallidum* subsp. *pallidum* infection, which can present multiple clinical stages separated by long latent and asymptomatic intervals. *T. pallidum* subsp. *pallidum* was first identified by Schaudinn and Hoffman in 1905 [106]. This subspecies exhibits characteristic motility that consists of rapid rotation about its longitudinal axis and bending, flexing, and snapping about its full length [102, 103]. Furthermore, this bacterium also presents a considerable evasion of host immune responses. It appears to be, at least in part, due to the unique structure of the treponemal outer membrane [102, 103]. Although the primary infection is localized, the bacterium can rapidly disseminate and cause manifestations throughout the body [118].

2.7.2 *Genome sequencing and analysis*

T. pallidum subsp. *pallidum* strain Nichols complete genome was sequenced and released for the first time in 1998, by Claire M. Fraser *et al.* [46]. The complete genome sequencing of *T. pallidum* str. Nichols was preformed using a whole genome random sequencing method. The bacterium genome as a single double stranded circular DNA chromosome of 1,138,006

base pairs with an average G+C content of 52.8% [46]. There was a total of 1,041 predicted open reading frames (ORFs) which are DNA sequences that begin in a start codon and finish in a stop codon and represent 92.9% of total genomic DNA. They contain exons, introns and protein coding sequences that are able to be translated into proteins. The predicted biological roles were assigned to 577 ORFs (55%) [46]. *T. pallidum* origin of replication was identified using two criteria: the co-localization of genes (*dnaA*, *dnaN*, *recF*, and *gyrA*) often found near the origin in prokaryotic genomes and GC skew.

T. pallidum is a known parasite and has limited biosynthetic capabilities, requiring the uptake of multiple nutrients from the host. The bacterium genome encodes a pathway for the conversion of phosphoenolpyruvate or pyruvate to aspartate (through oxaloacetate) [46]. Despite being unable to synthetize enzyme co-factors, fatty acids, and nucleotides de novo, the presence of pathways for the interconversion of several amino acids was confirmed in *T. pallidum* [46].

This bacterium has a broad spectrum of amino acid transporters in response to its limited biosynthetic capabilities. *T. pallidum* genome contains 57 ORFs (5% of the total) that encode 18 distinct transporters with predicted specificity for amino acids, carbohydrates, and cations. In particular, they observed the existence of three ATP-binding cassette transporters with specificity for galactose (*mglBAC*), ribose (*rbsAC*), and multiple sugars (y4oQRS) with a broader substrate specificity associated [46]. The absence of phosphoenolpyruvate/phosphotransferase (PTS) systems for the import of carbohydrates was also noticed in *T. pallidum*. Moreover, this bacterium also requires an environment with limiting glucose concentrations for maximal expression of the mgl transporter. As *T. pallidum* has no recognizable inorganic phosphate uptake system, the uptake of glycerol-3-phosphate through the multiple sugar transporter may represent the primary means whereby this organism obtains inorganic phosphate [46].

Regarding the energy metabolism of the bacterium, *T. pallidum* only uses glucose, mannose and maltose as energy sources. Furthermore, the lack of the necessary anabolic and catabolic pathway genes allowed discarding the possibility of use amino acids as a source of carbon and energy [46]. Despite the presence of genes encoding all of the enzymes of the glycolytic pathway, there was no evidence of the genes encoding components of the tricarboxylic acid cycle or oxidative phosphorylation. This leads to the possibility that reducing power is generated through the oxidative branch of the pentose phosphate pathway [46]. To complete the energy metabolism analysis, the lack of a respiratory electron chain was observed. For that reason, ATP production should require phosphorylation at the substrate-level. As a result, membrane potential must be established by the reverse reaction of the ATP synthase, through a Na^+/H^+ gradient [46].

One of the cellular processes addressed on Claire M. Fraser et al. *T. pallidum* genome sequencing were the response of the bacterium to oxygen toxicity. The need of protection

mechanisms against oxygen toxicity are evidenced by its microaerophilic behavior. Due the lack of genes encoding oxygen-consumption related enzymes, NADH oxidase is the only in the organism to use O_2, being the only one known to protect the bacterium against oxygen toxicity [46].

Other aspect taken into consideration was the bacterium motility and chemotaxis. 36 genes encoding proteins involved in flagellar structure and function were highlighted [46]. *T. pallidum* has three core proteins (FlaB1, FlaB2, and FlaB3), a sheath protein (FlaA), and two uncharacterized proteins, also containing two copies of the flagellar motor switch protein, FliG [46].

One of the main components of this type of bacteria, and major cause of the their highly pathogenicity are the membrane proteins and lipoproteins. This is an important feature because it allows the bacterium to evade the human immune response. The analysis allowed the identification of 2 candidate outer membrane proteins and only 22 putative lipoproteins [46]. Furthermore, this bacterium contains a large family of duplicated genes (paralogs) (*tpr*A-L) that encode putative membrane proteins that may function as porins and adhesins [46].

Finally, although the lack of previous studies regarding the production of exotoxins by this organism, cytotoxic activity against neuroblasts and other cell types has been observed at extremely high concentrations of the bacterium. Genome analysis has revealed five genes encoding proteins similar to bacterial hemolysins [46].

A resequencing of the *T. pallidum* ssp. *pallidum* Strain Nichols genome was performed and released on 2013 by Helena Petrosova and coworkers [100]. In the original genome sequence, a total of 134 errors were identified including 25 substitutions and 109 indels. One hundred and seven errors (79.9%) were found in the coding regions (23 substitutions and 84 indels) and 27 (20.1%) errors were located in the intergenic regions (IGRs) [100]. In the coding regions of the re-sequenced genome, 19 substitution errors (out of 23; 82.6%) were corrected resulting in amino acid changes in the corresponding protein sequences including 13 nonconservative and 1 conservative changes [100].

3

METHODS AND TOOLS

In this chapter a detailed description of the main approaches, bioinformatics tools and Java libraries used in *merlin*'s improvements and in the development of new features will be performed.

3.1 RELATIONAL DATABASES

merlin supports two different Relational Database Management System (RDBMS), H2 and MySQL, for storing *GSM* models' data. The first one, is recommended when users are reconstructing a *GSM* model in their own computers, as *merlin* is ready for use with this system. The second one requires an external dependency installation (MySQL), which depends on the platform being used to host the database server. Thus, during this work the use of SQL language and its queries as well as the JDBC Application Programming Interface (API) will be imperative and mandatory to keep the interaction between the relational databases and the software tools development.

3.1.1 *MySQL*

MySQL (www.mysql.com) is an open source RDBMS, written in C and C++, which supports the use of SQL to access the information stored in its databases. It runs in practically all platforms, including MS Windows, Linux and Mac OSX. MySQL is a free, fast, stable multi-user and multi-threaded database server [12], which makes it the most popular RDBMS in the world [2]. Despite *merlin* does not embedded MySQL in its current version, it still supports connections to this server.

3.1.2 *H2*

H2 Database Engine (www.h2database.com) is an open-source RDBMS written purely in Java. It offers a reliable alternative to traditional RDBMS when it comes to processing

and memory costs, offering an extremely fast performance and leaving a very small disk footprint. Like MySQL, H2 fully supports SQL and JDBC API. In the H2 database engine all data are encrypted, and each database uses just a couple of files for storage. Furthermore, it can be embedded in any Java applications or run in the client-server mode. Thus, this database and its features make a very good match with *merlin*'s cross-platform versatility and different operating systems adaptability.

3.1.3 *The Java Database Connectivity API*

The Java Database Connectivity (JDBC) API is the standard for database-independent connectivity between the Java programming language and a wide range of databases, providing universal data access from the Java programming language [9]. Using this API, any data source, from relational databases to spreadsheets and flat files [9] can be accessed. Furthermore, using JDBC technology we can use the Java programming language for SQL-based database access. Comprised of two packages (java.sql and javax.sql), the JDBC API makes it possible to perform three tasks:

- Establish a connection with a database or access any tabular data source;
- Send SQL statements;
- Process the results.

JDBC API is characterized by key features like the full access to metadata, no installation needed as it is included in the Java Platform thus being automatically downloaded with it, and the identification of database connections by URL [10].

Finally, it is expected that this specific API will be a key piece in all developing work, since it will manage all interactions between the software and the databases.

3.2 AIBENCH FRAMEWORK

The main goal of this work is the improvement and the development of new functionalities for the *merlin* framework, a Java application built on top of AIBench framework. Regarding its architecture, AIBench is organized in several layers [51]. The Core layer is composed by the Core and Workbench plug-in. The Core plug-in is responsible to process all application-specific operations, detecting and registering as well as executing them when requested. The other plug-in, Workbench, is responsible for creating the main application window, the menu bar with all the implemented operations and generating the input dialogs requested by user operations when they are executed. It is also responsible for the graphical user

interface implementation. AIBench is based on three main concepts, which are present in any AIBench application: operations, datatypes and views.

3.2.1 *Operations*

AIBench operations are single Java classes that define "high-level problem-oriented processes" [51]. They are triggered upon request and represent the algorithm implementation as well as their Input/Output interface. Operations' input and output are defined through a set of ports, which are points where it can stablish whether data is an input or an output. For each created operation port, a method of the same operation class is associated to it. Ports can be classified as three types: In (for input data), Out (for output data) and In-Out (for both input and output data) [51].

AIBench operations will be a key resource during this work, since these not only facilitate the automation of several processes but also offer a fast and easy way to communicate between Core plug-in and Workbench plug-in, through the automatic input dialogs generation. The reusable character of Operations also highlights their capacity to expedite development work.

3.2.2 *Datatypes*

AIBench data-types are as important as operations and can be defined as Java classes used as inputs or outputs of AIBench operations. Mainly created to carry problem-specific data structures, datatypes do not require additional code to be considered AIBench datatypes [51]. Several operations will be created and used among this work, thus datatypes will be constantly created and/or manipulated.

3.2.3 *Views*

AIBench Views are used to graphically present result datatypes of the executed operations to the users. These allow the visualization of datatypes in a more understandable, simple and interactive way. Each view is implemented through a Java class and is associated with a data-type [51]. In this work, new views will be used to visualize the results of newly implemented operations, and existing views will be subjected to improvements.

3.2.4 *Graphical User Interface*

AIBench framework is also able to generate automatic standard GUIs for a given operation. It not only provides a GUI skeleton, but also dynamically builds input dialogs for every operation [51]. This AIBench capability will be very useful during this work as it will expedite the development of *merlin*'s new functionalities.

3.3 NCBI ENTREZ PROGRAMMING UTILITIES

The Entrez Programming Utilities (E-Utilities) are a set of nine programs [1] which provide a stable structured interface into the NCBI Entrez system. To search and retrieve the required data, E-Utilities receives a set of parameters as input and translates them into NCBI software readable values, using a fixed URL syntax. E-Utilities can access a large set of data from Entrez system, which currently has 38 databases covering a variety of biological data and literature. To access the required data, the first step involves posting an E-utility request to NCBI. E-Utilities has a base URL, which is used for all nine programs, `https://eutils.ncbi.nlm.nih.gov/entrez/eutils/`. After NCBI processes the data, results can be retrieved. Any computer language is supported as long as it is able to send a request to the E-utilities server and interpret the XML response [5].

In this work two E-utilities tools, namely E-Search and E-Summary, will be used to access the Assembly database.

3.3.1 *E-Search*

E-Search is one of the nine server-side programs in E-Utilities. This program is accessed by adding its directory, "esearch.fcgi", to the E-Utilities base URL. E-Search requires two parameters: "db" and "term". "db" value must be a valid Entrez database name. The "term" value must be an Entrez text query. Furthermore, this query can contain search fields like the organism taxonomy identifier. Therefore, E-Search can be used to provide a list of Unique Identifiers matching a given text query [13]. During this work this feature will be used to access NCBI Assembly records with organism taxonomy identifier as input "term".

3.3.2 *E-Summary*

The second program used from E-Utilities was E-Summary. This tool can be applied to retrieve document summaries for a list of input "UIDs". The E-Summary sends its requests to NCBI using E-Utilities base URL concatenated with its directory, "esummary.fcgi". It has

two required parameters: ”db” value, which must be a valid Entrez database name from where document summaries will be retrieved, and the ”id” value, which requires a single Unique Identifier or a list of Unique Identifiers from the previously specified database [13]. In this work the Unique Identifiers' list to be passed in E-Summary will be obtained using E-Search.

3.4 METABOLIC ENGINEERING WORKBENCH BIOCOMPONENTS

The Metabolic Engineering Workbench (MEWorkbench) Biocomponents, available at https://github.com/MEWorkbench/biocomponents, is a project developed by the Optflux development team and is an integrant part of the *GSM* models' simulation tool. MEWorkbench Biocomponents is a user-free project distributed under the GNU Lesser General Public License version 2.1 and it is available in Maven Repository with its latest release 1.1.0. This project contains classes that automate several biological data processing operations and proved to be very important aid for the development of SBML models. From this project three major classes turned out to be useful in SBML files data reading and processing: '*JSMBLReader*', '*JSBMLLevel3Reader*' and the '*Container*'. '*JSBMLReader*' and '*JSBMLLevel3Reader*' classes implement an interface called '*IContainerBuilder*' and are responsible for reading and parsing SBML level 2 and level 3 files, respectively. The '*Container*' class receives an '*IContainerBuilder*' ('*JSBMLReader*' or '*JSBMLLevel3Reader*' objects) as input and converts all the SBML's data (metabolites, reactions, genes, etc.) into MEWorkbench standard containers.

3.5 MODELSEED DATABASE FILES

Besides MEWorkbench Biocomponents, other resources were used to support the development of SBML import tool. One of those resources were the ModelSEED Database files. These files store all the biochemical data used to reconstruct metabolic models in ModelSEED, such as: all reactions and compounds, their identifiers, reactions equations and stoichiometry, etc. ModelSEED database files can be retrieved using ModelSEED's GitHub repository https://github.com/ModelSEED/ModelSEEDDatabase.

In this work the ”compounds.tsv” and ”reactions.tsv” files, as well as ”KEGG.pathways” file, which contains information of all KEGG pathways and associated reactions, will be used. This database files comprise information about over 27000 compounds, over 34000 reactions and over 400 pathways.

3.6 BASIC LOGICAL ALIGNMENT SEARCH TOOL

Basic Logical Alignment Search Tool (BLAST) [15, 16] is a sequence comparison algorithm developed to accelerate the search for optimal alignments in sequence databases, for a given query sequence. This algorithm can be used with amino-acid or nucleotides sequences. BLAST is a heuristic algorithm that performs local alignments, thus aims to retrieving results quickly. This characteristic makes BLAST a very useful tool when looking for motifs or domains in proteins from different species. Considering that several proteins have one or more functional domains, performing global alignments would detect fewer similarities and would likely miss domains or motifs.

There are many different variations of BLAST searches. In table 2, a brief overview on the different BLAST variations is displayed.

Table 2.: A small overview over the different BLAST variants.

BLAST variant	Protein-protein search	Nucleotide-nucleotide search	Protein-nucleotide database	Nucleotide-protein database	Protein-protein database	Notes
megaBLAST [86, 120]		X				optimized for very similar sequences
BLASTN		X				Looks for distant sequences
BLASTP	X					i
BLASTX				X		
TBLASTN			X			
PSI-BLAST					X	ii
RPSBLAST					X	ii
DELTA-BLAST [22]					X	ii

i - Basis of BLASTX and TBLASTN ; ii - Uses a Position-Specific-Scoring-Matrix (PSSM);

Furthermore, BLAST provides a variety of ways to perform a search, like through the NCBI BLAST website, BLAST URL API, BLAST standalone applications, BLAST+ remote service or even through C++ API.

3.6.1 *BLAST+ v2.7.1*

As mentioned above, one way to perform a BLAST search is to run BLAST locally with BLAST+ applications. These applications are very useful for users requiring resources

beyond those offered by the NCBI. These applications allow users to employ their own data and/or use other databases. This approach has a high computational cost but provides a faster and customized BLAST search. BLAST+ applications can run on Windows, and on most versions of UNIX (Mac OSX/LINUX). It also allows users to run BLAST using command line. All documentation can be retrieved through NCBI website via https://www.ncbi.nlm.nih.gov/books/NBK279691/.

In this work, *merlin*'s functionalities will be updated to allow using the BLAST+ applications, when properly installed on the computer. BLAST+ version 2.7.1 was downloaded from ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST/, and used to test this new feature.

Input

Running the BLAST+ application through the command line requires setting input parameters. For example, parameters such as the source of the subject sequences (e.g. FASTA file or public databases), the query FASTA file and output report folder path, and choosing which BLAST+ variant will perform the searches (BLASTP or BLASTN), must be specified. Furthermore, it is also possible establish specific thresholds to filter the outcome results, such as the maximum Expected value (E-value) allowed for saving hits or the maximum number of High Scoring Pairs (HSPs) ("max_hsps") to keep for any single query-subject pair. BLAST+ also allows to choose the BLAST output format, offering several alternatives like XML BLAST output and tabular file.

Output

As mentioned before, BLAST+ can output the similarity searches report in different formats. In this work the XML BLAST output will be used, as it offers an easily programmatically readable format for accessing the information. In this case, and as the name suggests, the results of the alignments search are display in an XML file whose structure can be visualized in figure 3.

In the XML BLAST output, for every hit obtained, different HSPs may be displayed, for which several statistics are calculated and presented. An HSP is a query-hit segment pair whose score cannot be improved by extension or trimming. E-value, bit score and raw score, alignments' representative values (like the alignment length or the number of gaps) are HSP statistics. In this work we will focus on e-value and bit score as thresholds for filtering BLAST results.

The **E-value** represents the expected number of matches with a given score (S) or higher that is likely to occur by chance, in a database search. This indicates the confidence level for an alignment, as the lower the E-value, the more significant the score and the alignment

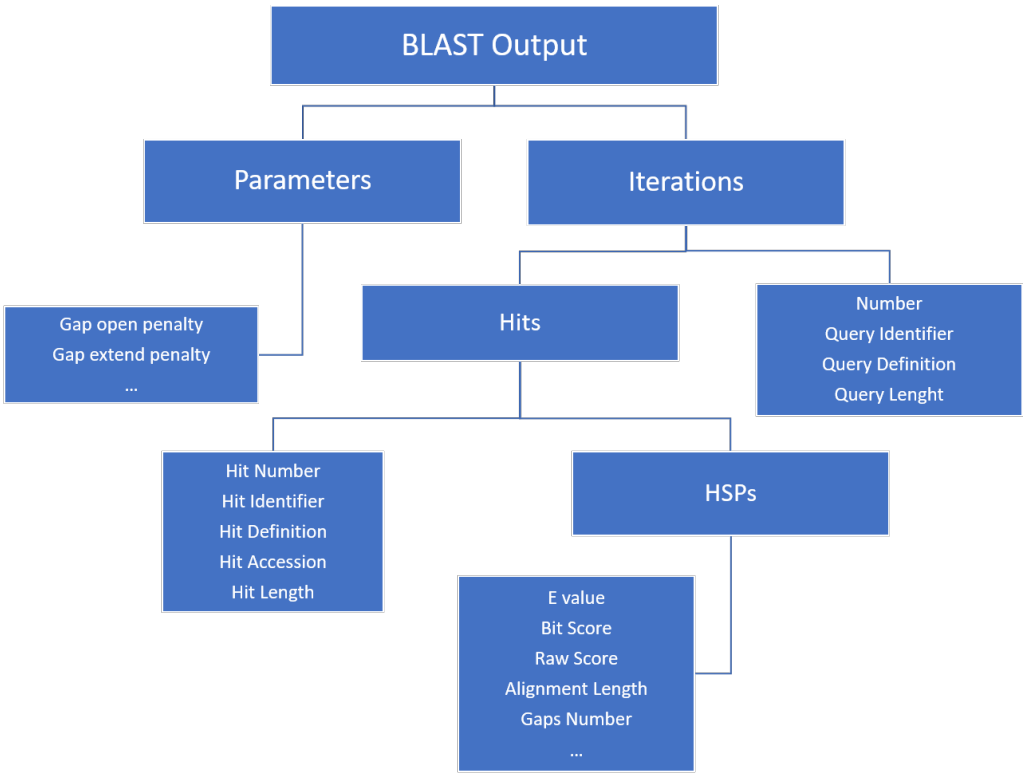


Figure 3.: Representative schema of the XML BLAST output structure.

[4]. Thus, the number of alignments (HSPs) that is expected to occur, with a score equal or higher than S, can be given by

$$E = Kmn\, e^{-\lambda S} \tag{1}$$

where K and λ are the Karlin-Altschul statistical parameters [65] that can be seen as natural scales for the search space size and the scoring system respectively, and m and n are the effective lengths of the query sequence and database, respectively [50]. This can be called the E-value for the score S.

The **raw score** of an alignment **(S)** is the sum of the substitution scores, given by the substitution matrix (PAM or BLOSUM), with the gap scores. Gap scores are generally calculated as the sum of the gap opening penalty (G_{op}) and the gap extension penalty (G_{ep}). Thus, for a gap of length L, the corresponding gap score is $G_{op} + G_{ep}L$. The choice of the gap penalty parameters (G_{op} and G_{ep}) is empirical, but typically higher values for G_{op} (between 10 and 15) and lower values for G_{ep} (between 1 and 2) are selected. Since raw scores have little meaning without the knowledge of the scoring system used, such as the statistical parameters K and λ, it is pointless to use this metric has a term of comparison. Thus, the calculation of a normalized score that takes the scoring system into account is mandatory.

The **score (S')**, is derived from the raw alignment score and intends addressing that problem. The bit score is then obtained by normalizing the raw score using the formula

$$S' = \frac{\lambda S - \ln K}{\ln 2} \tag{2}$$

in which S is the alignment raw score, and K and λ are the statistical parameters representing the statistical properties of the scoring system.

Therefore, the E-value for a given bit score can be calculate by

$$E = mn2^{-S'} \tag{3}$$

where m and n are the effective lengths of the query sequence and database, respectively.

3.6.2 *Java Architecture for XML Binding*

The Java Architecture for XML Binding (JAXB) [7] was used to perform the BLAST output report reading and parsing. This API provides a fast and convenient way to bind XML schemas and Java representations, expediting tasks like the incorporation of XML data in Java applications [8]. For that, JAXB provides methods for reading ('*unmarshall*') XML instance documents into Java content trees, as well as methods for writing ('*marshall*') that content back into XML instance documents [7, 8]. JAXB also provides a way to generate Java objects from an XML schema. It allows to automatic generate Java objects corresponding the BLAST output XML report fields [7]. In this work, that operation was accomplished using the binding compiler XJC. This compiler accessed the BLAST output DTD file to generate the Java objects (BLAST output classes) [8]. A DTD file is a text file that specifies elements and attributes used in an XML file. It is also used to validate the XML file before performing any parsing operation.

Furthermore, to manage the XML/Java binding information necessary to implement the JAXB method '*unmarshaller*', an abstraction provided by '*JAXBContext*' class was used. This class concedes the client's entry point to the JAXB API and inherits methods from the '*javax.xml.Object*' class.

3.7 SBML level 3

SBML Level 3 is the most recent specification of the SBML format. It can be described as a modular language, consisting in a core with a complete format capable to be used alone. Its modular character enables the development and addition of extra Level 3 packages, which will extend the core, offering additional features.

Since one of the implemented tools aimed enabling *merlin* to export in SBML level 3 format, its core structure was used. Additionally, new components were used when exporting to SBML 3. Namely, the '*GeneProduct*', '*Objective*' and '*GeneProductAssociation*' components. The '*Group*' class object from SBML Level 3 Groups package was also used. It provides a way of defining a group of entities: a simple mechanism for indicating that components of an model are related in some way.

3.7.1 *"Identifiers.org" Uniform Resource Identifiers*

The "identifiers.org" Uniform Resource Identifiers (URIs) were used to annotate Species, Reaction and GeneProduct components with a proper data referencing. "Identifiers.org" is an established resolving system, which focus on the Life Sciences domain, and enables the referencing of data for the scientific community. This system allows "the referencing of data in both a location-independent and resource-dependent manner" [63].

The canonical URI used for the unique identification of data entries is represented as follows:

```
http://identifiers.org/[collection]/[entity]
```

In which one needs to replace [collection] with the namespace of a data collection (all the 665 supported collections can be accessed at `https://identifiers.org/rest/collections`) and [entity] with the identifier of the entity created by the original data provider. All information regarding this system can be accessed online at available at `https://identifiers.org/`.

3.7.2 *JSBML library*

JSBML is a project that aims to create a free, open-source, pure Java library for reading, writing, and manipulating SBML files and data streams [39]. One of JSBML core qualities are the SBML parser and library to map all SBML elements to a flexible and extended type hierarch. All SBML constructs up to Level 3 Version 1 Release 1 are supported by JSMBL. However, some of the more complex functions of libSBML Java API such as consistency-checking and SBML validation are not reimplemented by JSBML. Currently, JSBML library is freely available through the JSBML API version 1.4. This library will be used in the development of the SBML Level 3 exporter tool, expediting the XML fields hierarchical structuring and organization [39].

3.8 REFERENCE *GSM* MODELS, GENOMES AND SELECTION CRITERIA

In this section all the metabolic models and respective organism's genome used during this project are displayed in table 3. The selection of candidate models to use as input in our *T. pallidum* draft reconstruction obeyed the following criteria: the organism needed to be a gram-negative bacterium whose model's SBML file was available online.

Table 3.: Summary of the organisms and respective *GSM* models used during this project, as well as the associated taxonomy identifiers and RefSeq's genomes.

Organism	Taxonomy Identifier	Genome's RefSeq Accession	Model Identifier
T. pallidum subs. *pallidum* strain Nichols	243276	NC_021490	*i*SM161[87]
Escherichia coli str. K-12 substr. MG165	511145	NC_000913	*i*AF1260 [42]
Bordetella pertussis Tohama I	257313	NC_002929	*i*BP1870 [23]
Piscirickettsia salmonis A1-15972	1269812	NZ_CP012413	*i*PSA584 [34]
Yersinia pestis CO92	214092	NC_003143	*i*PC815 [31]
Klebsiella pneumoniae MGH 78578	272620	NC_009648	*i*YL1228 [80]

3.9 CLUSTAL OMEGA

Clustal Omega [112] is a multiple sequence alignment software tool, used to align three or more sequences and can be accessed at https://www.ebi.ac.uk/Tools/msa/clustalo/. It aims to produce biologically meaningful multiple sequence alignments of divergent sequences and displays the results in Clandograms or Phylograms to observe possible evolutionary relationships.

It comprises the execution of three simple steps:

1. Setting the alignments tool input sequences;
2. Change the alignments search tool default parameters and set the output format;
3. Submit the alignments search request.

This tool allowed determining the phylogenetic proximity between six different organisms, which have manually curated *GSM* models published in peer-reviewed journal. Hence,

this tool was used to assess which models would be used to perform the automatic reconstruction of the *T. pallidum* draft model. The Clustal Omega multiple sequence alignment search was performed using these organisms' 16S rRNA sequences as the input sequences. The alignment searches were performed using the tool's default parameters.

3.10 KBase apps

Before the draft reconstruction of *T. pallidum* in *merlin*, the selected reference models for the reconstruction were processed in KBase to convert the BIGG identifiers into ModelSEED identifiers. Thus, three KBase apps were used:

- **"Import TSV/XLS/SBML File as an FBAModel from Staging Area":** Developed by James Jeffryes, allows to import a SBML model from the user's computer into KBase's workspace, only requiring the model to import as input. It requires 4 parameters, namely the format of model file to upload, the model file path, the biomass reaction identifier in model file and the compounds file path, if uploading a TSV file. This app was used to import *i*PSA590s model of *P. salmonis* A1-15972 into KBase's workspace for further processing. This app specification is available at https://github.com/kbaseapps/kb_uploadmethods/;
- **"Import model SBML from web"**: Developed by Filipe Liu, it allows the users to upload an SBML model from web, by providing the SBML model URL. This app was used to upload both the *i*BP1870 of *B. pertussis* and *i*AF1260 of *E. coli* K12 MG1655 into KBase's workspace for further processing. This app specification is available at https://github.com/Fxe/kbase-SBMLTools;
- **"Integrate Imported Model into KBase Namespace"**: Developed by Filipe Liu, it aims to integrate the imported model namespace into KBase's, allowing the conversion of identifiers from a specific database to ModelSEED identifiers. This tool was used to convert the three above-mentioned models' "Species" and "Reactions" components identifiers from BIGG's to ModelSEED's. This app specification is available at https://github.com/Fxe/kbase-SBMLTools/.

3.11 Biomass assembly

The biomass composition for *T. pallidum* was assembled using the core biomass composition of the *i*AF1260 model of *E. coli* K12 as template, thus the same weight fraction for each macromolecule composing the biomass was the same. Firstly, all available molar fraction coefficients, generated by the *merlin*'s updated e-BiomassX tool, were retrieved for

each molecule composing the macromolecules RNA, proteins and DNA. The molar fraction coefficients of *E. coli* were used for the cofactors, lipids, murein and lipopolysaccharide (LPS). Thus the *E. coli*'s biomass lipids (phosphatidylethanolamine dihexadecanoyl and phosphatidylethanolamine dihexadec-9enoyl) and LPS ((Kdo)2-lipid IVA) were included in *T. pallidum*'s biomass composition, as these are core components of bacterial cells. Cofactors that were not available in the draft network were not included in the biomass composition.

The full biomass assembly table with all coefficients is available as supplementary table s.1.

3.12 SUPPLEMENTARY FILES

All supplementary files, namely the SBML files, including the *T. pallidum* draft reconstruction generated by *merlin*, and the "TpallidumBiomassAssembly.xls" file, used in this work are available at https://nextcloud.bio.di.uminho.pt/s/LzNkdKiA3otmxfR.

3.13 THIRD-PARTY TOOLS AND LICENSES

The third-party tools used during the development of this work, namely their type of license and source, are displayed in table 4.

Table 4.: Third party tools used in the development of *merlin*'s new features.

Tool	Task	Type of license	Source
Eclipse photon	Development of the software	Eclipse Public License - v2.0	https://www.eclipse.org/downloads/
AIBench	Implement all operations	GNU Lesser General Public License v3.0	https://github.com/sing-group/aibench-project
JSBML	Read/Write SBML files operations	GNU Lesser General Public License v2.1 or later	https://github.com/sbmlteam/jsbml/tree/jsbml-1.4
BLAST+ 2.7.1	*merlin*'s similarity searches	Open Source License	ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/
KBase	Convert SBML models' namespaces/ Generate a draft of *T. pallidum*	Open Source License	https://kbase.us/
MEWorkbench Biocomponents	Read/Write SBML files operations	GNU Lesser General Public License v2.1	https://github.com/MEWorkbench/biocomponents
JAXB	BLAST parser implementation	CDDL 1.1 and GPL 2.0 with Class-path Exception	https://github.com/javaee/jaxb-v2

4

SOFTWARE DEVELOPMENT

Several improvements and new features were implemented in *merlin*. These improvements were motivated by a continuous need to expedite and automatize the reconstruction of *GSM* models. All the improvements and new tools were implemented on the Java programing language using the development environment Java™ Platform Standard Edition Development Kit 8. Additionally, SQL language was also used, as *merlin* relies on MySQL and H2 RDBMS to store data. In this chapter, all development work will be described in detail.

4.1 GENOME SEQUENCE AND ANNOTATION DATA FILES AUTOMATIC DOWNLOAD

As already mentioned above, to start a *GSM* model reconstruction in *merlin* the genome sequence of the target organism is required. Thus, the first logical step is to retrieve the genome sequence files for the desired organism (accessing an online database or with a customized file from an independent source) and then upload it into *merlin*. One way to retrieve such file is to access NCBI's Assembly database, using the taxonomy identifier of the organism as a search reference, and download the protein sequence FASTA file. This task comprises a few steps and does not requires programmatic knowledge, nor specific modeling knowledge. However, it can be confusing and time consuming for inexperienced users. Therefore, the existence of a feature that can expedite and simplify these steps can be useful to users, by simplifying the first step of the *GSM* model reconstruction.

Thus, a tool that executes the above-mentioned tasks in a simple and automatic way was developed. This tool takes the organism taxonomic identifier as input, and then searches for the corresponding NCBI's Assembly records presenting them in a GUI designed specifically for this purpose. Then, it allows users to choose from which Assembly record and repository (GenBank or RefSeq) the genome files will be downloaded. This tool can be divided in two major components: the fetching component, which will search and retrieve the Assembly records for a given taxonomy identifier; and the graphical component, represented by a GUI incorporated in the "New Workspace" GUI, displaying the result Assembly records and relevant information of each one.

4.1.1 *Searching and retrieving Assembly's records*

As the first component of this tool aims to search and retrieve all Assembly records associated with a given taxonomy identifier, the first step is to get a list of Assembly's unique identifiers from the requested taxonomy identifier. This list of Assembly unique identifiers can be retrieved using a library available in *merlin* that uses the NCBI REST API and provides several classes including, '*EntrezServiceFactory*' and '*EntrezService*' interface. The first one takes a base URL as input argument and it is responsible to generate an object that can be used to build the '*EntrezService*' object. The '*EntrezService*' class is an interface that provides access to the NCBI Entrez system from *merlin*, by selecting which of the E-utilities application will the request be send to. The first step is to retrieve a list of Assembly identifiers, for a given taxonomy identifier. Thus, the E-Search application was used, and the request received the "Assembly" string and the taxonomy identifier as the "db" and "term" input parameters, respectively. Thereafter, the E-Summary service is accessed using the '*EntrezService*' class and the Assembly records for the given unique identifiers list retrieved. The E-Summary application returns the results in XML format whose structure comprises three main Elements: "ESummaryResult", "DocumentSummarySet" and "DocumentSummary". Therefore, three classes were implemented to handle and read the result output, namely:

- ***'DocumentSummary'*** - this class aims accessing and storing elements within each XML's "DocumentSummary". Since each "DocumentSummary" element is associated to a distintinct unique identifier and represents a single Assembly record, the number of "DocumentSummary" in a "DocumentSummarySet" will be equal to the number of unique identifiers in the input list and therefore to the number of Assembly records associated with the target taxonomy identifier;

- ***'DocumentSummarySet'*** - this class is responsible to access the data coming from the '*DocumentSummary*' class. It accesses the data within the "DocumentSummarySet" element, namely the "DocumentSummary" elements and their descendants, and retrieves a list of '*DocumentSummary*' objects.

- ***'ESummaryResult'*** - this is the main class of the two classes referred above. It retrieves a list of '*DocumentSummarySet*' objects. Thus, it contains all elements data in the E-Summary XML output.

A Java class comprising several static methods that executes all tasks associated with the Assembly records data fetching and processing, as well as the download of genome files, the '*AssemblyUtils*' class, was also implemented.

4.1.2 *Graphical Component*

The graphical component of this tool comprises two customized GUIs: the "Assembly Panel" and the "Download NCBI Files" panel.

Assembly Panel GUI

The "Assembly Panel" GUI aims to display all Assembly records associated to a given taxonomy identifier, as well as relevant information about those records. This GUI is implemented in the *'AssemblyPanel'* class and is organized in two "JPanel" objects. The main panel comprises a "JComboBox" to display all the Assembly record entries found, a "JTextArea" to fill with the relevant information regarding the selected Assembly record and a second "JPanel". This second "JPanel" contains two "JRadioButton" objects, which allow users to choose from which database (GenBank or RefSeq) the genome files will be downloaded from. The "Assembly Panel" GUI was then embedded in the *merlin*'s "New workspace" GUI, together with other new features such as the insertion of the "Search" button. This button is responsible to trigger all the processes related to the search of the Assembly records, the retrieval of information and the generation and insertion of the "Assembly Panel" into the "New Project" GUI. The layout of this new GUI and the reformulated "New Project" GUI can be visualized in Figure 4.

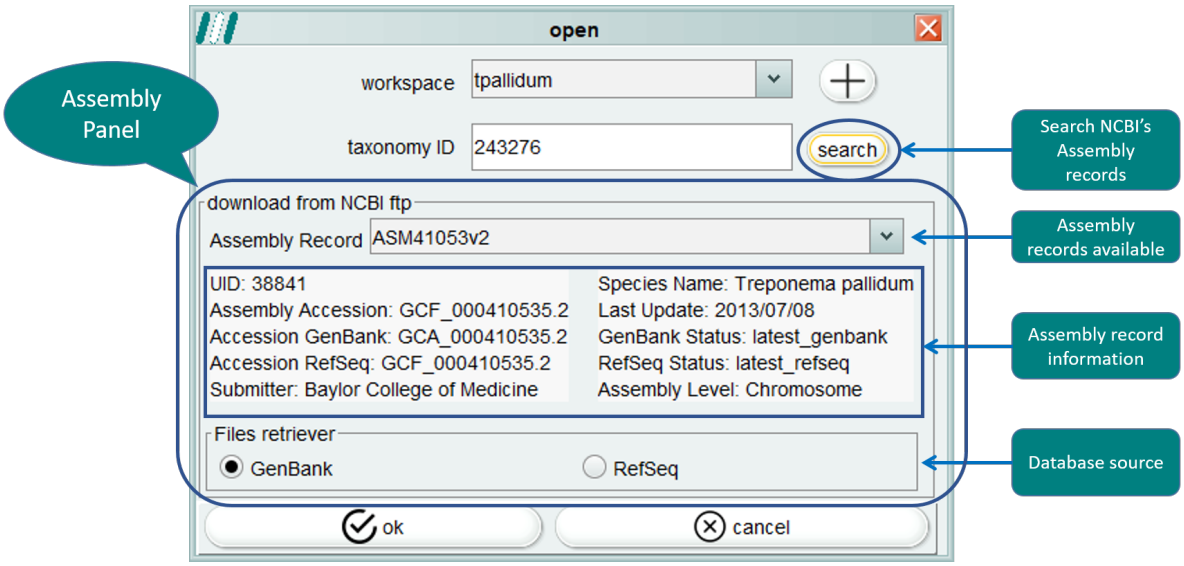


Figure 4.: "New Project" GUI final layout with the "Assembly Panel" GUI embedded.

Download NCBI Files GUI

The "Download NCBI Files" panel is a customized GUI implemented in the '*DownloadNcbiFilesGUI*' class. It was created with the purpose of being called separately from "New workspace" GUI, offering users the chance to download genomes files from the NCBI Assembly ftp even after a project is already initiated. This class has the "Assembly Panel" GUI embedded, such as in the "New workspace" GUI. Despite having a layout similar to the "New workspace" GUI, it contains several differences not only regarding the graphical aspect, but also regarding the related processes, such as the "workspace" combo box filling process or the associated AIBench Operations. Another difference is the fact that only the taxonomy identifier specified when the project initiation can be used to search the Assembly records. Furthermore, the "Download NCBI Files" GUI association with the '*DownloadNcbiFiles*' operation and the insertion into *merlin*'s menu bar was performed inside the Core Plug-in. Its layout can be visualized in Figure 5.

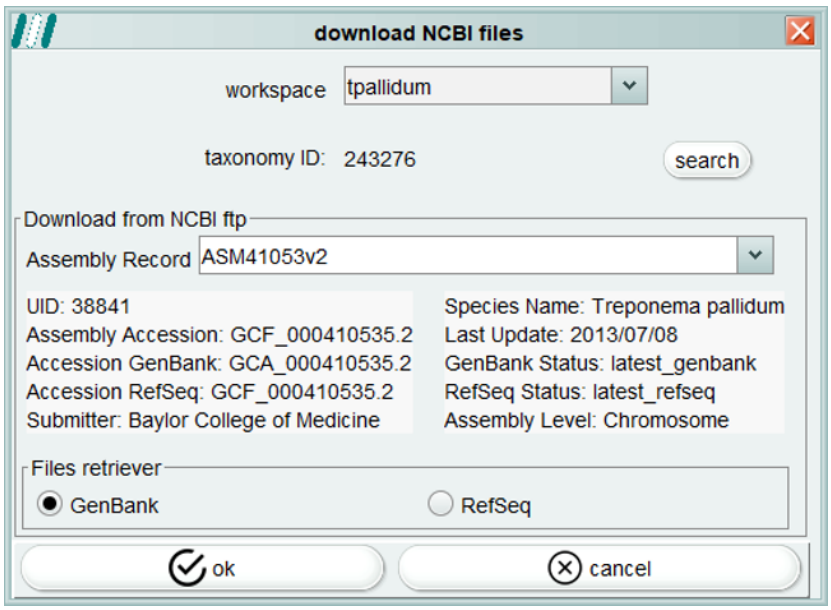


Figure 5.: "Download NCBI Files" GUI layout.

4.1.3 *"DownloadNcbiFiles" Operation*

Since *merlin* is built on top of the AIBench framework, the use of its Operations is crucial not only to add customized GUIs such as "Download NCBI Files" GUI but also to automatize different tasks and to make them reusable. Hence, the '*DownloadNcbiFiles*' operation, composed by 4 "In" ports receiving as input a "DatabaseAccess" object that represents the connection between *merlin* and the chosen RDBMS, a "Long" object corresponding to the

taxonomy identifier of the organism, a "Boolean" representing the repository (*true* representing GenBank and *false* representing RefSeq) from which the genome files will be downloaded, and the "DocumentSummary" of the Assembly record prior selected in one of the GUIs, was developed and implemented. This operation runs a pipeline in which several methods from '*AssemblyUtils*' and other *merlin* classes are used, for instance, to:

- Get the FTP URL from the "DocumentSummary" object;
- Download all files present in the FTP to the current workspace folder;
- Unzip, if needed, and process the FASTA files.

4.2 SBML MODELS IMPORTER

The major feature to be added to *merlin*, developed and implemented in this work was the SBML model's importer. This tool aims to import models built using other reconstruction software and exported in the SBML language

During the SBML importer development already implemented methods, not only from merlin but also from MEWorkbench (`https://github.com/MEWorkbench`), were used. This tool can be divided in two major components: a component responsible to read and process the SBML model data and a component to load the model data into database. A simplified schema of the SBML models import tool pipeline is displayed in Figure 6.

4.2.1 *Reading and processing SBML model data*

This first component aims at reading the SBML document, retrieving and processing all the data contained in it. The MEWorkbench Biocomponents project was used to expedite this task.

As mentioned above, the MEWorkbench Biocomponents project contains methods able to read and process models in the SBML format. Furthermore, it also has the Container class that allows to organize and store all SBML model's data into subcontainers. Therefore, three classes from MEWorkbench Biocomponents were used:

- ***'JSBMLReader'*** - this class implements the '*IContainerBuilder*' interface. It allows to read and get the information present in the SBML model, namely the network information such as compartments, reactions, metabolites, genes and stoichiometry, saving them in the container classes, namely: '*CompartmentCI*', '*ReactionCI*', '*MetaboliteCI*', '*GeneCI*' and '*ReactionConstraintCI*', respectively. These subcontainers are then passed to the '*Container*' class for further processing.

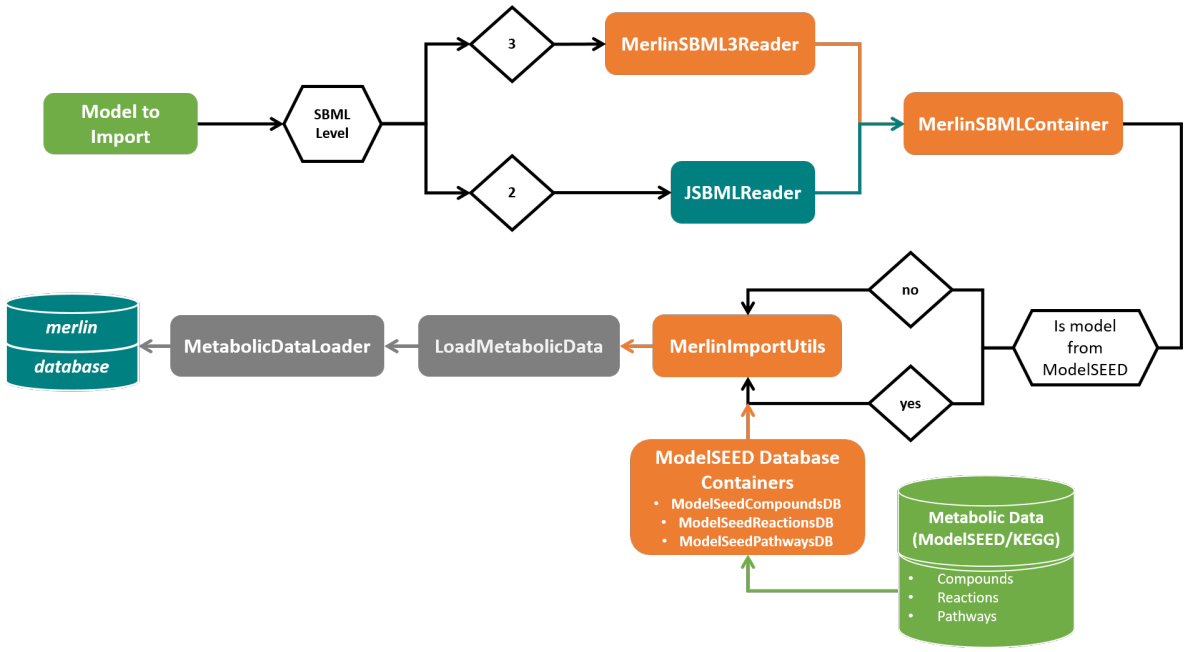


Figure 6.: Representative schema of *merlin*'s SBML models' importer pipeline, correlating the external data (green), already implemented classes (blue), adapted classes (grey) and the newly implemented classes (orange) used in it.

- ***'JSBMLLevel3Reader'*** - like *'JSBMLReader'*, this class implements the *'IContainerBuilder'* interface and works the same way. However, it is implemented to deal with SBML Level 3 models reading.
- ***'Container'*** - this class receive an *'IContainerBuilder'* as input and is responsible to process all information coming from the containers built by the *'JSBMLReader'* or *'JSBMLLevel3Reader'*.

However, the lack of information retrieved from SBML level 3 models by the *'JSBMLLevel3Reader'* class led to the creation of two new classes, *'MerlinSBML3Reader'* and *'MerlinSBMLContainer'*, extending the *'JSBMLLevel3Reader'* and *'Container'* classes, respectively. These classes were developed to manage and adapt the reader and container builder class at *merlin*'s requirements. Hence, the *'MerlinSBML3Reader'* also allows to retrieve proteins information, such as the EC numbers or database reference identifiers. These modifications were also handled in the *'MerlinSBMLContainer'* class.

Despite all implemented classes to process and organize the SBML information, a new one was needed to process and convert MEWorkbench Biocomponents containers into classes readable by *merlin*. Thus, the *'MerlinImportUtils'* class was implemented. Its constructor receives a *'MerlinSBMLContainer'* object and the Java Enumerator *'ModelSources'* class as input. The model source input requirement allows the *'MerlinImportUtils'* class to process SBML model's information and use different resources according the model source.

The *'ModelSources'* enumerator class was created to specify all SBML model sources from which *'MerlinImportUtils'* can import and process data, with higher detail and quality. In these cases, the *'MerlinImportUtils'* class is able to use the information contained in the *'MerlinSBMLContainer'* retrieved from the SBML file, and retrieve information from external sources, such as specific source database files or online databases. In this work, *'MerlinImportUtils'* was only fully implemented to process SBML models from ModelSEED and KBase. Nevertheless, it is able to receive and process SBML models from databases other than the ones available in the *'ModelSources'* enumerator class, although the quality of the imported model may decrease, because no additional information resources will be available, and it will rely totally on the information present in the SBML file.

'MerlinImportUtils'

'MerlinImportUtils' class focus on processing the six major model components: metabolites, enzymes, reactions, genes, compartments and pathways. Thus, six methods were implemented within this class to read the MEWorkbench containers from *'MerlinSBMLContainer'*, and to process and convert them into *merlin*'s containers (MetaboliteContainer, EnzymeContainer, ReactionContainer, GeneContainer, CompartmentContainer and PathwaysHierarchyContainer). From all the six above-mentioned model components, metabolites, reactions and pathways required a more detailed processing.

Regarding the processing of metabolites, same species with different compartments were treated as the same metabolite and boundary metabolites were ignored. Furthermore, metabolites specific data such as name, formula and molecular weight are retrieved from ModelSEED compounds database file, when available. Otherwise, that data is retrieved from the SBML model, if present.

Concerning reactions, components such as the gene rules or the drains and transport reactions took special consideration. The original reaction gene rules, retrieved from the SBML file and present in the *'MerlinSBMLContainer'*, were unfolded (*merlin*'s adopted format for reactions gene rules), using a recursive method that retrieved all possible gene combinations for each rule. Drains and transport reactions were identified and marked. In addition, the reaction equations were constructed from the reagents and products data coming from the SBML, in the absence of reactions information in the ModelSEED reactions database file. Other reaction's specific data such as the associated KEGG identifiers and the associated enzyme names are also retrieved from ModelSEED reactions database file, if available.

Finally, pathways were structured hierarchically, being the less specific ones marked as super-pathways, and comprising the more specific ones. Pathways processing fully depended on the ModelSEED pathways database file to retrieve all pathways and all hierarchical connections. Furthermore, the pathways-reactions associations were performed using

the reactions' KEGG identifiers, as ModelSEED's pathways file only contained pathways from KEGG, and KEGG identifiers.

ModelSEED Database Files Containers

The processing of reactions, metabolites and pathways was supported by the ModelSEED database files, available at https://github.com/ModelSEED/ModelSEEDDatabase. Three ".tsv" files retrieved from "ModelSEEDDatabase" GitHub repository, containing ModelSEED database information, were used. Hence, three container classes were implemented to read and store the data present in these files, namely:

- ***'ModelSeedCompoundsDB'*** - this class aims to read and retrieve the data of the ModelSEED "compounds.tsv" file, saving it in a Java "HashMap" object, where the keys are the ModelSEED compound identifiers and the values are Java arrays of "Strings", with compounds specific data. The specific data retrieved for each compound comprises the name, abbreviation, formula, mass, International Chemical Identifier (InChI), charge and linked compounds.
- ***'ModelSeedReactionsDB'*** - this class aims to read and retrieve the data of the ModelSEED "reactions.tsv" file, saving it in a Java "HashMap" object where the keys are the ModelSEED reaction identifiers and the values are Java arrays of "Strings", with the reactions specific data. The specific data retrieved for each reaction comprises the abbreviation, associated enzyme name, associated KEGG identifier, stoichiometry, direction, equation, spontaneity, compounds in the reaction, linked reactions and also if it is reversible and a transport reaction.
- ***'ModelSeedPathwaysDB'*** - this class aims to read and retrieve the data of the ModelSEED "KEGG.pathways" file, saving it in a Java "HashMap" object where the keys are the KEGG pathways identifiers and the values are Java "ArrayList" objects, with the pathways specific data comprising the pathway name, aliases and associated reactions. However, in this case the pathways data retrieved from the file are processed to build the hierarchical associations need to load the database.

4.2.2 *Loading model data into database*

The second component/step of the SBML importer tool aims to load the processed model data, from the *'MerlinImportUtils'* containers into the database. For this, *merlin* classes were used, namely the *'LoadKeggData'* and *'KeggLoader'* classes. The first class implements the Java Runnable interface and aims to execute the loading process of the data coming from KEGG database during the "load KEGG metabolic data" or "load KEGG annotation"

operations. The second one provides the connection between the runnable class and the Database-Connector Services. The '*LoadKeggData*' class was reworked into the broader and more general class called '*LoadMetabolicData*', that reuses most methods of the original class and implements a new constructor to run specifically with the import models' tool, without interfering with the KEGG data loading mechanism. The same approach was done for the '*KeggLoader*' class , from which resulted the new and broader '*MetabolicDataLoader*' class.

These adaptions allow to load the data available in the containers filled by '*MerlinImportUtils*', correctly into the database.

4.2.3 *'ImportSbmlModel' Operation*

Finally, an AIBench Operation, that executes the entire SBML Models Import tool pipeline, was created. This operation has four "In" ports that receive as input: a "Project" object corresponding to the working workspace, a "ModelSources" enumerator, a "SBMLModelLevel" enumerator (level of the imported SBML model), and the SBML file. This operation was added to the Core plug-in and is present in *merlin*'s menu task bar. In this case, no customized GUI was created, thus the input dialog was automatically generated by AIBench. This auto generated GUI allows users to select in which workspace the model should be loaded, the source of the model, the level of the SBML model as well as the file directory.

4.3 INTEGRATING NCBI BLAST+ INTO *merlin*

Similarity searches are one of the most useful and applied strategies in the reconstruction of *GSM* models, especially when annotating a genome or finding identic proteins in another organism to fill gaps. *merlin* uses several tools to perform similarity search, namely the Smith-Waterman algorithm and the EBI BLAST, to perform several tasks such as the enzymes and transporters annotations. The EBI BLAST runs remote similarity searches, in a relative short period of time, against a set of online databases. However, its availability and search time directly depend on the internet connection and web server availability. On the other hand, the Smith-Waterman algorithm, a high precision algorithm, runs the similarity searches in the user's computer; however, it is very time consuming. Thus, *merlin* lacked a resource that allowed the execution of the similarity searches locally and in a relative short period of time.

The alternative presented here was to adapt *merlin* mechanisms, to allow using NCBI BLAST+ applications, as these allow to perform similarity searches relatively quickly and locally.

Two major logical steps come out regarding the BLAST+ integration into *merlin*. The first one involves enabling *merlin* to read and parse the XML BLAST output, as that was the

selected BLAST+ output format and the parser *merlin* used was deprecated. The second one involves enabling *merlin* to convert the parsing results into structured and organized container data, able to be used any *merlin*'s tools. BLAST+ is able to run BLAST using the command line. Thus, the *'java.lang.Process'* class was used to embed this feature in *merlin*. This Java class allows executing commands and therefore to run BLAST+.

4.3.1 *NCBI BLAST Parsing*

The XML BLAST output parsing was performed by the *'NcbiBlastParser'* class. This class was implemented in collaboration with a first-year masters' student and is supported by the JAXB library. The JAXB's binding compiler XJC was used to generate 13 BlastOutput classes, using the command displayed in the grey box below.

```
>xjc -d "…\merlin\merlin-utilities\src\main\java\pt\uminho\ceb\biosystems\merlin\utilities" -p blast.ncbi_blastparser -dtd http://www.ncbi.nlm.nih.gov/dtd/NCBI_BlastOutput.dtd
```

The "-d" field specified the output directory and the "-p" field the destination package in which the generated classes were saved. Thus, all 13 BlastOutput classes were saved into the "blast.ncbi_blastparser" package inside merlin-utilities project. Furthermore, the URL for the DTD file was specified in the "-dtd" field.

The *'NCBIBlastParser'* class take as input a "String" object or a "File" object corresponding to the XML BLAST output directory or file, respectively, and also the JAXBContext. Then it creates the JAXB "Unmarshaller" object from the JAXBContext and uses it to "unmarshal" the XML file converting the XML into a full-filled Java class object, the "BlastOutput". Moreover, the "BlastOutput" is used to get all iterations data and store it in a Java "List". Therefore, the *'BlastIterationData'* class was implemented. This container class allowed to retrieve and store all the information within each XML BLAST Iteration element, such as the iteration's list of hits, the hit's list of HSPs and all the statistic parameters of the high scored HSP.

4.3.2 *BlastAlignment Class*

Once the BLAST output read and parsing component was complete, the next step was to get the parsed XML information, filter it and save it into a structured datatype able to be read by *merlin*. *merlin*'s *'AlignmentCapsule'* container class suited perfectly the storing

purpose, as it allows to organize data coming from similarity searches in a structured and readable way.

'BlastAlignment' class was implemented to filter XML BLAST output data and insert that data into *'AlignmentCapsule'* container objects. This class extends the java.Runnable abstract interface and comprises three methods:

- ***'run()'*** - this method executes three tasks, namely the construction of the BLAST output file path, the execution of BLAST and the calling for the *'buildAlignmentCapsules()'* method;
- ***'buildAlignementCapsules()'*** - this method aims building a Java "ConcurrentLinked Queue" object carrying all the created "AlignmentCapsule" objects;
- ***'getAlignmentsCapsules()'*** - method which aims retrieving the resulting "Concurrent LinkedQueue" object.

The *'buildAlignementCapsules()'* method iterates over a list of "BlastIterationData" objects retrieved from the *'NcbiBlastParser'* to get alignment specific information. In this method, the following tasks are executed to build each "AlignmentCapsule":

- Query and target sequence identifiers are processed regarding defined criteria, such as if it is a transporters search or not;
- An alignment score, S_{align}, is calculated by normalizing the alignment's raw score using the following formula:

$$S_{align} = \frac{S - S_{min}}{S_{max} - S_{min}} \tag{4}$$

  where S is the alignment's raw score and S_{max} and S_{min} are the maximum and minimum score for the aligned query sequence, respectively;

- The alignments are filtered by specific thresholds:
  - Bit Score threshold;
  - E-value threshold;
  - Query coverage threshold;
  - Normalized score threshold.
- Depending on the purpose of the BLAST alignment, the filtration can be processed in different ways. If the BLAST purpose is to identify transporters or it is undefined, the filtration is performed using the bit score, the e-value and the query coverage as criteria. Else, it will only use the alignment score (S_{align}) as threshold. Both bit score,

e-value and query coverage threshold values can be freely set by the users, otherwise the default threshold values are used.

Even though identity percent thresholds (>30%) are often used to infer homology, the E-values and bit scores can provide more accurate and reliable thresholds then the identity percent, since the last misses several homologs that would be easily detected [98]. In addition, in protein:protein alignments and for average length proteins, E-value thresholds between 10^{-6} and 10^{-10} are very useful to found homologs, even if these are not at least 30% identical. Furthermore, bit scores of 50 are also very useful for inferring homology, highlighting alignments that are almost always significant; hence being a good rule-of-thumb for inferring homology [98].

In this sense, the bit score and E-value default threshold values were set to 50 and 1E-6, respectively. Moreover, the query coverage default threshold value was set 70%, to ensure that the alignment covered most of the query sequence length.

Regarding the '*AlignmentCapsule*' containers instantiation, these were filled with relevant information regarding each selected alignment, such as the query and target sequence identifiers, the alignment e-value and the alignment bit score.

During the development of '*BlastAlignment*' class, a support class called '*AlignmentUtils*' was implemented to provide supplementary methods related with sequence alignments and similarity searches. This class contains methods that allow, for instance, checking if the BLAST+ applications are installed and getting maximum sequence alignment scores.

4.3.3 *merlin's Similarity Searches to use BLAST*

The final step in the NCBI BLAST+ integration was to rearrange *merlin*'s similarity searches mechanisms, allowing these to use this feature.

First, the '*RunSimiliratiySearch*' operation was targeted since it is the *merlin*'s core class to interact with *merlin*'s local-alignments project that executes local similarity searches. This class methods were only configured to run the Smith-Waterman algorithm using the '*PairwiseSequenceAlignment*' class. Thus, such methods were adapted to allow users to select the BLAST method. This method firstly divides the query sequences FASTA file into subfiles corresponding to the number of processor cores available in user's computer, with the purpose of running BLAST in multiple threads. A new method was implemented in '*CreateGenomeFile*' class to perform this division. After the FASTA file division, the '*BlastAlignment*' class is called for each created thread.

Then, the *merlin* tools that required similarity searches execution were identified to perform the necessary adaptations. Three tools were identified, namely: the transport proteins identification, the gaps filling operation and the GPR rules identification; implemented in the '*TransportersSimilaritySearch*', '*FindGenes*' and '*CreateGPRs*' operations, respectively. In

both operations BLAST was set as the default method to perform the similarity searches. Moreover, in both operations, the BLAST+ applications installation was verified. The '*TransportersSimilaritySearch*' operation allows the user to select the similarity searches method in its GUI, whereas the other two operations require editing the configuration files.Thus, a customized GUI is displayed, if the BLAST+ is not installed, when running the '*FindGenes*' and '*CreateGPRs*' operations. That customized GUI warns the users for the absence of a BLAST+ installation, giving the choice to continue the operation using Smith-Waterman method or to cancel the operation. Regarding the '*TransportersSimilaritySearch*', it interrupts the operation and throws a warning for users either to install BLAST+ or change method.

4.4 HOMOLOGY-BASED DRAFT MODELS RECONSTRUCTION TOOL

The reconstruction of *GSM* models is considered a very time-consuming task due to, among others, the functional annotation of their proteins. Thus, every tool that can expedite and automatize this process in a reliable way can be of extremely value and utility.

Following this premise, a whole new tool was implemented in *merlin*. This tool aims to provide a new model draft reconstruction using well-known and curated models imported into *merlin*. This kind of approach is present in literature as it has been already used by Richard A. Notebaar in AUTOGRAPH-method (AUtomatic Transfer by Orthology of Gene Reaction Associations for Pathway Heuristics) published in 2006 [90]. In AUTOGRAPH-method, this approach proved to be quite reliable and to have potential to expedite the reconstruction of metabolic networks.

The tool implemented in this work retrieves the metabolic data, the proteins annotations and the gene-protein-reaction associations from the input models. Therefore, to retrieve the proteins annotations and GPR associations the imported models' genomes are aligned against the genome of the new model, to find orthologous genes. These similarity searches are performed using BLAST and accessing the already described and implemented BLAST integrating merlin classes. Once the orthologous genes are retrieved, only those with GPR associations are selected. Then all data related to those genes, including associated reactions and proteins, is retrieved from the input models and loaded into the new model database.

The implementation of this tool comprises two major classes: the '*AutomaticDraftReconstructor*' operation class containing the overall workflow of the tool and the '*SelectAndLoadOrthologData*' class responsible for retrieving and selecting the input models' information regarding the selected orthologous genes, and to load that data into the new model database.

4.4.1 *Processing and retrieving subject model's data*

A key step in this tool development was enabling *merlin* to fetch orthologous genes' data present in the input model, namely its gene identifiers and associated reactions, and to insert it in the new model. Therefore, the '*SelectAndLoadOrthologousData*' class was implemented. This class aims at searching the selected orthologous related information in the original model (input model), to fetch and process it, namely crossing gene references between models, and to load it into the new model database. Before the actual algorithm implementation, this class creates a cross-reference map with the retrieved ortholog genes database identifiers and the new model associated gene identifers. Furthermore, three main methods within '*SelectAndLoadOrthologousData*' class were implemented:

- ***'processAndGetOrthologousData()'*** **-** this method was implemented to retrieve the gene-reaction associations for the similarity search matched orthologous. It retrieves all orthologous gene-reaction associations from one of two different ways: the reaction's gene rules or the association between gene, enzyme and reaction database tables. Orthologous genes with no gene-reaction association or whose associated reactions' gene rules are not valid for the similarity searches retrieved orthologous genes set are discarded. On the other hand, the orthologous genes whose gene-reaction associations were retrieved are saved in a map containing the gene sequence identifier and *merlin*'s database gene identifier. Furthermore, this method also performed the conversion of the old gene identifier in those associations to the new model gene identifiers using the cross-reference identifiers map. The orthologous gene names are also retrieved and associated with the corresponding new model gene, if it does not have a name already.

- ***'updateTablesDataAndGeneIdentifiers()'*** **-** this method was implemented to update new model database data, namely the reactions "in model" and its gene rules as well as the gene identifiers references in all new model database. Since this method only updates the reactions and replaces all the gene identifiers references, it is only executed when the first input model is being processed as the new model only contains the data dumped from such input model. Furthermore, this method deletes all gene-reactions associations that do not involve any of the orthologous genes matched in the similarity searches.

- ***'insertDataIntoNewModel()'*** **-** as its name suggests this method was implemented to insert data from the input model into the new model database. Firstly, it updates and transfers into the new model's database the gene-protein associations related with the orthologous genes matched in the similarity searches. Then, it verifies which of the reactions associated with the orthologous genes are not available in the new model's

database. This verification is executed by searching the reaction name identifier, thus the use of input models from the same source will considerably increase the search accuracy and consequently reduce the probability of duplicate reactions. Regarding the reactions that are already present in the database, the already existent gene-reaction association is merged with the new gene-reaction association. Furthermore, containers with relevant data, such as reaction name, equation, bounds, gene rules, associated proteins, associated pathways and stoichiometry, are retrieved for each reaction that is not present in the new model's database. Those containers are then loaded into the new model's database.

The supplementary method "*loadNewReactions()*", was also implemented to load the new reactions into new model's database. It takes the containers retrieved in the method "*insertDataIntoNewModel()*" and a map with the gene rules for each reaction as input. Then, for each reaction it retrieves the information from the associated container, builds a *merlin*'s "DatabaseReactionContainer" object and loads it into the new model's database using the existing "*loadReaction()*" method, from *merlin*'s '*DatabaseLoaders*' class. Before it proceeds with the loading, all reference identifiers regarding the reaction's related pathways, proteins, compartments, metabolites and genes are updated. Furthermore, each of those components that are not present in the new model's database are created.

4.4.2 *Implementing homology-based reconstruction pipeline*

The '*AutomaticDraftReconstructor*' class implements an AIBench operation composed by six "In" ports and executes the homology-based draft models reconstruction algorithm. No customized GUI was created for this operation thus an automatic GUI is generated by the AIBench framework. Furthermore, the requested inputs by the automatically generated input dialog comprises the new model workspace, the models that will be used as structure for the new model reconstruction, also called input models, the alignment method which can be either BLAST or Smith-Waterman and finally the similarity searches threshold parameter values, namely the bit score, the e-value and the query coverage. The algorithm implemented in this operation is presented in the pseudo-code displayed in algorithm 1.

4.5 SBML LEVEL 3 EXPORTER

As described in *merlin*'s overview on Chapter 2, *merlin* is a *GSM* models reconstruction software that supports the export of reconstructed models to the SBML format. However, previously *merlin* supported the export of models to level 2 SBML exclusively, despite the existence and the increasing expansion of level 3 SBML. Since the latest level (3) of SBML provides the generation of a document more structured, complete and with more and de-

Algorithm 1 Homology-based draft models reconstruction algorithm

```
1: Input: newModel
2:         referenceModelsSet
3:         similaritySearchesThresholds
4:         alignmentMethod
5:
6: queryGenome ← newModel.getGenome()
7:
8: for each referenceModel in referenceModelsSet do
9:
10:    sujectGenome ← referenceModel.getGenome()
11:    isFirstMerge ← false
12:
13:    if newModel.isEmpty() then
14:        referenceModel.dumpTablesToNewModel()
15:        newModel.setAllReactionsNotInModel()
16:        isFirstMerge ← true
17:    end if
18:
19:    similaritySearch ← RunSimilaritySearch(queryGenome, subjectGenome,
20:                alignmentMethod, similaritySearchesThresholds)
21:    queryOrthologousGenesMap ← similaritySearch.getOrtholousgsGenesMap()
22:
23:    process ← SelectAndLoadOrthologousData(queryOrthologousGenesMap)
24:    process.processAndGetOrthologousData()
25:
26:    if isFirstMerge then
27:        process.updateTablesDataAndGeneIdentifiers()
28:    else
29:        process.insertDataIntoNewModel()
30:    end if
31:
32:    listOfMatchedGenes ← process.getGenesWithReactionAssociations()
33:    queryGenome.remove(listOfMatchedGenes)
34:
35: end for
```

tailed information. Hence, updating *merlin*'s exporting mechanism to support the export to SBML level 3 was mandatory.

This new feature pipeline comprises two major steps: retrieving of the model's data from *merlin*'s database and the construction of the SBML document.

4.5.1 *Retrieving model's data from database*

This first step in SBML level 3 exporter pipeline was supported by two already implemented classes: the *merlin*'s '*ContainerBuilder*' class and MEWorkbench Biocomponents' '*Container*' class. The first one implements the '*IContainerBuilder*' interface and aims to retrieve the data contained in the model database, converting it into containers objects which are a workable and structured way to handle the data. Thus, all information retrieved from the model database is saved in *merlin* containers and transferred into MEWorkbench containers, after the proper processing provided by the '*Container*' class. This class receives an '*IContainerBuilder*' as input and aims to process all information from the containers built by the '*ContainerBuilder*' class. The conversion of *merlin*'s containers to MEWorkbench containers by the '*ContainerBuilder*' class allows using the '*Container*' class and its methods.

4.5.2 *Building the SBML level 3 document*

The next and final step of the SBML level 3 exporter concerns the construction of the SBML document. Thus, the '*SBMLLevel3Writer*' class was implemented to write the SBML level 3 document with the retrieved model data. '*SBMLLevel3Writer*' receives a MEWorkbench '*Container*' and accesses all its relevant data to write the SBML document. This class is supported by the JSBML library and its classes such as the "SBMLDocument", "Model" and "FBCModelPlugin". The first step in the SBML file construction is the creation of the main SBML element, "Model". It is created using a JSBML Model object, in which all children SBML elements are added after being created in the form of JSBML objects. '*SBMLLevel3Writer*' implements methods to generate JSBML objects corresponding to nine core SBML level 3 elements of the model, which are:

- **"annotation"** this SBML component is created and added to the "Model" using the '*Annotation*' class. Each "Annotation" object contains the "identifiers.org" URIs referencing the element to which it was associated. Those "identifiers.org" URIs were generated by *merlin*'s '*ExternalRef*' class.

- **"listOfGeneProducts"** this SBML level 3 element is created and added to the "Model" in the form of plugin using the '*FBCModelPlugin*' class. This class is filled with "GeneProduct" objects representing the genes of the model. Each "GeneProduct" contains

an identifier composed by the ”G_” prefix together with the gene locus tag. Furthermore, it also contains a label with the gene locus tag, the name if available and the respective annotation. At last, an ”annotation” element containing the ”identifiers.org” URIs for ”ncbigene”, ”kegg.reaction” and ”ec-code” collections is added to each ”GeneProduct”.

- **”listOfGroups”** this SBML level 3 element is created and added to the ”Model” as a ”GroupModelPlugin” object, containing a ”ListOf” object carrying all created ”Group” elements. Each ”Group” represents a specific pathway and is characterized by the following parameters: ”id”, ”name”, ”kind” and ”listOfMembers”. The parameter ”name” was set as the pathway's name. Regarding the ”listOfMembers” element, it represents the pathway's associated reactions and is composed by a set of ”Member” objects, each one characterized by a reaction identifier. Furthermore, the parameter ”kind” was defined as ”partonomy” for all ”Group” objects.

- **”listOfUnitDefinitions”** this SBML element is added to the ”Model” as ”ListOf” object containing all defined ”UnitDefinition” elements. Each ”UnitDefinition” contains a ”ListOf” object carrying ”Units” objects for the specified ”UnitDefinition”. In this work, three different Units were used, namely the ”mole”, ”gram” and ”second”.

- **”listOfObjectives”** this SBML element was created and added to the ”Model” as a ”FBCModelPlugin” containing ”Objective” objects. An ”Objective” with the ”maximize” type was adopted in this tool as default. It was composed by one ”FluxObjective” object representing the biomass, if there is one, and associated with the biomass reaction model identifier.

- **”listOfCompartments”** this SBML element is created and added to the ”Model” as a ”ListOf<Compartment>” object. For each ”Compartment” element, the ”name” and the ”id” are established. The ”id” is composed by the prefix ”C_” followed by an incremented five digits number. Furthermore, all compartments ”constant” field are set ”true”.

- **”listOfSpecies”** this SBML element is created and added to the ”Model” as a ”ListOf <Species>” object. For each ”Species” a set of attributes including the ”id”, the ”compartment” reference, the ”charge”, the ”name”, the ”chemicalFormula” and also the Boolean atributes ”boundaryCondition”, ”constant” and ”hasOnlySubstanceUnits” is assigned. Regarding the ”Species” id, it was composed by the prefix ”M_” followed by a five digits number corresponding to the metabolite's id in *merlin* database. The compartment reference was set using the corresponding ”Compartment” element id. Moreover, the Boolean attributes ”constant” and ”hasOnlySubstanceUnits” were classified as ”false” for all ”Species” in those associations and the ”boundaryCondition”

attribute only took value ”true” when the ”Species” were generated for drains reactions. Regarding the ”chemicalFormula” and ”charge” attributes, they were added to the ”Species” element through a ”FBCSpeciesPlugin” object. Finally, an ”annotation” containing the ”identifiers.org” URIs for ”kegg.compound” collection is added to each ”Species”.

- **”listOfParameters”** this SBML element is created and inserted into ”Model” as a ”ListOf<Parameter>” object. For each created ”Parameter” element, the attributes ”id”, ”name”, ”value” and the Boolean ”constant” are assigned. ”Reactions” bounds are represented by ”Parameter” objects, whose units are millimoles per gram (dry weight) per hour. Thus, at first, three default ”Parameter” objects are created containing the *merlin*'s reaction bounds default values, whose static module values are 999999.0 for the lower and upper bounds, and 0.0 for the zero bound. Then, for each processed reaction, when the bounds are different than *merlin* defaults, new ”Parameter” objects are created and added to the ”ListOf<Parameter>”. All generated ”Parameter” are defined as constant. After all reactions are processed the result ”ListOf<Parameter>” object is added to the ”Model”.

- **”listOfReactions”** the method that generates this SBML element uses the ”Reaction” object and inserts it directly into ”Model”. It aims to load all model's reactions into the SBML document. Drain reactions are also loaded into the ”listOfReactions” and new ”Species” (boundary species) for each one of them are defined. For each ”Reaction” object created, the ”id”, ”name”, ”metaid” and the Boolean ”reversible” are assigned. The assigned ”id” is generated using the prefix ”R_” followed by the automatic database identifier and the assigned ”name” is result of a concatenation of the reaction's KEGG identifier and the reaction's equation, separated by a double underscore. Regarding reaction's ”lowerFluxBound” and ”upperFluxBound”, they are added into ”Reaction” object using the *'FBCReactionPlugin'* class. Moreover, the reactants and products are added to ”Reaction” as ”SpeciesReference” objects containing the corresponding SBML ”Species” identifier, the stoichiometry and the Boolean parameter ”constant”. Furthermore, the ”geneProductAssociation” element is also added, using a ”FBCReactionPlugin” object containing a ”GeneProductAssociaton” object. The ”geneProductAssociation” elements are composed by ”geneProductRef” elements that points to specific ”geneProducts” within the SBML document. At last, an ”annotation” element containing the ”identifiers.org” URIs for ”kegg.reaction” collection is added to each ”Reaction”.

The ”Model” object can contain its own annotation too. Thus, an ”Annotation” object was added into the ”Model” object. This ”Annotation” object is composed by a list of a maximum of three ”CVTerm” objects. Each ”CVTerm” is defined by a ”Qualifier” enu-

merator value representing a Minimal Information Required In the Annotation of Models (MIRIAM) model or biological qualifier. This model created by this plugin, contains three ”CVTerm” objects. The first ”CVTerm” contains the ”hasTaxon” MIRIAM biological qualifier associated to the taxonomy identifier of the organism. The second contains the ”is” MIRIAM model qualifier associated to the model identifier. The last ”CVTerm” contains the ”isDescribedBy” MIRIAM model qualifier associated with a PubMed identifier. Only the ”CVTerm” with the organism's taxonomy identifier is always present. The other ”CVTerms” are added if the model identifier and/or PubMed identifier is available.

After the JSBML ”Model” object is created, it is inserted into a ”SBMLDocument” object. Then, the JSBML *'SBMLWriter'* class was used to generate the SBML level 3 file. This class requires a ”SBMLDocument” object and an ”OutputStream” object to write the SBML file. This writer class also allowed to set program name and version from which the model was exported. The final structure of the SBML level 3 document exported by this new tool is represented in Figure 7.

```
<?xml version='1.0' encoding='UTF-8'?>
<sbml xmlns="http://www.sbml.org/sbml/level3/version2/core"
    fbc:required="false" level="3" version="2"
    xmlns:fbc="http://www.sbml.org/sbml/level3/version1/fbc/version2"
    xmlns:html="http://www.w3.org/1999/xhtml">
  <model fbc:strict="true" id="model_sbml3" metaid="bc902641" name="SBML3_Structure">
    <fbc:listOfObjectives>
        ...
    </fbc:listOfObjectives>
    <fbc:listOfGeneProducts>
        ...
    </fbc:listOfGeneProducts>
    <listOfUnitDefinitions>
        ...
    </listOfUnitDefinitions>
    <listOfCompartments>
        ...
    </listOfCompartments>
    <groups:listOfGroups>
        ...
    </groups:listOfGroups>
    <listOfParameters>
        ...
    </listOfParameters>
    <listOfSpecies>
        ...
    </listOfSpecies>
    <listOfReactions>
        ...
    </listOfReactions>
  </model>
</sbml>
```

Figure 7.: The skeleton of the model definition expressed in the SBML level 3 exported by *merlin*'s (showing only the top-level elements of the document).

4.5.3 *SBML Level 3 Exporter Operation*

Unlike the other tools/features implemented in this work, SBML level 3 exporter did not require the implementation of a new AIBench operation. In this case, the existing '*ExportToSBML*' operation was used. However, minor adaptations were performed in this operation. Namely, two new variables, "L3V1" (level 3, version 1) and "L3V2" (level 3, version 2), were added to '*SBMLLevelVersion*' enum type used to specify the level and version of the SBML file in which the model will be exported. Lastly, a checker was added to the operation to verify the user requested SBML level and version, and a new module with the SBML level 3 exporter pipeline was added to run when the requested SBML level was 3.

4.6 OTHER *merlin*'S FEATURES IMPROVEMENT

Besides the development of the already described major tools, smaller but no less important improvements were implemented on *merlin*, aiming at a better organization, hastening processes and a more detailed and intuitive graphical interface. The most relevant small improvements will be described below.

4.6.1 *Improving FASTA files processing and uploading*

The selected organism's protein FASTA file is a key piece in *merlin*'s reconstruction process. This file contains the protein sequences encoded in the organism's genome and are used for several tasks in *merlin*, such as the enzymes annotation and the transport proteins annotation. All protein sequences are associated with the respective NCBI protein identifier, but not with the associated gene locus tag. Furthermore, despite the existence of a table within *merlin*'s database to store all the organism protein sequences, *merlin* was not using it for backup or as the main source. Instead, *merlin* uses the FASTA file to perform all tasks that require the organism's protein sequences, thus depending on the file's presence in the workspace folder. The same happens with the RNAs FASTA file.

Therefore, alterations were performed to enable *merlin* to load the genome protein and/or RNA sequences every time that a new FASTA file was added, either manually or by the automatic genome files download. Furthermore, an AIBench operation was implemented to associate the NCBI identifiers of the protein sequences with the corresponding gene locus tags. This operation searches in the downloaded GenBank annotation FASTA file (".gbff") of the organism, in UniProt database and in the NCBI's Protein database, in this order, for the gene locus tags associated to the protein sequences NCBI identifiers. This operation can be used to retrieve genes locus tags not only when the genes and the protein sequences are

being loaded to the database but also when the enzymes annotation are being integrated in the model.

This improvement allows *merlin* to use the internal database to keep the genome, thus preventing undesired issues inherent of fasta file's deletion or alteration. Moreover, with the locus tags update operation the results of the annotation, specifically the enzymes annotation, can now be integrated in the model with a standardized and homogenous genes identification.

4.6.2 *Genes view improvement*

Another improvement performed in *merlin* concerns the "genes" visualizer panel. A new column that displays the number of associated reactions to each gene was added. This counting is achieved with one of two approaches: searching for reactions that contains the gene in its gene rule, or by accessing the associations made within the database between the reactions table and the gene. This new column provides users with an instant overview on the gene-reaction associations distribution of the model.

The result layout of the *merlin*'s "genes" view is displayed in Figure 8.

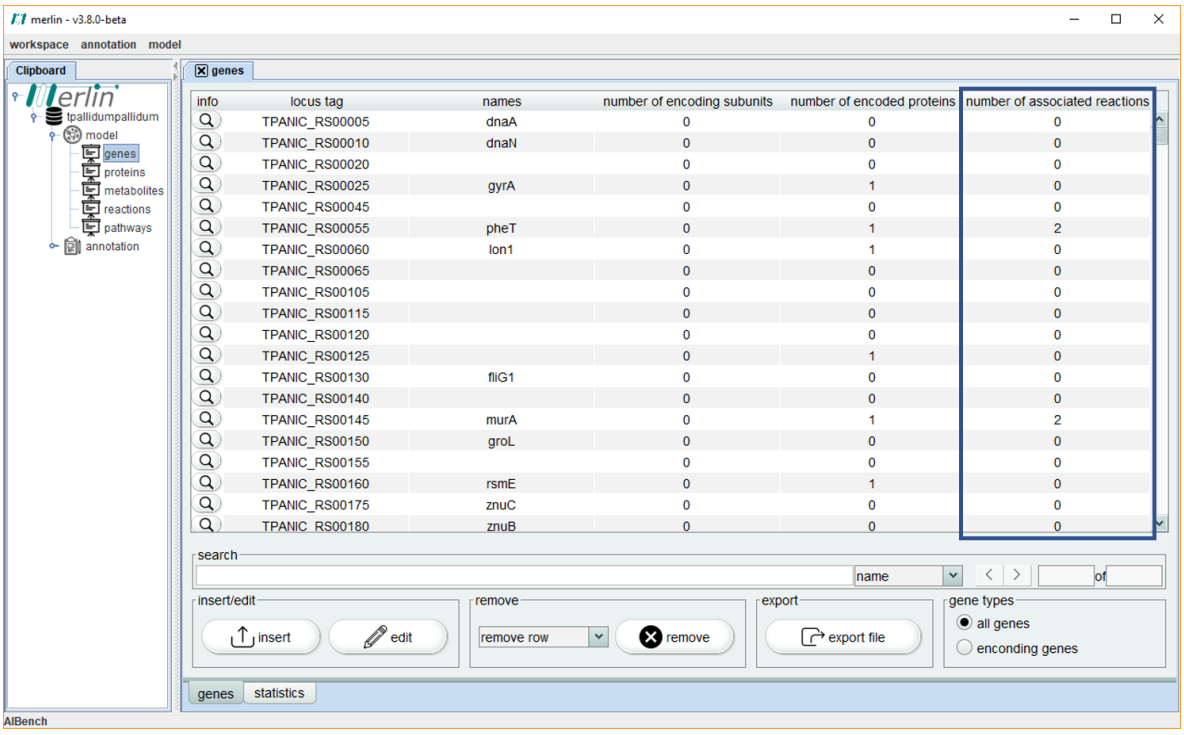


Figure 8.: *merlin*'s "genes" view layout after the addition of the "number of associated reactions" column.

4.6.3 *e-Biomass Support of ModelSEED's Compounds*

Hitherto, *merlin* exclusively used KEGG data to develop *GSM* models. Hence, supporting the integration of ModelSEED models into *merlin* created issues for tools that relied in KEGG identifiers, such as e-BiomassX. Thus, the e-BiomassX tool was updated, including its "biomassIdentifiers" configuration file that contained the default biomass components associated with the KEGG identifiers. The corresponding ModelSEED identifiers were added to each component and minor alterations were also performed in *merlin* to enable the parsing of the new "biomassIdentifiers".

5

RESULTS AND DISCUSSION

5.1 *merlin*'s IMPROVED ARCHITECTURE

The main result of this work was the implemented set of new *merlin* tools. Thus, all the developed tools were embedded on *merlin*'s framework, resulting in a reformulated and improved *merlin*'s architecture, that can be visualized in Figure 9.

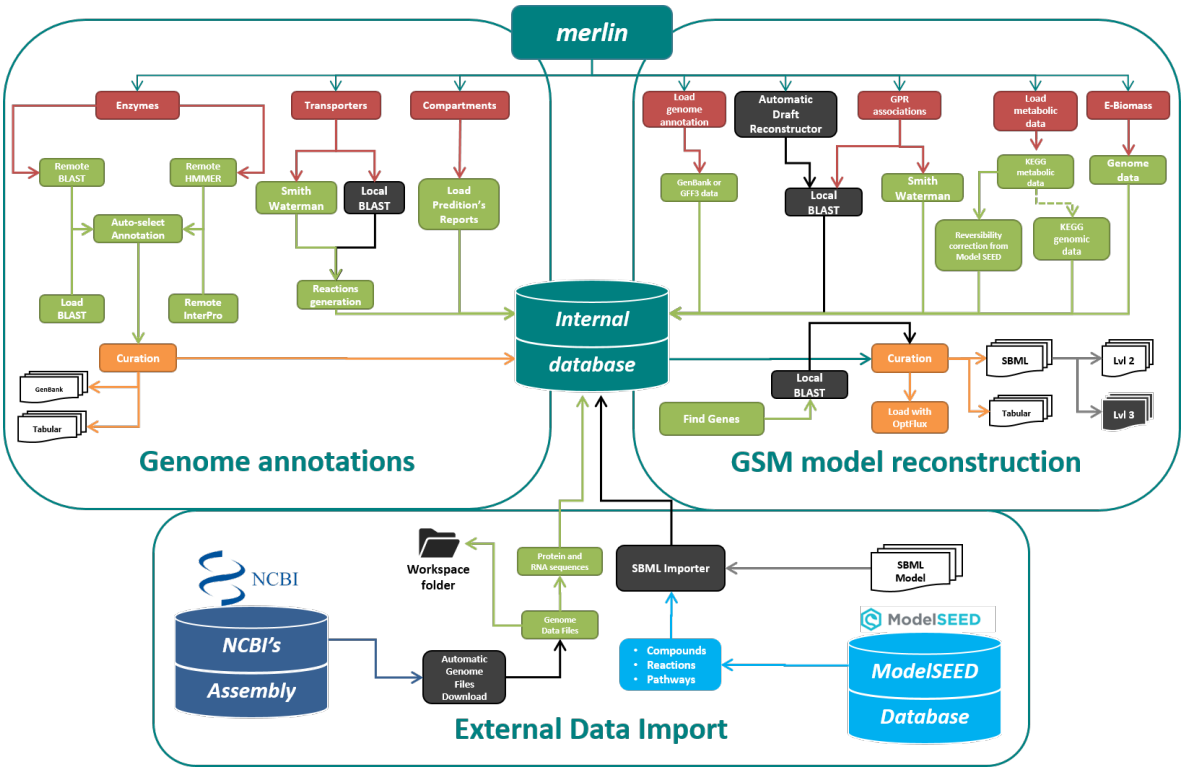


Figure 9.: A schematic representation of *merlin*'s improved architecture, as a result of the inclusion of new features (represented with color black) developed in this work.

5.2 Validating SBML model's importer and SBML level 3 exporter

A *GSM* model for *Escherichia coli* was imported and exported to SBML level 3 using the new tool developed in this project, to confirm that the implemented tools were working correctly in *merlin* and with without bearish bugs. It should be emphasized that this validation test does not replace the complete debug realized when developing both tools.

In this validation test, the published model *i*AF1260 [42] of the *Escherichia coli* str. K-12 substr. MG165 was used. This curated model was stored in a SBML level 2 version 1 file containing the ModelSEED identifiers for both metabolites and reactions, thus allowing the use of ModelSEED database files to support the loading process. Though the original model comprises BIGG identifiers, KBase's app "Integrate Imported Model into KBase Namespace" was used convert these modelSEED identifiers. The genome files were retrieved using the Genome Sequence and Annotation Data Files Automatic Download tool developed in this work. Using the taxonomy identifier 511145, it accessed the NCBI's Assembly record ASM584v2 and downloaded the genome files from RefSeq repository.

The original SBML model encompasses drains reactions, but no boundary metabolite was represented within the list of species. Furthermore, it was composed of 1668 different species elements, representing 1039 unique metabolites, 2382 reactions of which 304 were drains and one represented the biomass equation, 718 transport reactions, 3 compartments (cytoplasm, periplasm and extracellular) and 1261 genes used to build 1944 gene-reaction associations.

Firstly, the SBML model was imported using the implemented SBML model's importer, already described in the last chapter. As this SBML model contains the ModelSEED identifiers for both metabolites and reactions, additional data such as metabolite names and reactions associated KEGG identifiers, were retrieved from ModelSEED database files and added with all SBML information. This tool represents a whole new feature, allowing *merlin* to import SBML models in a relatively fast way, which allows to use *merlin* as a proxy for the curation of other software draft reconstructions, or the possibility to build draft networks using already curated models as basis for new reconstructions. However, this tool has limitations, being the most obvious one the fact that, for now, it can only retrieve additional information for models developed using ModelSEED identifiers. Otherwise, it will be strictly dependent on the amount and quality of the information available in the SBML, such as metabolites formulas, reaction pathways and reactions reversibility. Thus, the upgrading of this tool to specifically support the loading of models from other sources, such as BIGG, seems a logical improvement.

The second step of this validation was executed using the implemented SBML level 3 exporter. This tool was used to export the *E.coli* K12 model previously imported to *merlin*. Ignoring the inherent differences between the imported *E.coli* model and the one gener-

ated by *merlin* due the different SBML levels of the respective models, *merlin*'s SBML level 3 exporter was able to write a SBML file with all the components present in the original model. A "GeneProduct" and a "Reaction" component containing a "GeneProductAssociation" and present in the generated SBML level 3 file, can be visualized in Figure 10 and Figure 11 ,respectively. Furthermore, additional components were added, namely the boundary compounds present for the drain reactions. Beyond the 1668 species present in the original SBML model, 304 new species representing the boundary compounds of the drain reactions were added to make a total of 1972 species in the file. Furthermore, the output SBML level 3 model presented no errors when validated in the online validator available at `http://sbml.org/Facilities/Validator`.

```
<fbc:listOfGeneProducts xmlns:fbc="http://www.sbml.org/sbml/level3/version1/fbc/version2">
    <fbc:geneProduct fbc:id="G_b4054" fbc:label="b4054" metaid="G_b4054">
      <annotation>
        <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:bqbiol="
        http://biomodels.net/biology-qualifiers/">
        <rdf:Description rdf:about="#G_b4054">
          <bqbiol:isEncodedBy>
            <rdf:Bag>
            <rdf:li rdf:resource="http://identifiers.org/ncbigene/b4054" />
            <rdf:li rdf:resource="http://identifiers.org/kegg.reaction/R01090" />
            <rdf:li rdf:resource="http://identifiers.org/kegg.reaction/R00734" />
            <rdf:li rdf:resource="http://identifiers.org/kegg.reaction/R00694" />
            </rdf:Bag>
          </bqbiol:isEncodedBy>
          <bqbiol:is>
            <rdf:Bag />
          </bqbiol:is>
        </rdf:Description>
        </rdf:RDF>
      </annotation>
    </fbc:geneProduct>
    <fbc:geneProduct fbc:id="G_b4053" fbc:label="b4053" metaid="G_b4053">
      <annotation>
        <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:bqbiol="
```

Figure 10.: Screenshot of a "GeneProduct" component present in the *E. coli*'s SBML level 3 generated by *merlin*'s SBML 3 exporter.

Up till now, *merlin* only exported models in SBML level 2 format. Thus, the implementation of this tool seemed to be necessary and relevant, as it is an update to *merlin*'s output engine and offers the latest solution within this format. However, other improvements can always be added. Such as the generation of "identifiers.org" URIs, for metabolites and reactions, with database references beyond KEGG's or ModelSEED's. These improvements can increase reliability on the model data and reduce possible mistakes when importing or using the exported model.

```
<reaction fbc:lowerFluxBound="merlin_0_bound" fbc:upperFluxBound="merlin_defaulUB" id="R_001455"
  name="rxn05152_p__(H2O+ATP+L-Aspartate=>ADP+Phosphate+L-Aspartate+H+)" reversible="false">
  <fbc:geneProductAssociation xmlns:fbc="http://www.sbml.org/sbml/level3/version1/fbc/version2">
    <fbc:and>
      <fbc:geneProductRef fbc:geneProduct="G_b0655" />
      <fbc:geneProductRef fbc:geneProduct="G_b0652" />
      <fbc:geneProductRef fbc:geneProduct="G_b0653" />
      <fbc:geneProductRef fbc:geneProduct="G_b0654" />
    </fbc:and>
  </fbc:geneProductAssociation>
  <listOfReactants>
    <speciesReference constant="true" species="M_00016" stoichiometry="1" />
    <speciesReference constant="true" species="M_00035" stoichiometry="1" />
    <speciesReference constant="true" species="M_00033" stoichiometry="1" />
  </listOfReactants>
  <listOfProducts>
    <speciesReference constant="true" species="M_00013" stoichiometry="1" />
    <speciesReference constant="true" species="M_00034" stoichiometry="1" />
    <speciesReference constant="true" species="M_00012" stoichiometry="1" />
    <speciesReference constant="true" species="M_00023" stoichiometry="1" />
  </listOfProducts>
</reaction>
```

Figure 11.: Screenshot of a "Reaction" component present in the *E. coli*'s SBML level 3 generated by *merlin*'s SBML 3 exporter.

5.3 *Treponema pallidum* DRAFT RECONSTRUCTION

One of the goals of this work was to perform the draft *GSM* model reconstruction of *T. pallidum*, thus accessing the consistency and capability of *merlin*'s new features. For this purpose, the homology-based draft network reconstruction tool implemented in this work was used. Three curated models, with ModelSEED identifiers, were used as input for implemented tool to generate the draft network. The result draft reconstruction was evaluated, and a biomass equation was added using *merlin*'s updated e-BiomassX tool. The whole process comprising: the selection of the reference models, the draft reconstruction pipeline and the draft network overview, as well as a critic analysis of each step, is described next.

5.3.1 *Selection of the reference models*

The selection of published models to use as input for our *T. pallidum* draft reconstruction was performed using a set of published manually curated genome-scale metabolic networks reconstructions. From all species of bacteria in the models' repository, 4 gram-negative bacteria whose model's SBML was available online were selected. The selected organisms, *Yersinia pestis* CO92, *Klebsiella pneumoniae* MGH 78578, *Bordetella pertussis* Tohama I and *Piscirickettsia salmonis* A1-15972, together with *Escherichia coli* str. K-12 substr. MG165 and *T. pallidum* were submitted to a phylogenetic analysis using Clustal Omega multiple sequence alignments online tool [112] available at `https://www.ebi.ac.uk/Tools/msa/clustalo/`. The alignments were performed using each species' 16S rRNA sequence, retrieved from

the respective genome files available in the respective RefSeq's repository. Therefore, a phylogenetic tree built with the alignment results was generated by Clustal Omega and is displayed in Figure 12.



Figure 12.: Neighbour-joining phylogenetic tree generated online with the results of the multiple sequence alignments of the 16S rRNA sequences of six different organisms (TPANIC_RS01120 - *T. pallidum*; BPr01 - *B. pertussis*; KW89_RS00920 *P. salmonis*; YPOr04 *Y. pestis*; KPN_RS15635 *K. pneumoniae*; b2591 *E. coli* K12), using the Clustal Omega multiple sequence alignment program. This tree was generated without distance corrections.

Attending on the results represented by the phylogenetic tree as well as the identities on the alignments result identity matrix, supplementary table S1, the lack of close phylogenetic relations between our case-study organism and the other models is noticeable, as the distances for all organisms are considerable. However, the point of this analysis was to find phylogenetically close species to *T. pallidum*, among the available candidates. Thus, the *i*BP1870 of *B. pertussis* and *i*PSA584 model of *P. salmonis* were selected as input reference models in the draft reconstruction of *T. pallidum*. Furthermore, the published model *i*AF1260 of the *E. coli* str. K-12 substr. MG165 was also selected, as it is the most well-curated model available for a reference organism. The template models' selection was limited not only by the defined criteria (SBML model available and the organism being a gram-negative bacteria), but also for the lack of published models for closer organisms, like the ones belonging to the *Treponema* genus.

5.3.2 *Draft reconstruction output*

Using the implemented homology-based model's reconstruction tool and the selected models (*i*AF1260, *i*BP1870 and *i*PSA584) as input, a new draft genome-scale metabolic model was constructed for *T. pallidum*. This reconstruction assembled the data associated to the matched orthologous in the three input models, in this order: 188 gene-reaction associations connecting 99 genes with 188 reactions, from *E. coli*'s *i*AF1260; 76 gene-reaction associations connecting 17 genes with 76 reactions of which 23 were not in *T. pallidum reconstruction*, from *B. pertussis*'s *i*BP1870; and 13 gene-reaction associations connecting 8 genes with 13 new reactions, from *P. salmonis*'s *i*PSA584. This distribution displayed in table 5.

Therefore, this draft reconstruction comprises 124 genes of the 1003 predicted ORFs present in the FASTA file, which are associated with 224 reactions and 354 metabolites. This draft was automatically generated by *merlin* using three models as reference, and no further manual curation was performed, since the main goal was the tool implementation followed by the generation of the draft reconstruction.

Table 5.: Origin of the genes, reactions and gene-reaction associations data incorporated in the *T. pallidum* draft reconstruction.

	***E. coli* (*i*AF1260)**	***B. pertussis* (*i*BP1870)**	***P. salmonis* (*i*PSA584)**
Genes	99	17	8
New Reactions	188	23	13
Gene-reaction associations	188	76	13

The results of the draft reconstruction were then compared to the recently published first manually curated draft reconstruction of the same organism carried out by Morini and collaborators [87], and also with a draft reconstruction generated in KBase. The comparison of the three *GSM* models' characteristics is displayed in table 6. Note that the "Reactions" count presented in the table also comprises the transport reactions.

Table 6.: Main model's components of three draft reconstructions of *T. pallidum*, namely the one generated on *merlin*, a recently published one (*i*SM161) and a draft generated by KBase. * - as the model was not available, it is not possible to determine if the reaction set included transport reactions and which reactions had not GPRs.

	merlin's draft reconstruction	***i*SM161**[87]	**KBase's draft reconstruction**
Metabolites	354	277	665
Reactions	224	239	529
Transport Reactions	52	*	67
Genes	124	151	83
Reactions without GPRs	36	*	350

As shown in table 6, the first two draft models (*merlin*'s draft reconstruction and the curated draft *i*SM161) have similar size and characteristics, whereas KBase's model is much larger. According to the authors, the *i*SM161 draft reconstruction was already curated, which shows that *merlin*'s draft reconstruction is robust as the number of reactions, metabolites and genes is very similar. Furthermore, considering the genome size and literature references [29, 46, 75, 89, 103], it can be noticed that this bacterium has one of the smallest prokaryotic genomes, and thus limited biosynthetic capabilities [46], depending largely on the host to obtain the necessary nutrients. Therefore, the small number of reaction-

associated genes seems reasonable and framed by the biological context of the organism. The reactions present in the *merlin*'s draft reconstruction were grouped by pathway and the percentage of reactions assigned to each pathway was calculated. The outcome of this analysis is represented in the pie chart displayed in Figure 13. Pathways with reactions' count lower than 5 were merged into the "Other pathways" category. Note that same reactions may be present in different pathways.

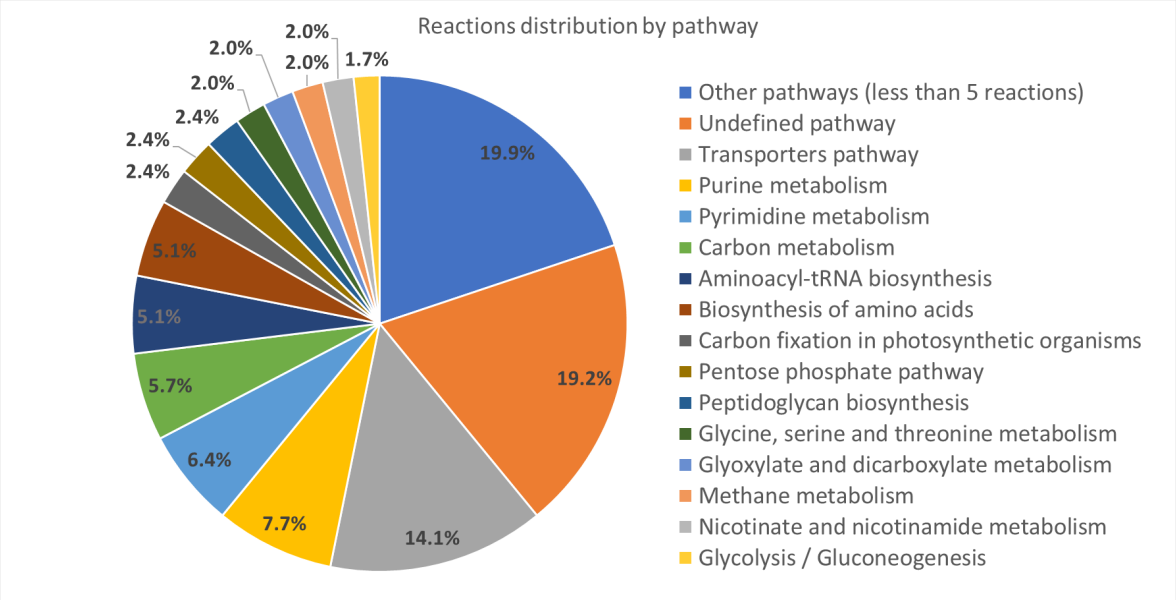


Figure 13.: Pie chart representing the relative frequency, in percentage, of reactions by pathway, in the merlin's draft reconstruction.

Transport reactions represent 14.1% of the metabolism. This is a significant fraction of the organism's reactions, which strengthens the claim that the organism limited biosynthetic capabilities, requiring considerable repertoire of transport proteins with broad substrate specificity to obtain the required nutrients from the environment [46]. Moreover, it should be considered in future work to perform an analysis of transport proteins with *TranSyT*, *merlin*'s new tool to track transport systems, as it is likely that this organism encodes in its genome several more transport proteins.

Furthermore, 19.2% of the draft model's reactions were not assigned to a pathway. This high percentage of reactions may be associated with the fact that not all identifiers on the reference models were converted to the ModelSEED's. Thus, to generate a better and consistent draft, one needs to ensure the homogeneity of the model's species and reactions identifiers.

5.3.3 *Biomass Assembly*

The biomass composition, represented in table 7, was assembled in the model and the biomass assembly equation was added to the draft model. The biomass core composition of the published model of *E. coli* K12, *i*AF1260, was used as template for the *T. pallidum* biomass assembly.

Table 7.: e-Biomass constitution assembled for the *T. pallidum* draft reconstruction and based on the *i*AF1260 *E. coli* K12 template. Each component participation in biomass generation is disposed in grams of macromolecule per gram of dry weight per hour.

Biomass component	**Coefficient (g MM . g DW-1 . h-1)**
e-Protein	0,578
e-DNA	0,033
e-RNA	0,216
e-Cofactor	0,014
e-Lipid	0,096
LPS	0,036
Murein	0,026

This process involved converting *E. coli*'s biomass composition to *T. pallidum*, which added new metabolites in the model. Hence, new transport reactions to import/uptake these molecules into the cell must be included in the model. Improving this model would involve wet-lab analysis to identify, for instance, the lipids that compose the membrane, as a better study on the organism's requirements is advised to generate a more accurate and reliable biomass assembly equation. Despite, that analysis and the draft manual curation, were not performed as it was not on the scope of this work. The full biomass content table is attached as supplementary figure S1, available in the Support Material section.

Once the biomass assembly and macromolecules polymerization equations were included in the model, the draft *GSM* model was exported to SBML level 3 format using the previously described new tool.

6

CONCLUSIONS

The work presented in this thesis aimed to expedite *merlin*'s reconstruction process and enrich it with a broad set of different tools. Thus, in this work a set of tools, including a SBML models' importer and an automatic draft generator, were developed. This new improved version of *merlin* can be accessed through the GitHub repository at `https://github.com/merlin-sysbio`. These new implemented tools confer a more automatic character to *merlin*'s model's reconstruction process and to *merlin* itself, as they succeeded to generate the draft genome-scale reconstruction of *T. pallidum*. Now, *merlin* not only is able to import draft reconstruction templates or well curated models from other sources, but to generate its own draft reconstruction to use as start point in the model reconstruction process, thus to apply its curation features reportory. The draft reconstruction of the bacterium *T. pallidum* subs. *pallidum* str. Nichols was also generated used the implemented tools, followed by the biomass assembly to finally be exported to SBML Level 3.

Therefore, globaly the main goals of this work were achieved, as a set of tools connected with each other to expedite *merlin*'s processes, as well as a draft genome-scale of *T. pallidum* is presented.

6.1 LIMITATIONS

Though several improvements were performed to *merlin* in the scope of this work, limitations are still present in this framework. Namely the new *model*'s importing tool reliance on ModelSEED's identifiers to retrieve additional data, although it may not be an issue if a very detailed SBML with a complete set of information for each of its components. Furthermore, another limitation relies on the automatic draft generator's dependency on the quality and consistency of the input models, which will likely come from the importing process, to generate a draft with as few inconsistencies as possible.

6.2 PROSPECT FOR FUTURE WORK

As the reconstruction of genome-scale models is continuously growing, so is the need to improve metabolic models' reconstruction software. *merlin* is not different, as more improvements and new tools can be performed in future works, such as:

- Update SBML models' importer to use data from other databases, such as BIGG;
- Implement a tool to retrieve published models available in online databases such as BIGG, for further import;
- Reduce *merlin*'s dependency on KEGG database.

Moreover, the curation and refinement of the *T. pallidum* draft genome-scale reconstruction, obtained in this work, seems mandatory.

BIBLIOGRAPHY

[1] A General Introduction to the E-utilities. https://www.ncbi.nlm.nih.gov/books/NBK25497/. Accessed: 2018-08-29.

[2] About MySQL. https://www.mysql.com/about/. Accessed: 2018-09-10.

[3] Basic Introduction to SBML. http://sbml.org/Basic_Introduction_to_SBML. Accessed: 2018-10-28.

[4] BLAST® Help [Internet]: BLAST Glossary. https://www.ncbi.nlm.nih.gov/books/NBK62051/. Accessed: 2018-08-20.

[5] Entrez Programming Utilities Help [Internet]. https://www.ncbi.nlm.nih.gov/books/NBK25501/. Accessed: 2018-08-29.

[6] Hibernate ORM: Your relational data. objectively. https://hibernate.org/orm/. Accessed: 2018-01-31.

[7] Java Architecture for XML Binding. https://docs.oracle.com/javase/8/docs/technotes/guides/xml/jaxb/index.html. Accessed: 2018-09-01.

[8] Java Architecture for XML Binding (JAXB). https://www.oracle.com/technetwork/articles/javase/index-140168.html. Accessed: 2018-09-05.

[9] Java JDBC API. https://docs.oracle.com/javase/8/docs/technotes/guides/jdbc/. Accessed: 2018-09-11.

[10] JDBC Overview. https://www.oracle.com/technetwork/java/overview-141217.html. Accessed: 2018-09-11.

[11] KBase: What is kbase? https://kbase.us/what-is-kbase/. Accessed: 2018-10-28.

[12] mysql-server - Linux Mint Community. https://community.linuxmint.com/software/view/mysql-server. Accessed: 2018-09-10.

[13] The E-utilities In-Depth: Parameters, Syntax and More. https://www.ncbi.nlm.nih.gov/books/NBK25499/. Accessed: 2018-08-29.

[14] R. Agren, L. Liu, S. Shoaie, W. Vongsangnak, I. Nookaew, and J. Nielsen. The RAVEN toolbox and its use for generating a genome-scale metabolic model for Penicillium chrysogenum. *PLoS computational biology*, 9(3):e1002980, 2013.

[15] S. Altschul. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402, sep 1997.

[16] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, oct 1990.

[17] A. P. Arkin, R. W. Cottingham, C. S. Henry, N. L. Harris, R. L. Stevens, S. Maslov, P. Dehal, D. Ware, F. Perez, S. Canon, et al. Kbase: The united states department of energy systems biology knowledgebase. *Nature biotechnology*, 36(7), 2018.

[18] R. K. Aziz, D. Bartels, A. A. Best, M. DeJongh, T. Disz, R. A. Edwards, K. Formsma, S. Gerdes, E. M. Glass, M. Kubal, F. Meyer, G. J. Olsen, R. Olson, A. L. Osterman, R. A. Overbeek, L. K. McNeil, D. Paarmann, T. Paczian, B. Parrello, G. D. Pusch, C. Reich, R. Stevens, O. Vassieva, V. Vonstein, A. Wilke, and O. Zagnitko. The RAST Server: rapid annotations using subsystems technology. *BMC genomics*, 9:75, feb 2008.

[19] S. A. Becker, A. M. Feist, M. L. Mo, G. Hannum, B. O. Palsson, and M. J. Herrgard. Quantitative prediction of cellular metabolism with constraint-based models: the COBRA Toolbox. *Nature Protocols*, 2, 2007.

[20] S. A. Becker and B. O. Palsson. Context-specific metabolic networks are consistent with experiments. *PLoS computational biology*, 4(5):e1000082, may 2008.

[21] L. V. BERTALANFFY. AN OUTLINE OF GENERAL SYSTEM THEORY. *The British Journal for the Philosophy of Science*, I(2):134–165, 1950.

[22] G. M. Boratyn, A. A. Schäffer, R. Agarwala, S. F. Altschul, D. J. Lipman, and T. L. Madden. Domain enhanced lookup time accelerated BLAST. *Biology Direct*, 7(1):12, 2012.

[23] F. Branco dos Santos, B. G. Olivier, J. Boele, V. Smessaert, P. De Rop, P. Krumpochova, G. W. Klau, M. Giera, P. Dehottay, B. Teusink, and P. Goffin. Probing the genome-scale metabolic landscape of bordetella pertussis, the causative agent of whooping cough. *Applied and Environmental Microbiology*, 83(21), 2017.

[24] C. Brooksbank, G. Cameron, and J. Thornton. The European Bioinformatics Institute's data resources: towards systems biology. *Nucleic acids research*, 33(Database issue):D46–53, jan 2005.

[25] A. P. Burgard, P. Pharkya, and C. D. Maranas. Optknock: a bilevel programming framework for identifying gene knockout strategies for microbial strain optimization. *Biotechnology and bioengineering*, 84(6):647–657, dec 2003.

[26] W. B. Cannon. Bodily Changes in Pain, Hunger, Fear and Rage. *Southern Medical Journal*, 22(9):870, sep 1929.

[27] R. Caspi, R. Billington, C. A. Fulcher, I. M. Keseler, A. Kothari, M. Krummenacker, M. Latendresse, P. E. Midford, Q. Ong, W. K. Ong, S. Paley, P. Subhraveti, and P. D. Karp. The MetaCyc database of metabolic pathways and enzymes. *Nucleic acids research*, oct 2017.

[28] R. Caspi, H. Foerster, C. A. Fulcher, P. Kaipa, M. Krummenacker, M. Latendresse, S. Paley, S. Y. Rhee, A. G. Shearer, C. Tissier, T. C. Walk, P. Zhang, and P. D. Karp. The MetaCyc Database of metabolic pathways and enzymes and the BioCyc collection of Pathway/Genome Databases. *Nucleic acids research*, 36(Database issue):D623–31, jan 2008.

[29] D. Čejková, M. Strouhal, S. J. Norris, G. M. Weinstock, and D. Šmajs. A Retrospective Study on Genetic Heterogeneity within Treponema Strains: Subpopulations Are Genetically Distinct in a Limited Number of Positions. *PLOS Neglected Tropical Diseases*, 9(10):e0004110, oct 2015.

[30] C. Chaouiya, D. Bérenguier, S. M. Keating, A. Naldi, M. P. van Iersel, N. Rodriguez, A. Dräger, F. Büchel, T. Cokelaer, B. Kowal, B. Wicks, E. Gonçalves, J. Dorier, M. Page, P. T. Monteiro, A. von Kamp, I. Xenarios, H. de Jong, M. Hucka, S. Klamt, D. Thieffry, N. Le Novère, J. Saez-Rodriguez, and T. Helikar. SBML qualitative models: a model representation format and infrastructure to foster interactions between qualitative modelling formalisms and tools. *BMC Systems Biology*, 7(1):135, 2013.

[31] P. Charusanti, S. Chauhan, K. McAteer, J. A. Lerman, D. R. Hyduke, V. L. Motin, C. Ansong, J. N. Adkins, and B. O. Palsson. An experimentally-supported genome-scale metabolic network reconstruction for Yersinia pestis CO92. *BMC Systems Biology*, 5(1):163, 2011.

[32] B.-S. Chen and C.-C. Wu. *Systems Biology as an Integrated Platform for Bioinformatics, Systems Synthetic Biology, and Systems Metabolic Engineering*, volume 2. 2013.

[33] N. R. Coordinators. Database resources of the National Center for Biotechnology Information. *Nucleic acids research*, nov 2017.

[34] M. P. Cortés, S. N. Mendoza, D. Travisany, A. Gaete, A. Siegel, V. Cambiazo, and A. Maass. Analysis of Piscirickettsia salmonis Metabolism Using Genome-Scale Reconstruction, Modeling, and Testing. *Frontiers in Microbiology*, 8, dec 2017.

[35] E. Demir, M. P. Cary, S. Paley, K. Fukuda, C. Lemer, I. Vastrik, G. Wu, P. D'Eustachio, C. Schaefer, J. Luciano, F. Schacherer, I. Martinez-Flores, Z. Hu, V. Jimenez-Jacinto,

G. Joshi-Tope, K. Kandasamy, A. C. Lopez-Fuentes, H. Mi, E. Pichler, I. Rodchenkov, A. Splendiani, S. Tkachev, J. Zucker, G. Gopinath, H. Rajasimha, R. Ramakrishnan, I. Shah, M. Syed, N. Anwar, O. Babur, M. Blinov, E. Brauner, D. Corwin, S. Donaldson, F. Gibbons, R. Goldberg, P. Hornbeck, A. Luna, P. Murray-Rust, E. Neumann, O. Ruebenacker, M. Samwald, M. van Iersel, S. Wimalaratne, K. Allen, B. Braun, M. Whirl-Carrillo, K.-H. Cheung, K. Dahlquist, A. Finney, M. Gillespie, E. Glass, L. Gong, R. Haw, M. Honig, O. Hubaut, D. Kane, S. Krupa, M. Kutmon, J. Leonard, D. Marks, D. Merberg, V. Petri, A. Pico, D. Ravenscroft, L. Ren, N. Shah, M. Sunshine, R. Tang, R. Whaley, S. Letovksy, K. H. Buetow, A. Rzhetsky, V. Schachter, B. S. Sobral, U. Dogrusoz, S. McWeeney, M. Aladjem, E. Birney, J. Collado-Vides, S. Goto, M. Hucka, N. Le Novere, N. Maltsev, A. Pandey, P. Thomas, E. Wingender, P. D. Karp, C. Sander, and G. D. Bader. The BioPAX community standard for pathway data sharing. *Nature biotechnology*, 28(9):935–942, sep 2010.

[36] O. Dias and I. Rocha. Systems Biology in Fungi. In R. Russell, M. Paterson, and N. Lima, editors, *Molecular Biology of Food and Water Borne Mycotoxigenic and Mycotic Fungi*, chapter 6, pages 69–92. CRC Press, 2015.

[37] O. Dias, M. Rocha, E. C. Ferreira, and I. Rocha. Reconstructing genome-scale metabolic models with merlin. *Nucleic Acids Research*, 43(8):3899–3910, apr 2015.

[38] O. Dias, M. Rocha, E. C. Ferreira, and I. Rocha. Reconstructing High-Quality Large-Scale Metabolic Models with merlin. *Methods in molecular biology (Clifton, N.J.)*, 1716:1–36, 2018.

[39] A. Dräger, N. Rodriguez, M. Dumousseau, A. Dörr, C. Wrzodek, N. Le Novere, A. Zell, and M. Hucka. User guide for jsbml. 2011.

[40] J. S. Edwards, M. Covert, and B. Palsson. Metabolic modelling of microbes: the flux-balance approach. *Environmental Microbiology*, 4, 2002.

[41] J. S. Edwards and B. O. Palsson. Systems properties of the Haemophilus influenzae Rd metabolic genotype. *The Journal of biological chemistry*, 274(25):17410–17416, jun 1999.

[42] A. M. Feist, C. S. Henry, J. L. Reed, M. Krummenacker, A. R. Joyce, P. D. Karp, L. J. Broadbelt, V. Hatzimanikatis, and B. Ø. Palsson. A genome-scale metabolic reconstruction for escherichia coli k-12 mg1655 that accounts for 1260 orfs and thermodynamic information. *Molecular Systems Biology*, 3(1), 2007.

[43] A. M. Feist and B. O. Palsson. The growing scope of applications of genome-scale metabolic reconstructions using Escherichia coli. *Nature biotechnology*, 26(6):659–667, jun 2008.

[44] R. Fleischmann, M. Adams, O. White, R. Clayton, E. Kirkness, A. Kerlavage, C. Bult, J. Tomb, B. Dougherty, J. Merrick, and E. Al. Whole-genome random sequencing and assembly of Haemophilus influenzae Rd. *Science*, 269(5223):496–512, jul 1995.

[45] M. Fondi and P. Lio. Genome-scale metabolic network reconstruction. *Methods in molecular biology (Clifton, N.J.)*, 1231:233–256, 2015.

[46] C. M. Fraser, S. J. Norris, G. M. Weinstock, O. White, G. G. Sutton, R. Dodson, M. Gwinn, E. K. Hickey, R. Clayton, K. A. Ketchum, E. Sodergren, J. M. Hardham, M. P. McLeod, S. Salzberg, J. Peterson, H. Khalak, D. Richardson, J. K. Howell, M. Chidambaram, T. Utterback, L. McDonald, P. Artiach, C. Bowman, M. D. Cotton, C. Fujii, S. Garland, B. Hatch, K. Horst, K. Roberts, M. Sandusky, J. Weidman, H. O. Smith, and J. C. Venter. Complete genome sequence of Treponema pallidum, the syphilis spirochete. *Science (New York, N.Y.)*, 281(5375):375–388, jul 1998.

[47] M. Garcia-Albornoz, S. Thankaswamy-Kosalai, A. Nilsson, L. Varemo, I. Nookaew, and J. Nielsen. BioMet Toolbox 2.0: genome-wide analysis of metabolism and omics data. *Nucleic acids research*, 42(Web Server issue):W175–81, jul 2014.

[48] J. L. Gardy, M. R. Laird, F. Chen, S. Rey, C. J. Walsh, M. Ester, and F. S. L. Brinkman. PSORTb v.2.0: expanded prediction of bacterial protein subcellular localization and insights gained from comparative proteome analysis. *Bioinformatics (Oxford, England)*, 21(5):617–623, mar 2005.

[49] M. Gavrilescu and Y. Chisti. Biotechnology - a sustainable alternative for chemical industry. *Biotechnology Advances*, 23, 2005.

[50] E. M. Gertz. Blast scoring parameters. *Manuscript*, 2005.

[51] D. Glez-Pena, M. Reboiro-Jato, P. Maia, M. Rocha, F. Diaz, and F. Fdez-Riverola. AIBench: a rapid application development framework for translational research in biomedicine. *Computer methods and programs in biomedicine*, 98(2):191–203, may 2010.

[52] T. J. Griffin, S. P. Gygi, T. Ideker, B. Rist, J. Eng, L. Hood, and R. Aebersold. Complementary profiling of gene expression at the transcriptome and proteome levels in Saccharomyces cerevisiae. *Molecular & cellular proteomics : MCP*, 1(4):323–333, apr 2002.

[53] S. P. Gygi, B. Rist, and R. Aebersold. Measuring gene expression by quantitative proteome analysis. *Current opinion in biotechnology*, 11(4):396–401, aug 2000.

[54] J. J. Hamilton and J. L. Reed. Software platforms to facilitate reconstructing genome-scale metabolic networks. *Environmental Microbiology*, 16(1):49–59, jan 2014.

[55] R. Hatti-Kaul, U. Tornvall, L. Gustafsson, and P. Borjesson. Industrial biotechnology for the production of bio-based chemicals - a cradle-to-grave perspective. *Trends in Biotechnology*, 25, 2007.

[56] C. S. Henry, M. DeJongh, A. A. Best, P. M. Frybarger, B. Linsay, and R. L. Stevens. High-throughput generation, optimization and analysis of genome-scale metabolic models. *Nature biotechnology*, 28(9):977–982, sep 2010.

[57] P. Horton, K.-J. Park, T. Obayashi, N. Fujita, H. Harada, C. J. Adams-Collier, and K. Nakai. WoLF PSORT: protein localization predictor. *Nucleic acids research*, 35(Web Server issue):W585–7, jul 2007.

[58] M. Hucka, F. T. Bergmann, S. Hoops, S. M. Keating, S. Sahle, J. C. Schaff, L. P. Smith, and D. J. Wilkinson. The systems biology markup language (sbml): language specification for level 3 version 1 core. *Journal of integrative bioinformatics*, 12(2):382–549, 2015.

[59] M. Hucka, A. Finney, H. M. Sauro, H. Bolouri, J. C. Doyle, H. Kitano, and the rest of the SBML Forum:, A. P. Arkin, B. J. Bornstein, D. Bray, A. Cornish-Bowden, A. A. Cuellar, S. Dronov, E. D. Gilles, M. Ginkel, V. Gor, I. I. Goryanin, W. J. Hedley, T. C. Hodgman, J.-H. Hofmeyr, P. J. Hunter, N. S. Juty, J. L. Kasberger, A. Kremling, U. Kummer, N. Le Novere, L. M. Loew, D. Lucio, P. Mendes, E. Minch, E. D. Mjolsness, Y. Nakayama, M. R. Nelson, P. F. Nielsen, T. Sakurada, J. C. Schaff, B. E. Shapiro, T. S. Shimizu, H. D. Spence, J. Stelling, K. Takahashi, M. Tomita, J. Wagner, and J. Wang. The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–531, mar 2003.

[60] D. R. Hyduke, N. E. Lewis, and B. O. Palsson. Analysis of omics data with genome-scale models of metabolism. *Molecular bioSystems*, 9(2):167–174, feb 2013.

[61] International Union and Nomenclature Committee of Biochemistry and Molecular Biology and E. C. E. C. Webb. *Enzyme nomenclature 1992 : recommendations of the Nomenclature Committee of the International Union of Biochemistry and Molecular Biology on the nomenclature and classification of enzymes*. San Diego : Published for the International Union of Biochemistry and Molecular Biology by Academic Press, rev. ed edition, 1992.

[62] L. Jerby, T. Shlomi, and E. Ruppin. Computational reconstruction of tissue-specific metabolic models: application to human liver metabolism. *Molecular systems biology*, 6:401, sep 2010.

[63] N. Juty, N. Le Novere, and C. Laibe. Identifiers.org and MIRIAM Registry: community resources to provide persistent identification. *Nucleic Acids Research*, 40(D1):D580–D586, jan 2012.

[64] M. Kanehisa and S. Goto. KEGG: kyoto encyclopedia of genes and genomes. *Nucleic Acids Research*, 28, 2000.

[65] S. Karlin and S. F. Altschul. Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. *Proceedings of the National Academy of Sciences*, 87(6):2264–2268, 1990.

[66] P. D. Karp, S. Paley, C. J. Krieger, and P. Zhang. An evidence ontology for use in pathway/genome databases. *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, pages 190–201, 2004.

[67] P. D. Karp, S. Paley, and P. Romero. The Pathway Tools software. *Bioinformatics (Oxford, England)*, 18 Suppl 1:S225–32, 2002.

[68] P. D. Karp, S. M. Paley, M. Krummenacker, M. Latendresse, J. M. Dale, T. J. Lee, P. Kaipa, F. Gilham, A. Spaulding, L. Popescu, T. Altman, I. Paulsen, I. M. Keseler, and R. Caspi. Pathway Tools version 13.0: integrated software for pathway/genome informatics and systems biology. *Briefings in bioinformatics*, 11(1):40–79, jan 2010.

[69] H. Kitano. *Foundations of systems biology*. The MIT Press, 2001.

[70] H. Kitano. Computational systems biology. *Nature*, 420(6912):206–210, 2002.

[71] H. Kitano. Systems biology: a brief overview. *Science (New York, N.Y.)*, 295(5560):1662–4, 2002.

[72] S. Klamt, J. Saez-Rodriguez, and E. D. Gilles. Structural and functional analysis of cellular networks with CellNetAnalyzer. *BMC Systems Biology*, 1, 2007.

[73] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. E. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. B. Hamrick, J. Grout, S. Corlay, et al. Jupyter notebooks-a publishing format for reproducible computational workflows. In *ELPUB*, pages 87–90, 2016.

[74] G. E. Krasner, S. T. Pope, et al. A description of the model-view-controller user interface paradigm in the smalltalk-80 system. *Journal of object oriented programming*, 1(3):26–49, 1988.

[75] A. Kubanov, A. Runina, and D. Deryabin. Novel Treponema pallidum Recombinant Antigens for Syphilis Diagnostics: Current Status and Future Prospects. *BioMed Research International*, 2017:1–12, 2017.

[76] A. Kumar, P. F. Suthers, and C. D. Maranas. MetRxn: a knowledgebase of metabolites and reactions spanning metabolic models and databases. *BMC bioinformatics*, 13:6, jan 2012.

[77] M. Lakshmanan, G. Koh, B. K. S. Chung, and D.-Y. Lee. Software applications for flux balance analysis. *Briefings in bioinformatics*, 15(1):108–122, jan 2014.

[78] M. Latendresse, M. Krummenacker, M. Trupp, and P. D. Karp. Construction and completion of flux balance models from pathway databases. *Bioinformatics (Oxford, England)*, 28(3):388–396, feb 2012.

[79] N. Le Novere, B. Bornstein, A. Broicher, M. Courtot, M. Donizelli, H. Dharuri, L. Li, H. Sauro, M. Schilstra, B. Shapiro, J. L. Snoep, and M. Hucka. BioModels Database: a free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems. *Nucleic Acids Research*, 34, 2006.

[80] Y.-C. Liao, T.-W. Huang, F.-C. Chen, P. Charusanti, J. S. J. Hong, H.-Y. Chang, S.-F. Tsai, B. O. Palsson, and C. A. Hsiung. An Experimentally Validated Genome-Scale Metabolic Reconstruction of Klebsiella pneumoniae MGH 78578, iYL1228. *Journal of Bacteriology*, 193(7):1710–1717, apr 2011.

[81] Y.-C. Liao, M.-H. Tsai, F.-C. Chen, and C. A. Hsiung. GEMSiRV: a software platform for GEnome-scale metabolic model simulation, reconstruction and visualization. *Bioinformatics (Oxford, England)*, 28(13):1752–1758, jul 2012.

[82] J. Linwood and D. Minter. *Beginning Hibernate*. Apress, Berkeley, CA, 2010.

[83] Z. Lu, D. Szafron, R. Greiner, P. Lu, D. S. Wishart, B. Poulin, J. Anvik, C. Macdonell, and R. Eisner. Predicting subcellular localization of proteins using machine-learned classifiers. *Bioinformatics (Oxford, England)*, 20(4):547–556, mar 2004.

[84] V. Mihalcea, S. Ebersole, A. Boriero, G. Morling, G. Badner, C. Cranford, E. Bernard, B. Meyer, H. Ferentschik, G. King, C. Bauer, M. R. Andersen, K. Maesen, R. Vansa, and L. Jacomet. Hibernate ORM 5 . 2 . 12 . Final User Guide, 2018.

[85] J. Monk, J. Nogales, and B. O. Palsson. Optimizing genome-scale network reconstructions. *Nature Biotechnology*, 32(5):447–452, may 2014.

[86] A. Morgulis, G. Coulouris, Y. Raytselis, T. L. Madden, R. Agarwala, and A. A. Schäffer. Database indexing for production MegaBLAST searches. *Bioinformatics*, 24(16):1757–1764, aug 2008.

[87] S. Morini, I. Casini, T. M. Hamm, K. Nieselt, and A. Drager. Metabolic Network Reconstruction of Treponema Pallidum Spp. Pallidum. In *COBRA 2018 - 5th Conference on Constraint-Based Reconstruction and Analysis.*, Seattle WA, USA, 2018.

[88] R. Nair and B. Rost. Mimicking cellular sorting improves prediction of subcellular localization. *Journal of molecular biology*, 348(1):85–100, apr 2005.

[89] A. A. T. Naqvi, M. Shahbaaz, F. Ahmad, and M. I. Hassan. Identification of Functional Candidates amongst Hypothetical Proteins of Treponema pallidum ssp. pallidum. *PLOS ONE*, 10(4):e0124177, apr 2015.

[90] R. A. Notebaart, F. H. J. van Enckevort, C. Francke, R. J. Siezen, and B. Teusink. Accelerating the reconstruction of genome-scale metabolic networks. *BMC Bioinformatics*, 7(1):296, jun 2006.

[91] M. A. Oberhardt, B. Ø. Palsson, and J. A. Papin. Applications of genome-scale metabolic reconstructions. *Molecular Systems Biology*, 5(320):1–15, nov 2009.

[92] M. A. Oberhardt, J. Puchalka, K. E. Fryer, V. A. P. Martins dos Santos, and J. A. Papin. Genome-scale metabolic network analysis of the opportunistic pathogen Pseudomonas aeruginosa PAO1. *Journal of bacteriology*, 190(8):2790–2803, apr 2008.

[93] B. G. Olivier and F. T. Bergmann. The systems biology markup language (sbml) level 3 package: Flux balance constraints. *Journal of integrative bioinformatics*, 12(2):660–690, 2015.

[94] S. Opdam, A. Richelle, B. Kellman, S. Li, D. C. Zielinski, and N. E. Lewis. A Systematic Evaluation of Methods for Tailoring Genome-Scale Metabolic Models. *Cell Systems*, 4(3):318–329.e6, mar 2017.

[95] R. Overbeek, T. Begley, R. M. Butler, J. V. Choudhuri, H.-Y. Chuang, M. Cohoon, V. de Crecy-Lagard, N. Diaz, T. Disz, R. Edwards, M. Fonstein, E. D. Frank, S. Gerdes, E. M. Glass, A. Goesmann, A. Hanson, D. Iwata-Reuyl, R. Jensen, N. Jamshidi, L. Krause, M. Kubal, N. Larsen, B. Linke, A. C. McHardy, F. Meyer, H. Neuweger, G. Olsen, R. Olson, A. Osterman, V. Portnoy, G. D. Pusch, D. A. Rodionov, C. Ruckert, J. Steiner, R. Stevens, I. Thiele, O. Vassieva, Y. Ye, O. Zagnitko, and V. Vonstein. The subsystems approach to genome annotation and its use in the project to annotate 1000 genomes. *Nucleic acids research*, 33(17):5691–5702, 2005.

[96] S. Pabinger, R. Rader, R. Agren, J. Nielsen, and Z. Trajanoski. MEMOSys: Bioinformatics platform for genome-scale metabolic models. *BMC systems biology*, 5:20, jan 2011.

[97] K. R. Patil, I. Rocha, J. Forster, and J. Nielsen. Evolutionary programming as a platform for in silico metabolic engineering. *BMC Bioinformatics*, 6, 2005.

[98] W. R. Pearson. An Introduction to Sequence Similarity (Homology) Searching. *Current Protocols in Bioinformatics*, 42(1):3.1.1–3.1.8, jun 2013.

[99] D. Petranovic and G. N. Vemuri. Impact of yeast systems biology on industrial biotechnology. *Journal of Biotechnology*, 144(3):204–211, 2009.

[100] H. Pětrošová, P. Pospíšilová, M. Strouhal, D. Čejková, M. Zobaníková, L. Mikalová, E. Sodergren, G. M. Weinstock, and D. Šmajs. Resequencing of Treponema pallidum ssp. pallidum Strains Nichols and SS14: Correction of Sequencing Errors Resulted in Increased Separation of Syphilis Treponeme Subclusters. *PLoS ONE*, 8(9):e74319, sep 2013.

[101] E. Pitkanen, P. Jouhten, J. Hou, M. F. Syed, P. Blomberg, J. Kludas, M. Oja, L. Holm, M. Penttila, J. Rousu, and M. Arvas. Comparative genome-scale reconstruction of gapless metabolic networks for present and ancestral species. *PLoS computational biology*, 10(2):e1003465, feb 2014.

[102] J. D. Radolf. Treponema. 1996.

[103] J. D. Radolf, R. K. Deka, A. Anand, D. Smajs, M. V. Norgard, and X. F. Yang. Treponema pallidum, the syphilis spirochete: making a living as a stealth pathogen. *Nature reviews. Microbiology*, 14(12):744–759, dec 2016.

[104] I. Rocha, P. Maia, P. Evangelista, P. Vilaça, S. Soares, J. P. Pinto, J. Nielsen, K. R. Patil, E. C. Ferreira, and M. Rocha. OptFlux: an open-source software platform for in silico metabolic engineering. *BMC Systems Biology*, 4(1):45, 2010.

[105] M. H. J. Saier. A functional-phylogenetic classification system for transmembrane solute transporters. *Microbiology and molecular biology reviews : MMBR*, 64(2):354–411, jun 2000.

[106] E. Schaudinn, Fritz Richard; Hoffmann. Vorläufiger Bericht über das Vorkommen von Spirochaeten in syphilitischen Krankheitsprodukten und bei Papillomen. *Berlin: Verlag von Julius Springer.*, 22:527–534, 1905.

[107] J. Schellenberger, J. O. Park, T. M. Conrad, and B. O. Palsson. BiGG: a Biochemical Genetic and Genomic knowledgebase of large scale metabolic reconstructions. *BMC bioinformatics*, 11:213, apr 2010.

[108] J. Schellenberger, R. Que, R. M. T. Fleming, I. Thiele, J. D. Orth, A. M. Feist, D. C. Zielinski, A. Bordbar, N. E. Lewis, S. Rahmanian, J. Kang, D. R. Hyduke, and B. O. Palsson. Quantitative prediction of cellular metabolism with constraint-based models: the COBRA Toolbox v2.0. *Nature protocols*, 6(9):1290–1307, aug 2011.

[109] I. Schomburg, A. Chang, O. Hofmann, C. Ebeling, F. Ehrentreich, and D. Schomburg. BRENDA: a resource for enzyme data and metabolic information. *Trends in biochemical sciences*, 27(1):54–56, jan 2002.

[110] D. Segre, D. Vitkup, and G. M. Church. Analysis of optimality in natural and perturbed metabolic networks. *PNAS*, 99, 2002.

[111] T. Shlomi, O. Berkman, and E. Ruppin. Regulatory on/off minimization of metabolic flux changes after genetic perturbations. *PNAS*, 102, 2005.

[112] F. Sievers, A. Wilm, D. Dineen, T. J. Gibson, K. Karplus, W. Li, R. Lopez, H. McWilliam, M. Remmert, J. Söding, J. D. Thompson, and D. G. Higgins. Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. *Molecular systems biology*, 7:539, 2011.

[113] N. Swainston, K. Smallbone, P. Mendes, D. Kell, and N. Paton. The SuBliMinaL Toolbox: automating steps in the reconstruction of metabolic networks. *Journal of integrative bioinformatics*, 8(2):186, nov 2011.

[114] I. Thiele and B. O. Palsson. A protocol for generating a high-quality genome-scale metabolic reconstruction. *Nature protocols*, 5(1):93–121, jan 2010.

[115] T. Vignolini, A. Mengoni, and M. Fondi. Template-Assisted Metabolic Reconstruction and Assembly of Hybrid Bacterial Models. *Methods in molecular biology (Clifton, N.J.)*, 1716:177–196, 2018.

[116] P. Vilaca, P. Maia, H. Giesteira, I. Rocha, and M. Rocha. Analyzing and Designing Cell Factories with OptFlux. *Methods in molecular biology (Clifton, N.J.)*, 1716:37–76, 2018.

[117] T. Weber and H. U. Kim. The secondary metabolite bioinformatics portal: Computational tools to facilitate synthetic biology of secondary metabolite production. *Synthetic and systems biotechnology*, 1(2):69–79, jun 2016.

[118] G. M. Weinstock, J. M. Hardham, M. P. McLeod, E. J. Sodergren, and S. J. Norris. The genome of Treponema pallidum : new light on the agent of syphilis. *FEMS Microbiology Reviews*, 22(4):323–332, oct 1998.

[119] N. Y. Yu, J. R. Wagner, M. R. Laird, G. Melli, S. Rey, R. Lo, P. Dao, S. C. Sahinalp, M. Ester, L. J. Foster, and F. S. L. Brinkman. PSORTb 3.0: improved protein subcellular localization prediction with refined localization subcategories and predictive capabilities for all prokaryotes. *Bioinformatics (Oxford, England)*, 26(13):1608–1615, jul 2010.

[120] Z. Zhang, S. Schwartz, L. Wagner, and W. Miller. A Greedy Algorithm for Aligning DNA Sequences. *Journal of Computational Biology*, 7(1-2):203–214, feb 2000.

[121] H. Zur, E. Ruppin, and T. Shlomi. iMAT: an integrative metabolic analysis tool. *Bioinformatics (Oxford, England)*, 26(24):3140–3142, dec 2010.

A

SUPPORT MATERIAL

Table S1.: Clustal Omega alignments result identity matrix.

	TPANIC_RS01120	**BPr01**	**KW89_RS00920**	**YPOr04**	**KPN_RS15635**	**b2591**
TPANIC_RS01120	**100**	74.72	76.12	74.71	74.4	74.87
BPr01	74.72	**100**	81.62	81.9	81.83	81.82
KW89_RS00920	76.12	81.62	**100**	84.33	84.52	85.45
YPOr04	74.71	81.9	84.33	**100**	94.95	94.22
KPN_RS15635	74.4	81.83	84.52	94.95	**100**	96.23
b2591	74.87	81.82	85.45	94.22	96.23	**100**