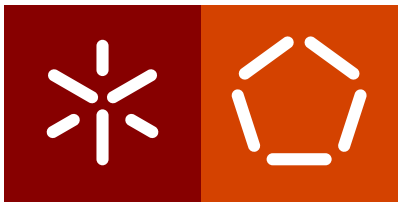
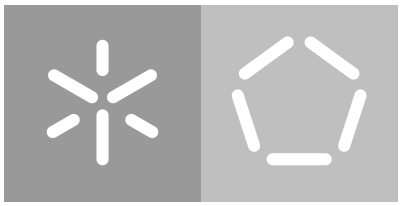


Universidade do Minho
Escola de Engenharia
Departamento de Informática

Davide Rafael Santos Lagoa

Development of Bioinformatics tools for the classification of transporter systems

December 2018



Universidade do Minho
Escola de Engenharia
Departamento de Informática

Davide Rafael Santos Lagoa

Development of Bioinformatics tools for the classification of transporter systems

Master dissertation
Master Degree in Bioinformatics

Dissertation supervised by
Oscar Manuel Lima Dias

December 2018

ACKNOWLEDGEMENTS

Consciente de que este não foi um trabalho individual, são diversas as pessoas às quais devo os mais profundos agradecimentos. Em primeiro lugar, ao meu orientador, Prof. Oscar Dias, obrigado pelo tempo dispensado na partilha de conhecimentos, pela confiança nas minhas capacidades, pela preocupação, disponibilidade e amizade demostradas ao longo de todo este tempo. Agradeço ainda a oportunidade de desenvolver este trabalho e de o integrar na KBase, a quem devo também um agradecimento, pois é bastante gratificante sentir a valorização do nosso trabalho.

Ao Amaro, colega e amigo nesta jornada, pelas largas discussões e contributos para esta dissertação, se hoje consegui alcançar um trabalho com qualidade, em parte a ti o devo.

À Catarina, que desde Vila Real partilhou todo este percurso comigo. Obrigado por toda a amizade e por tantas vezes me arrancares de casa para ir beber aquele balde ou às inúmeras jantaradas. Que nunca acabem...

Ao Ivo, que apesar de ter divergido da vida académica, depois de tantos anos continua ser o melhor amigo que se podia desejar e continua a ter sempre as melhores histórias para contar.

A ti Bruna, que me trouxeste para o mundo da Bioinformática. O teu contributo e apoio em todas as situações tornam todos os percursos mais fáceis. O meu maior obrigado!

Por fim, à minha família. Aos meus avós que me estão sempre a perguntar quando vou acabar a escola. À minha irmã que tanto me ajudou até aqui (vai ser um menino). A vocês, Pai e Mãe, que de tanto abdicaram para que tudo isto fosse possível. Ficarei eternamente agradecido por tudo o que fizeram e fazem por mim.

ABSTRACT

The Transport Systems Tracker (*TranSyT*) is a new approach to the problem of identifying genome-wide transmembrane transport systems, annotating these with reactions. *TranSyT* is the next iteration of *TRIAGE* (29), though more efficient and designed to overcome its limitations. This new approach still relies on the TCDB (118) to perform the annotation of transporters systems; however, *TranSyT* automatically retrieves and processes information from this source.

The information available in TCDB allows determining which metabolites are carried by each transporter system and the respective reactions. In *TranSyT*, these metabolites are assigned with identifiers and hierarchies through Biosynth (81), which combines information retrieved from several sources such as ModelSEED, KEGG, MetaCyc and BiGG. *TranSyT* generates new transport reactions using automatic text processing to determine the direction (in/out or out/in), reversibility and most suitable transport type (e.g. symport, antiport, *etc)* for each reaction retrieved from TCDB. All information is stored in a Neo4j graph database. The identification of the genes encoding transporter systems is the only module with user's interaction, as the database is available for remote access, without the need of storing it in the users' machine. Users start the identification of the transporters after loading a genome and the respective taxonomy identifier. *TranSyT* uploads the genome to its remote service and performs the homology search using BLAST against all records available in TCDB. Afterwards, it selects the correct Transporter Classification (TC) family to annotate each transport system and associates transport reactions to the encoded protein, creating Gene-Protein-Reaction (GPR) associations. The integration with genome-scale models filters metabolites not available in the reconstructed network. The localization of the reactions can be inferred from third-party tools such as PSORTb 3.0 or LocTree3. Other tools that determine whether the genes encoding transport proteins have transmembrane domains, allow assigning confidence levels to the systems.

The *i*AF1260 *Escherichia coli* model (38) was used to validate the developed framework. *TranSyT* was able to automatically create reactions for nearly 75% of the metabolites described in *i*AF1260 model transporters. Moreover, it allowed identifying transport reactions incorrectly assigned to genes that do not encode reactions transporting such metabolites.

TranSyT is an open-source Java™ software available at `https://gitlab.bio.di.uminho.pt/TranSyT`, currently being implemented in *merlin* and KBase.

Keywords: *TranSyT*, Transporter Systems, *TRIAGE*, Transport Reactions, Genome-Scale Metabolic Models, Transporters Inference, Bioinformatics

RESUMO

O Transport Systems Tracker (*TranSyT*) é uma nova abordagem ao problema de identificação de sistemas de transporte transmembranares em genomas. O *TranSyT* é a próxima iteração da ferramenta *TRIAGE* (29), embora mais eficiente e desenvolvido para ultrapassar as limitações existentes. Tal como o *TRIAGE*, ainda depende da TCDB (118) para a anotação de sistemas, no entanto, o *TranSyT* recolhe e processa automaticamente a informação desta fonte.

A informação disponível na TCDB permite determinar quais os metabolitos que são transportados por cada sistema de transporte e quais as respectivas reacções. No *TranSyT*, identificadores e descendentes hierárquicos são atribuídos aos metabolitos utilizando o Biosynth (81), que combina informação de várias fontes, como o ModelSEED, KEGG, MetaCyc e BiGG. O *TranSyT* gera reacções de transporte utilizando processamento automático de texto para determinar a direcção (interior/exterior ou exterior/interior), reversibilidade, e o tipo de transporte adequado (e.g. *symport*, *antiport*, *etc)* para cada reacção recolhida da TCDB. Todas as informações recolhidas são guardadas numa base de dados de grafos Neo4j. A identificação de sistemas transportadores codificados por genes é o único módulo com o qual o utilizador tem interacção, visto que a base de dados está disponível remotamente, não havendo a necessidade de armazená-la na máquina do utilizador. Ao carregar um genoma e o respectivo identificador taxonómico, o utilizador é capaz de começar a identificação dos transportadores. O *TranSyT* carrega o genoma para o seu serviço remoto e executa a procura de homologias utilizando BLAST (3) contra todos os registos da TCDB. De seguida, selecciona a família TC correcta para anotar cada sistema transportador e associa reacções de transporte à proteína codificada, criando associações Gene-Proteína-Reacção (GPR). A integração com o modelo à escala genómica usada para filtrar metabolitos que não estão indisponíveis na rede. A localização dos sistemas pode ser inferida utilizando ferramentas de terceiros, tais como PSORTb 3.0 ou LocTree3. Outras ferramentas que determinam se os genes que codificam proteínas de transporte têm domínos transmembranares, permitem a atribuição de graus de confiança às classificações.

O modelo *i*AF1260 da *Escherichia coli* (38) foi utilizado para validar a plataforma desenvolvida. O *TranSyT* foi capaz de criar automaticamente reacções para aproximadamente 75% dos metabolitos associados a transportadores do modelo. Permitiu ainda identificar reacções de transporte associadas a genes que não codificam reacções que transportam os metabolitos que lhes são atribuídos.

TranSyT é um software *open-source* Java™ disponível em `https://gitlab.bio.di.uminho.pt/TranSyT`, actualmente a ser implementado no *merlin* e na KBase.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

E. COLI *Escherichia coli*

TRIAGE Transport Reactions Annotation and Generation

TRANSYT Transport Systems Tracker

MERLIN Metabolic models reconstruction using genome-scale information

A Adenine

ABC ATP-binding cassette

ATP adenosine triphosphate

BIGG Biochemical, Genetic and Genomic

BLAST Basic Local Alignment Search Tool

BRENDA BRaunschweig ENzyme DAtabase

C Cytosine

CEB Centre of Biological Engineering

CHEBI Chemical Entities of Biological Interest

CPU Central Process Unit

DNA deoxyribonucleic acid

EC Enzyme Commission

ETC electron transport chain

FBA Flux Balance Analysis

FVA Flux Variability Analysis

G Guanine

GC Gas Chromatography

GPR Gene-Protein-Reaction

GSM Genome-Scale Metabolic

GUI Graphical User Interface

H2 H2 Database Engine

IUBMB International Union of Biochemistry and Molecular Biology

JSON JavaScript Object Notation

KBASE The Department of Energy Systems Biology Knowledgebase

KEGG Kyoto Encyclopedia of Genes and Genomes

KO KEGG Orthology

LMOMA Linear MOMA

ME Metabolic Engineering

MFA Metabolic Flux Analysis

MN Metabolic Network

MOMA Method of Optimization of Metabolic Adjustment

MS Mass Spectrometry

NCBI The National Center for Biotechnology Information

NGS Next Generation Sequencing

NMR Nuclear Magnetic Resonance

OPM Orientations of Proteins in Membranes

OS Operating System

PEP phosphoenolpyruvate

PGDBS Pathway/Genome Databases

PHMMS profile Hidden Markov Models

PTS phosphotransferase system

RAM Random Access Memory

RAST Rapid Annotation using Subsystem Technology

RAVEN Reconstruction, Analysis and Visualization of Metabolic Networks

RNA ribonucleic acid

ROOM Regulatory On/Off Minimization

SB Systems Biology

SBML Systems Biology Markup Language

SN Signaling Network

SVM Support Vector Machines

T Thymine

TC Transporter Classification

TCDB Transporter Classification Database

TCG Transport Candidate Gene

TM Transmembrane

TMA Transmembrane α-helical

TMB Transmembrane β-barrel

TMD Transmembrane Domains

TN Transcriptional Network

TR IDENTIFIER Transport Reaction identifier

TRANSATH Transporters via Annotation Transfer by Homology

UNIPROT Universal Protein Resource

URL Uniform Resource Locator

VMS Virtual Machines

1

INTRODUCTION

1.1 CONTEXT AND MOTIVATION

In the last decades, the improvement of Bioinformatics tools and the increasingly vast biological information available in databases, allowed the reconstruction and application of Genome-Scale Metabolic (GSM) models. These models are now seen as tools that allow obtaining a better understanding of cellular metabolism, and promote the development of metabolic engineering strategies, which reduce the high cost and time consumption on a larger batch of experiments (31).

The reconstruction of GSM models involves the assembly of all biochemical reactions that take place inside of a target organism. Information about the compounds and the enzymes involved in these reactions is initially retrieved from literature and biological databases (e.g. The National Center for Biotechnology Information (NCBI) (24), Universal Protein Resource (UniProt) (23), etc). However, it is important to notice that the reconstruction of these networks is a laborious and iterative process, as not only it is important to have the information about the compounds involved in the reactions, but also the correct stoichiometry, reversibility and cellular location, for each reaction (31). Lately, the reconstruction of GSM models has been simplified with the use of tools, such as Metabolic models reconstruction using genome-scale information (*merlin*) (31) and ModelSEED (54). However, most of these tools fail in the automatic identification and characterization of transport reactions. These have to be manually added or removed, by the user, based on experimental data and literature evidences, decreasing the quality of the models. Tools like ModelSEED, despite providing transport reactions, are not able to associate these to genes. Therefore, the identification of genes encoding transport proteins and the metabolites transported by those should be improved, as these are important to reconstruct accurate GSM models, both for eukaryotes and prokaryotes (38).

Metabolic models reconstruction using genome-scale information (*merlin*) is a Java™ application that allows reconstructing GSM models for any sequenced organism. It allows performing the functional genomic annotation of its entire genome (31), and provides a draft model, which can be easily curated within the framework, and later exported to the

Systems Biology Markup Language (SBML) (61) standard format. Transport Reactions Annotation and Generation (*TRIAGE*) (29), a tool currently embedded in *merlin*, performs the identification of membrane transport systems and automatically generates transport reactions for every metabolite transported by those carriers (29). Reactions generated by *TRIAGE* can be directly integrated with GSM models, as all metabolites have Kyoto Encyclopedia of Genes and Genomes (KEGG) and/or Chemical Entities of Biological Interest (ChEBI) identifiers. To our knowledge this tool is the only able to identify and generate such reactions. *TRIAGE*'s pipeline for Transport Candidate Gene (TCG)s identification is very strict, since it combines several tools to decrease the number of false positives, which implies that a negative prediction in one of the modules will exclude the gene of the membrane transport systems encoding genes set.

One of the main disadvantages of the current architecture of *TRIAGE* is that it is only available while embedded in *merlin*'s core projects, which limits its accessibility. To exploit the full potential of this tool, it is of paramount importance to provide *TRIAGE* in a standalone software, capable of generating results from a specific input. To accomplish this goal, the internal structure of the database will be updated. During development, the software was completely redesigned, implemented from scratch and followed a different philosophy, completely independent from *merlin*; hence this software should have a new designation. Here, *TranSyT*, a new software for the annotation of transport systems and generation of transport reactions is presented. The development of this software, will ultimately allow its inclusion in other platforms, such as *merlin* and The Department of Energy Systems Biology Knowledgebase (KBase).

1.2 OBJECTIVES

The main goal of this work is to improve *TRIAGE*, a Bioinformatics tool that performs the identification and annotation of transport systems. To fulfill this objective, the following tasks will be performed:

- Improve *TRIAGE*'s method of automatically generating reactions;
- Relax *TRIAGE* rigorous rules to improve the identification of transport systems;
- Include the identification of other membrane structures namely β-barrels in the tool's algorithm;
- Create a standalone application, allowing its inclusion in other software, such as KBase;

• Compare *TRIAGE*'s internal transport systems database to the new updated version of Transporter Classification Database (TCDB);

• Integration of the new tool in KBase.

1.3 STRUCTURE OF THE DOCUMENT

chapter 2 - state of the art

This chapter presents a brief introduction to the reconstruction of GSM models and the problematic of annotating and including transport reactions in these models. This section consists of an overview of current methods for reconstruction of genome-scale metabolic models, and methods for automatic annotation of transporter systems, such as *merlin*, and a detailed description of *TRIAGE*. Available methodologies for the identification and annotation of transmembrane transport proteins, including the existing methods for prediction of transmembrane α-helices and β-barrel structures, are also discussed in this chapter. This chapter closes with a brief description of the case studies.

chapter 3 - methods

After providing context regarding the application environment and functionalities of interest, the methodologies that steered *TranSyT*'s architectural design are described in this chapter. The methods used to perform the study that allowed assessing tools developed to predict transmembrane β-barrels are described in this section. Finally, the process developed for validating TCDB's recent updates, regarding transported compounds in each transport system, is also detailed at this stage.

chapter 4 - software development

This chapter describes all improvements and new implementations performed in *merlin*, as well as a detailed description of *TranSyT*'s software implementation. An explanation of the methods used to validate *TranSyT* is also available in this section.

chapter 5 - results and discussion

Results regarding *merlin*'s updates and new features are presented in this chapter. The results of the study regarding β-barrels predictors and TCDB data validation, described in chapter "Methods", are shown and discussed here. A section with the outcome of the development of a graphical user interface is provided, as well as the results of *TranSyT*'s validation, and the main differences between *TRIAGE* and *TranSyT*.

chapter 6 - conclusion

Finally, all conclusions of the work developed are summarized at this chapter, and prospects for the future are also discussed. An overview of the outcomes of this work is also provided.

2

STATE OF THE ART

2.1 OMICS IN SYSTEMS BIOLOGY

During 1953, while working in the Cavendish Laboratory in Cambridge, James Watson and Francis Crick made a discovery that would forever change our perception of life: the structure of the deoxyribonucleic acid (DNA) molecule. Considered one of the biggest discoveries of the 20th century, this finding was responsible for the beginning of a new era in Science (73, p. 5). The DNA is an anti-parallel double helix molecule, composed by four organic compounds, nucleobases, that complement each other and are linked together by hydrogen bounds: Adenine (A) - Thymine (T), and Guanine (G) - Cytosine (C) (73, p. 8). Located inside the nucleolus of eukaryotic organisms and in the nucleoid (poorly demarcated area that lacks of membrane's boundaries to separate the genetic material from the cytoplasm) of prokaryotic organisms, (2, pp. 5-9), the DNA is wrapped around histones forming the chromosomes. These chromosomes (or single chromosome in prokaryotes), are composed by genes that encode functional DNA. Since the discovery of DNA, science has been evolving very fast, supported by the advance on technology, allowing nowadays to be easier and cheaper to sequence the whole-genome of organisms, using technology like Next Generation Sequencing (NGS) (88). With the increase of data available new fields of science, such as Systems Biology (SB) (30), have emerged.

SB is a field of science that aims at understanding the complex systems of living organisms. It is commonly described as a holistic approach that has taken advantage of computational tools and high-throughput experimental data to achieve its goal. Understanding the complexity of living cells can be useful in several areas, such as medicine (to find new biomarkers for diseases, target drugs and new treatments) or industry (increasing the productivity of a compound of interest, such as biofuels and pharmaceutical products) (85). As this field aims at understanding the behavior and the relationships between the elements of an active biological system, instead of an isolated cellular component, it uses several techniques such as Genomics, Transcriptomics, Proteomics, and Metabolomics to study these networks (27; 30).

Genomics is the field of science that studies the organisms' DNA (genome). Unlike genetics that explores exhaustively the function and the composition of a single gene, genomics aims at studying the complete set of genes within a cell, and how they interact leading to the growth and development of the organism(73, p. 47). Gene expression can be summarized in three main steps. During the first step, transcription, the gene's DNA is transcribed to pre-messenger ribonucleic acid (RNA). In the second step, this RNA is processed by splicing (introns are removed, and exons are joined together) and the messenger RNA (mRNA), the transcript, is created. The study of the complete record of these transcripts produced by an organism during its lifetime, under specific conditions or in a unique cell, is called Transcriptomics. Finally, the messenger RNA is translated by ribosomes into a protein. The study of proteins, and the patterns in expression of proteins in a tissue or cell of an organism during its lifetime, is performed with severak techniques available in the field of Proteomics (27; 56). Both fields, Transcriptomics and Proteomics use high-throughput methods in their analysis such as microarray analysis or RNA-seq (Transcriptomics), and 2D gel electrophoresis and Mass Spectrometry (MS) (Proteomics), obtaining valuable information and complementing each other (56). Metabolomics provides the identification and measurements of all metabolites available within a biological system, and the respective concentrations, using combination of different techniques such as Gas Chromatography (GC) and MS. Metabolite profiling has paramount importance for SB, as the availability of metabolites will determine the connectivity between the networks (30).

2.2 MODELLING

The post-genomic era brought new data sources to Science. During the pre-genomic era, biological network reconstructions were performed with information retrieved from literature and biochemical characterization of enzymes (30). Nowadays, various databases containing most of the information required to generate networks at the genome-scale are available, allowing this process to be automated. Examples of such databases are described in Table 1.

Understanding biological networks has become increasingly important as more knowledge and information is being generated. This allows understanding how a biological system responds to changes of environmental conditions. To accomplish this objective, metabolic, transcriptional, and signaling networks can be reconstructed. However, genome-scale models that integrate these networks and others found in biological systems into one comprehensive model, are still under development.

A Transcriptional Network (TN) can be seen as a snapshot of the state of expression of the genome in a cell. As not all genes are expressed inside the cell at the same time, this network provides a blueprint of the genes expression under specific conditions. A

Signaling Network (SN) provides an overview of the proteins that transduce information, with the purpose of changing the transcriptional state of the cell. By receiving these signals, the cell is stimulated to adjust its transcriptional state accordingly to the environmental conditions. A Metabolic Network (MN) model is used to characterize the metabolic status of a cell. It is constituted by a series of biochemical reactions and constraints, and can be modelled at different levels of complexity. The first level, is the cellular, in which only the intakes and outputs are taken into consideration, whilst all other activities within the cell are ignored. The second level is the functional level where the MN is divided by metabolic functionalities, such as anabolism or catabolism. The third level, is related with pathways of an individual functional group. At the basis of the MN, the fourth level, are the biochemical reactions. This is the level in which GSM models are reconstructed (78).

Table 1.: Bioinformatics Databases

Database	Description	Reference
ExPASy	Integrative portal that reaches a wide range of domains of life sciences such as proteomics, genomics, evolutionary biology, systems biology, population genetics, transcriptomics, biophysics, etc.	(6)
MetaCyc	A non-redundant database, containing experimentally verified metabolic pathways. This curated database, contains pathways involved in primary and secondary metabolism and associated compounds, enzymes and genes.	(20)
BioCyc	As a collection of Pathway/Genome Databases (PGDBs), BioCyc provides the predicted metabolic network for a specific organism, including metabolic pathways, enzymes, metabolites and reactions predicted by the Pathway Tools and using MetaCyc as reference database. Additionally, BioCyc provides tools for visualization and analysis of the PGDBs.	(20)
Biochemical, Genetic and Genomic (BiGG)	A database containing high-quality, manually-curated genome-scale metabolic models.	(70)
KEGG	A database containing extensive information about genes, enzymes, metabolites, reactions, and pathways.	(94)
Reactome	Manually curated resource of human pathways and reactions.	(25)
BRENDA	A curated database containing functional and molecular information of enzymes.	(121)
Universal Protein Resource Knowledgebase (UniProtKB)	A database about proteins consisting in two sections: UniProt/Swiss-Prot, a manually annotated source containing information extracted from literature and from computer analysis; UniProt/TrEMBL, containing data automatically retrieved with computational methods, waiting for manual curation.	(23)
SABIO-RK	A database containing comprehensive information about biochemical reactions, their kinetic rate equations and experimental conditions.	(143)
Genomes OnLine Database (GOLD)	A curated resource for comprehensive access to information regarding genome and metagenome sequencing projects.	(93)
NCBI	A collection of several databases that provides tools for analysis, visualization, and retrieval of biomedical, genomic, and other biological information.	(24)

2.2.1 *Genome-Scale Metabolic Models*

For the reconstruction of GSM models, a widely set of Bioinformatics tools that will be later discussed in this document, is currently available. Its development is important in SB, as these models can be used to systematically test and predict manipulations (e.g. genes knockouts), without using expensive and time-consuming wet-lab experiments (85). The reconstruction of GSM models is an iterative and laborious process described by Thiele and Palsson as having at least 96 steps (130), which can be summarized in four main stages (30).

The first stage consists in the genome annotation by retrieving information from different databases. The annotation should contain, ideally, information such as Enzyme Commission (EC) and TC numbers, as well as the associated genes and gene product names, if available. As this information is retrieved from different databases, and the annotation was likely performed after genome sequencing, it can be out-dated. Therefore, it might be necessary to perform a functional re-annotation of the genome, to find possible gaps and discard unnecessary information. Performing a precise and unbiased annotation is critical for the development of high-quality GSM models, as the annotation is usually performed once and assumed as correct (30).

Subsequently (stage two), based on the annotation and available literature, the assembling of the MN is performed, through the identification of biochemical reactions associated with the organism. In this stage, several steps must be followed, the first being, determining the GPR rules. Here, data retrieved from several databases is cross-checked to identify the proteins encoded by the genes, and which are the reactions associated to those proteins. In Figure 1, exceptions to the '1 gene - 1 protein - 1 reaction' rule can be found. Next, the addition of spontaneous reactions is performed, followed by the validation of the reactions' stoichiometry using online databases such as BRaunschweig ENzyme DAtabase (BRENDA) (121), KEGG (94), and MetaCyc (20). The last step of the second stage, is the compartmentation of the identified reactions. This is an important task, as the availability of compounds that have a role in the assigned reactions is not the same for every compartment of the cell. Thus, a reaction will only take place, when the required metabolites are present in the location predicted for such reaction (30; 114).

The third stage is the conversion of the MN to a stoichiometric model. Before the conversion, it is mandatory to include reactions required for the formation of biomass, i.e. a set of reactions that denotes a drain of building blocks (e.g. amino acids and nucleotides) or, in alternative, reactions that lead to the production of macromolecules which constitute the biomass. Subsequently, when all numeric values for the bounds of uptake/excretion reactions fluxes and the non-growth adenosine triphosphate (ATP) requirements are defined, the complete set of reactions can be converted into a stoichiometric matrix. In this matrix, the rows represent the compounds, while columns represent the chemical transformations.

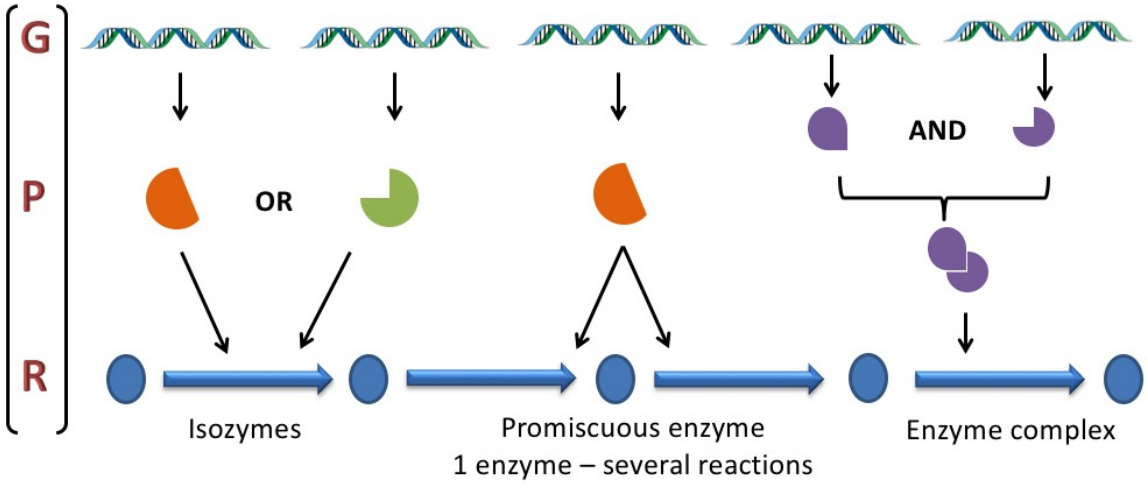


Figure 1.: Representation of GPR associations. Here, some exceptions to the '1 Gene - 1 Protein - 1 Reaction' rule are illustrated: one reaction can be catalyzed by multiple enzymes (isozymes); several reactions can be catalyzed by the same enzyme (promiscuous enzyme); one reaction is catalyzed by one protein that is constituted by subunits encoded by multiple genes (enzyme complex). Figure adapted from Machado et al. (86).

Whereas, the entries of the matrix correspond to stoichiometric coefficients. The mathematical model can then be saved in the standard SBML (61) format, to allow further *in silico* simulations and respective validation.

Finally, at the last stage, the mathematical representation can be used to compare the organism's behavior prediction with experimental data (31). This allows to increase the predictive capabilities and accuracy of the model. To measure the *in vivo* fluxes of metabolites within a cell, wet laboratory processes such as MS or Nuclear Magnetic Resonance (NMR) spectroscopy can be used to analyze substrates labeled with stable isotope tracers, such as ^{13}C markers (142).

Other approaches, such as the constraint-based Metabolic Flux Analysis (MFA), aim to determine the flux of metabolites, based on the knowledge of the metabolic network, measurements of the fluxes, and in the assumption that these metabolites do not accumulate inside the cells over time (15). A MN containing five metabolites is shown in Figure 2, being A the only external substrate available to the system, with a maximum uptake of $\boldsymbol{u}$. The metabolites C and E are excreted by the system as metabolic products (114);

One problem regarding the MFA approach is that for most GSM models, the number of reactions exceeds the number of compounds. This creates an under-determined system, as there are more variables than equations (15). This problem is exemplified in Figure 2, in which for nine fluxes, there are only five internal metabolites, resulting in four degrees of freedom. To overcome this problem, other techniques, described next, were developed:

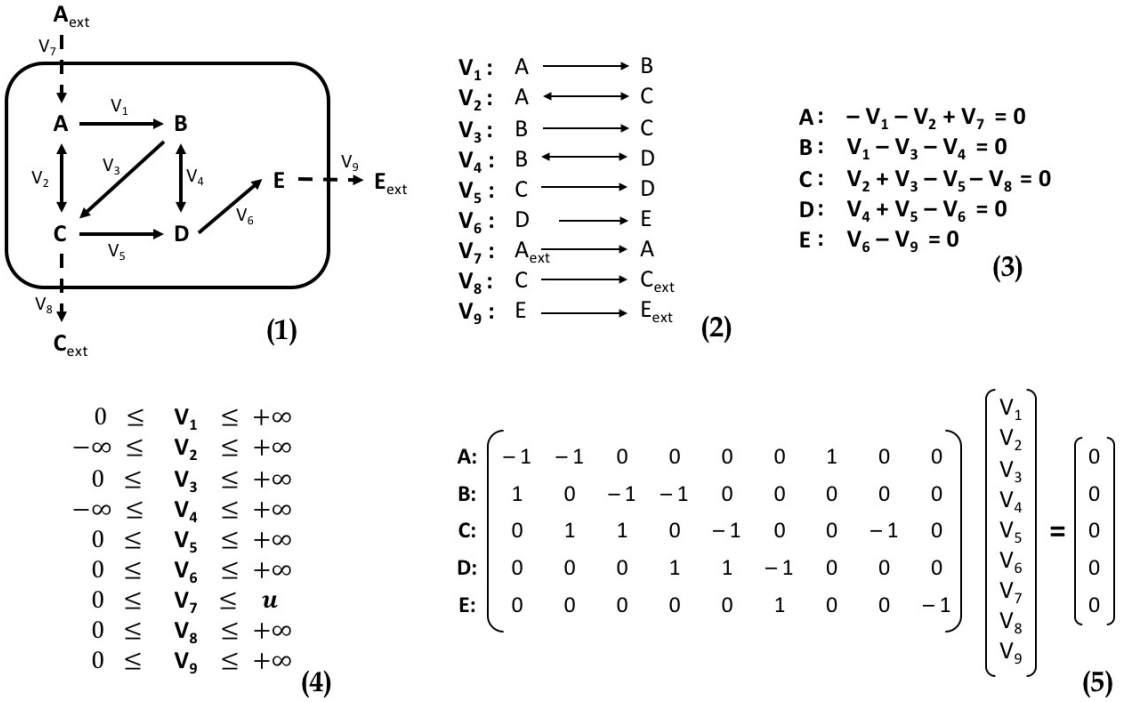


Figure 2.: Example of a MN with **(1)** representing the reactions, the reversibility, and the boundaries of the system. The network is composed by five metabolites (**A** to **E**) and nine metabolites (v_i). The flux v_7 represents the metabolic substrate **A**, and fluxes v_8 and v_9 represent the metabolic products C and E, respectively. **(2)** contains the stoichiometry and reversibility of the network. **(3)** illustrates the steady-state of the network. **(4)** shows the constraints of the fluxes, being $\boldsymbol{u}$ the maximum uptake of substrate **A**. **(5)** represents the mass balance equations in matrix format.

- Flux Balance Analysis (FBA): based on linear programming, FBA is used to determine optimal flux distributions of metabolites in steady-state, using a linear objective function (maximization of growth rate for instance). Thus, it is possible to determine the contribution of each reaction, for the observed phenotype (15; 114). The linear programming formulation for the FBA problem can be specified as stated in 1;

$$\begin{aligned} &maximize \Rightarrow X \\ &\quad subject \Rightarrow S \cdot v = 0 \quad (assumption\ of\ steady-state) \\ &\qquad\qquad \alpha_i \leq v_i \leq \beta_i\ , \quad i = 1, \cdots N \end{aligned} \tag{1}$$

Where $\boldsymbol{X}$ is the linear function to be maximized, $\boldsymbol{S}$ the stoichiometric matrix, v the flux vector, α the lower bound, and β the upper bound.

- Method of Optimization of Metabolic Adjustment (MOMA): based in the same stoichiometric constraints as FBA, MOMA intermediates the difference between wild-type conditions, with minimal perturbation, and the gene knockout for optimal flux (126), by searching for a sub-optimal solution using quadratic programming. A graphical representation of the sub-optimal solution is shown in Figure 3;

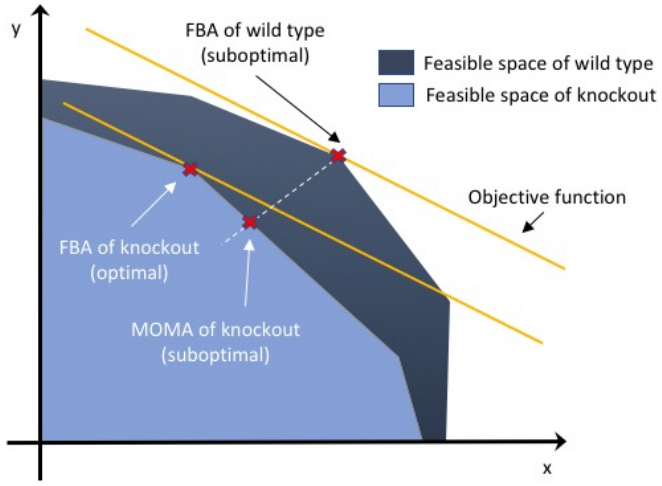


Figure 3.: The MOMA solution is found using quadratic programming. It is the point that minimizes the distance between the optimal solution of FBA of knockout and the FBA of wild type (MOMA of knockout). (Figure adapted from Nanette *et al.* (16)).

- Linear MOMA (LMOMA): a version of MOMA that implements linear programming, instead of quadratic programming (12);
- Regulatory On/Off Minimization (ROOM): like MOMA, searches for a flux distribution that is close to wild-type conditions, by minimizing the number of significant flux changes using Mixed Integer-Linear Programming (127).
- Flux Variability Analysis (FVA): based on linear programming, it calculates the minimum and maximum feasible flux of each reaction, providing the range of variability in which the flux still respects a given set of constraints (87).;

However, it is important to emphasize that the GSM models' reconstruction is an iterative process; hence, the model should be tested and updated with new convenient information, if required (114).

2.2.2 *Reconstruction/simulation and optimization tools*

A wide set of Bioinformatics tools developed to perform computer simulations is currently available. Besides the possibility of performing simulations for phenotype predictions, using the metabolic network analysis approaches aforementioned, some tools also offer to the user the possibility of carrying out the reconstruction process as well. Examples of tools able to perform phenotype prections are described in Table 2.

Table 2.: Tools for reconstruction of GSM models and modelling/analysis.

Tool	Type of Software	Support for Model Reconstruction	Support for MN modelling/analysis	Reference
FASIMU	Standalone	✓	✓	(57)
OptFlux	Standalone	–	✓	(115)
COBRA v2.0	Toolbox	✓	✓	(120)
ModelSEED	WebServer	✓	✓	(54)
FAME	WebServer	✓	✓	(14)
FBA-SimVis	Toolbox	✓	✓	(47)
SBRT	Standalone	–	✓	(144)
FluxExplorer	Standalone	✓	✓	(84)
CellNetAnalyzer	Toolbox	–	✓	(71)
merlin	Standalone	✓	–	(31)
RAVEN	Toolbox	✓	✓	(1)
MEMOSys	WebServer / Standalone	✓	–	(95)
MicrobesFlux	WebServer	✓	✓	(39)
CoReCo	WebServer	✓	–	(103)
SurreyFBA	Standalone	–	✓	(44)
SuBliMinal	Toolbox	✓	–	(129)

As the reconstruction of GSM models is a laborious process that can take from weeks to years, depending on the size of the genome and other factors, Bioinformatics tools that can help speed up the process, while maintaining a high level of confidence, are of paramount importance (31). The Rapid Annotation using Subsystem Technology (RAST) (7), is a fully automated service for annotating bacterial and Archaeal genomes, though not performing the remaining steps of the reconstruction process. Nevertheless, this tool can be used to fill the gap of other tools like the FAME, MEMOSys, ModelSEED, and MicrobesFlux that do not allow user to perform these annotations, and several other steps of the reconstruction process.

Regarding compartments prediction, RAVEN, ModelSEED, and *merlin* assign automatically compartments to the reactions, while FAME and MEMOSys allow the manual assign-

ment of the locations. Although almost all tools provide support for both reconstruction of the GSM models and modelling/analysis, they perform these steps using different approaches. For example, besides *merlin* and ModelSEED, all other tools that perform reconstruction neglect the GPR association, which are, as aforementioned in this document, essential to understand the relationship between genes and reactions at the MN analysis stage.

Despite the standardized methodology for the development of GSM models, important information is still disregarded, and most of the times poorly annotated and consequently excluded from the models, such as the genes encoding transport proteins (51). Transport proteins and the promoted transport reactions, are usually obtained from experimental data and literature. These reactions are often included in models for metabolites that have, at least, one of the following three characteristics:

1. taken in from the medium
2. excreted from the cell
3. transported across intracellular membranes

Since this methodology does not allow to automatically associate genes with transport reactions, it impairs the models' predictions. Therefore, the proper identification of genes encoding transport proteins and the metabolites transported by those carries, is extremely important for the improvement of GSM models' reconstruction (130). Regarding the annotation of transporters, ModelSEED can perform this task, adding transport reactions based on the information from the genome, but it is all targeted for prokaryotic organisms, adding spontaneous reactions to fill in pathways when necessary (54). With this purpose, *merlin* includes a module specifically developed for the identification and semi-automatic annotation of genes encoding transport proteins, the Transport Reactions Annotation and Generation (*TRIAGE*) (29). Besides *TRIAGE* (*merlin*), none of the aforementioned tools is capable of simultaneously identify, classify and annotate membrane transporters, while generating reactions for each of these proteins, both for prokaryotic and eukaryotic organisms.

Regarding the tools that support MN modelling/analysis, all provide phenotype simulation with at least one method (FBA). OptFlux and FASIMU provide also other methods such as MOMA, ROOM, and MFA. Another functionality of the tools is the strain optimization. Among the tools mentioned in Table 2, only SBRT, OptFlux, and COBRA v2.0 implement OptKnock (18) and OptGene (99). OptKnock is a bi-level programming framework that identifies the best set of genes to knockout in order to achieve a desired result, while OptGene (an extension of OptKnock) uses genetic algorithms that incorporates evolutionary information, to identify the best genetic modifications, that allow achieving the desired phenotype. MN analysis using FVA is supported by all tools, except for MicrobesFlux and

RAVEN. OptFlux and CellNetAnalyser also provide elementary flux modes and topological network analysis.

All tools described in Table 2 support SBML format. Another important feature is providing to the user a graphical interface, as most of the times, the user is not familiarized with command line based programs or similar approaches. Optflux, *merlin*, SBRT, and FluxExplorer offer a user-friendly environment. Notice that Optflux was the first software to offer a friendly interface for strain optimization.

2.3 APPLICATIONS

After reconstruction, metabolic models can be used in a myriad of applications from industry to health. These models allow:

- predicting the phenotypical behavior of an organism, under different environmental and genetic perturbations;
- performing the analysis of the robustness of the network, when changing the flux levels of essential gene products;
- and performing *in silico* metabolic engineering (30).

Regarding health applications, GSM models' simulations may lead to the identification of optimal drug targets to attack pathogenic organisms. For example, identifying the organism's essential genes allows performing gene knockouts that may lead the infectious organism to death. If the drug developed to knockout the essential genes of the pathogenic organism has no harmful side effects in the host, the drug can be further developed (30).

The use of microorganisms as cell factories for producing of drugs, biofuels and chemical compounds of interest is currently exploited by Metabolic Engineering (ME), which aims at modifying the metabolism of an organism, through genetic manipulations, to obtain an improved strains. Initially, these modifications were performed randomly until the desired result was achieved. These modifications became more localized, targeting a specific known function of a network. Nowadays, the use of genome-scale models, allows performing systems-wide manipulations, through the computation of genetic modifications with global results (96).

2.4 TRANSMEMBRANE PROTEINS

The plasma membrane is a semi-permeable structure in the cell that surrounds its surface and is responsible for keeping all cellular structures inside, while allowing exchanges with the extracellular environment. This membrane contains two layers of lipid molecules that

direct their hydrophobic tails, formed by fatty acids, towards each other with the objective of forming a barrier that does not allow hydrophilic agents to cross it (2, p. 9). However, there are several types of transport to allow important compounds for the cellular metabolism to cross the membrane. Uniporters have the function of moving a single solute between compartments, while symporters and antiporters are related to a subcategory of membrane transport proteins, the co-transporters, and are responsible for moving also a second metabolite in equal or opposite direction, respectively. These transporters are usually associated with Active Transport (2, p. 654) because solutes are moved against the cell's electrochemical gradient. Nevertheless, there are still other types of transport, such as ATP-binding cassette (ABC), phosphotransferase system (PTS), and electron transport chain (ETC). Examples of three types of transporters are illustrated in Figure 4.

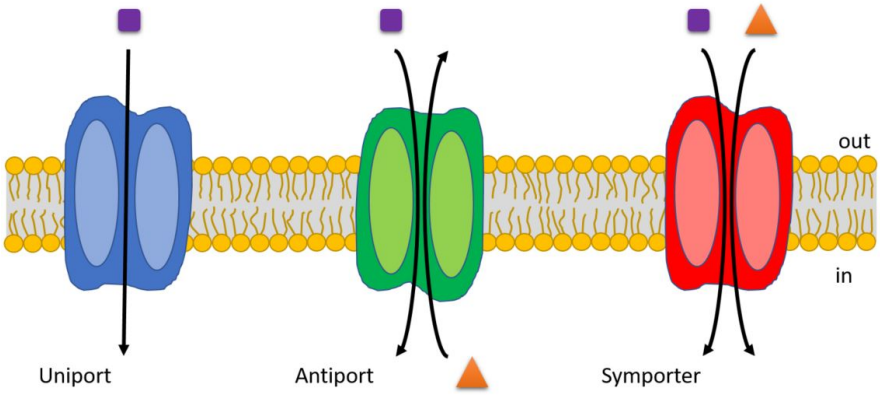


Figure 4.: Example of uniport, symport and antiport processes.

These transports are mediated by Transmembrane (TM) proteins. These are a type of integral membrane proteins, also called intrinsic proteins, that can be classified into two different classes according to their secondary structure: α-helices and β-sheets (72).

Most of the intrinsic proteins can interact with the phospholipid bi-layer of the membrane, because of the presence of hydrophobic side chains in the residues of the protein. This property allows the protein to anchor on the cell's membrane, allowing most TM proteins to cross the entire phospholipid bi-layer. Transmembrane α-helical (TMA) proteins consist mainly of hydrophobic residues and are mostly found in the cytoplasmic (or inner) membranes of prokaryotic and eukaryotic organisms, performing a variety of biologically important functions, such as the transport of compounds across the membrane and signal transduction (133). An example of an α-helix is shown is figure 5.

On the other hand, Transmembrane β-barrel (TMB) proteins are located in outer membranes of bacteria, mitochondria and chloroplasts (124; 65). These proteins perform diverse functions such as bacterial adhesion, structural integrity of the cell wall and material transport, which allows organizing these proteins in subfamilies like porins (123), preprotein translocases (108; 122), and lipocalins (40). Porins transport ions and small molecules

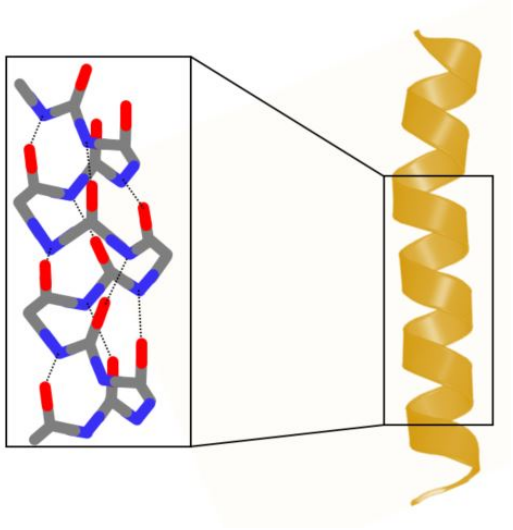


Figure 5.: Example of the structure of an α-helix.

that are not able to diffuse through the membrane. Preprotein translocases are found in endosymbiont derived organelles, such as mitochondrion and chloroplast. This complex allows the movement of proteins through the membrane. Lipocalins, have a variety of functions such as transport of small hydrophobic molecules, formation of macromolecules, and regulation of the immune response. Examples of TMBs are represented in Figure 6.

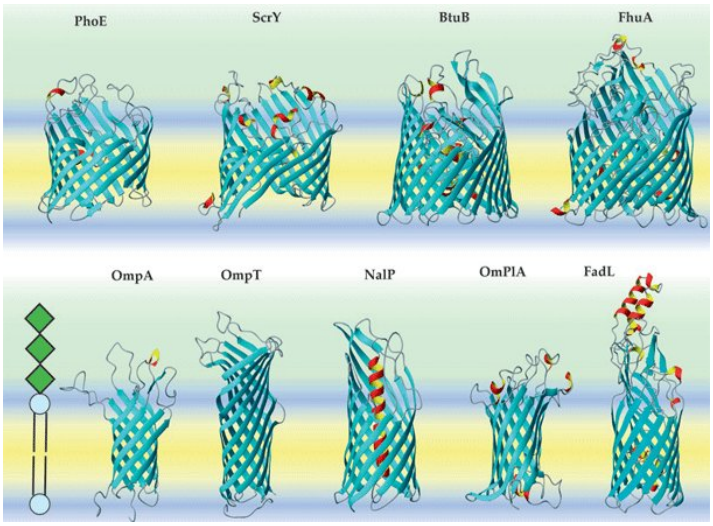
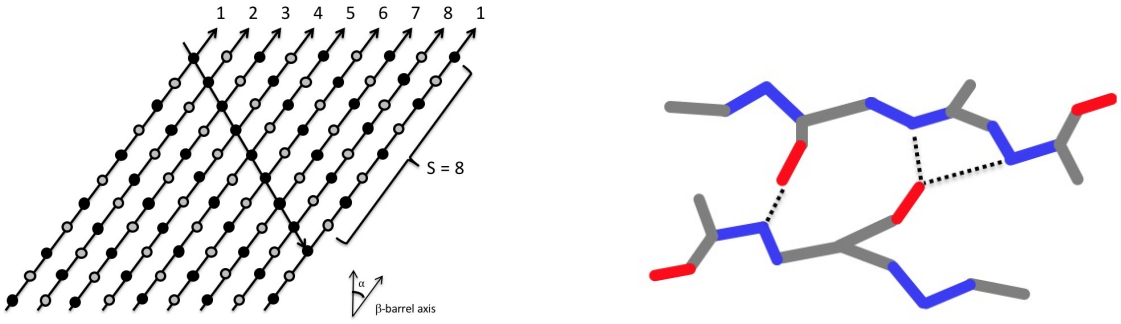


Figure 6.: Crystal structures of β-barrel membrane proteins of the outer membrane of bacteria. Image from Kleinschmidt, 2005 (72).

These proteins' structure consists in a β-sheet, arranged in anti-parallel β-strands connected by backbone hydrogen bonds, that coils and twists around itself, bounding the first β-strand to the last one, creating a structure that resembles a barrel and is characterized by

the number of anti-parallel β-strands and by the shear number. The shear number, defined by McLachlan et al. in 1979, can be described as the displacement between the β-strains, as shown in Figure 7a (89). In 1998 Liu refined this definition, as according to him, a shear number is not unique for a given β-barrel if it contains one or more uneven β-bulges (82). A β-bulge, as shown in Figure 7b, is a region of irregularity in a β-sheet involving two consecutive hydrogen bonds that include two residues, where the β-sheet is disrupted by the presence of an extra residue (111). The strands of a β-barrel are also tilted from 36° to 44° relative to the barrel axis.



(a) (b)

Figure 7.: **a)** Method to determine the shear number; **b)** Example of a beta-bulge.

According to G. Schulz in *β-Barrel membrane proteins*, all known bacterial membranes' β-barrels followed the same construction principle (124). Thus, the existence of β-strands and β-sheets in a protein, does not indicate that the protein is a β-barrel. Also, according to the author, the prediction of transmembrane β-barrels from a sequence, should be an easy task for a program that takes these rules into consideration. A wide range of computational methods for prediction of transmembrane α-helical proteins, with high levels of accuracy and precision is already available. In recent years, the development of methods for transmembrane β-barrels predictions has grown, with the introduction of algorithms, web servers and databases to improve the quality of the predictions.

2.4.1 *TMA proteins predictors*

The TMA proteins are encoded by 20% to 30% of the genes in an organism and can be found in all cellular membranes (76; 36). Some publicly available databases contain information about this type of TM proteins such as TCDB (118), PDBTM (74), TOPDB (136), TOPDOM (135), MPtopo (63), MPDB (106), TMPDB (62), and TMFunction (50). When dealing with topology predictions, one of the largest problems is that due their high hydrophobicity, signal peptides are often mistakenly predicted as TM proteins. Signal peptides are sequences

with a length between 16 and 30 amino acids, responsible for directing proteins to the correct compartment (2, p. 701). To overcome this problem, some predictions tools make signal peptides prediction along with the topology prediction. Both types of predictors are described in Tables 3 and 4.

Table 3.: Tools for prediction of transmembrane α-helices.

Tool	Method	Year	System
TMHMM (75)	Hidden Markov Model	2001	Web Server / standalone soft.
HMMTOP 2.0 (137)	Hidden Markov Model	2001	Web Server
MEMSAT3 (64)	Neural Networks	2007	Web Server
OCTOPUS (139)	Hidden Markov Model and Artificial Neural Networks	2008	Web Server / standalone soft.
SCAMPI2 (102)	Search for hydrophobicity in N and C terminals.	2015	Web Server

Table 4.: Tools for prediction of TMAs and signal peptides.

Tool	Method	Year	System
PolyPhobius (66)	Hidden Markov Models	2005	Web Server / standalone soft.
Phobius (67)	Hidden Markov Models	2007	Web Server / standalone soft.
SPOCTOPUS (138)	Hidden Markov Models	2008	Web Server / standalone soft.
Philius (110)	Bayesian Networks	2008	Web Server / standalone soft.
TOPCONS2 (134)	Consensus	2015	Web Server / standalone soft.
CCTOP (33)	Consensus	2015	Web Server / standalone soft.

2.4.2 *TMB proteins predictors*

Part of the aforementioned databases for TMA proteins (TCDB, PDBTM, TOPDB, MPtopo, MPDB, TMPDB, and TMFunction) also include information about certain families of β-barrel proteins. Others, like PSORTdb (145), TMBETA-GENOME (49), TMBB-DB (42), Orientations of Proteins in Membranes (OPM) (83), and OMPdb (131) only contain data of TMB proteins. Regarding topology predictions, a brief summary of some tools available for predicting TMB is shown in Table 5.

Table 5.: Some of the available methods for prediction of transmembrane β-barrels.

Tool	Method	Year	System
BOMP (13)	Motifs finding	2003	Web Server
MCMBB (8)	Markov Chain Model	2004	Web Server
ConBBPRED (9)	Consensus	2005	Web Server
TMBETA-NET (48)	Neural-network	2005	Inactive
TMB-Hunt (43)	k-nearest neighbor	2005	Inactive
partiFold-TMB (140)	Boltzmann partition function	2007	Web Server / standalone soft.
TMBpro (107)	Neural-network	2007	Web Server
TMB-KNN (59)	k-nearest neighbor	2008	Web Server
BetAware (119)	Machine learning methods	2013	Web Server / standalone soft.
BOCTOPUS2 (53)	SVM and HMM	2015	Web Server / standalone soft.
PredbβTM (22)	Support Vector Machine	2015	Web Server
PRED-TMBB2 (132)	Hidden Markov Model	2016	Web Server

A study assessing the performance of TMA predictors is already available in Tsirigos et al.: *Topology of membrane proteinspredictions, limitations and variations* (133). Phobius, currently used by *TRIAGE*, is described in the study as prepared to tackle one of the major problems in of topology predictions: signal peptides that are erroneously predicted as transmembrane segments. Thus, as this tool is characterized as a good predictor, no further assessments regarding TMAs are made in this work. However, one of the objectives is the assessment of tools for predicting transmembrane β-barrels. Therefore, a study similar to the aforementioned will be performed, regarding the following tools:

BOMP

The β-barrel Outer Membrane Protein (BOMP) predictor is a program, based on two modules, developed to recognize β-barrel proteins encoded within genomes of gram-negative bacteria. The first module searches for C-terminal patterns typical of many TMB proteins, while the second calculates an integral β-barrel score of the sequence, based on the similarity to amino acids typical of transmembrane β-strands. BOMP apparently has no limit to the number of sequences allowed for the submission, and also allows using Basic Local Alignment Search Tool (BLAST) to retrieve additional information. The output is a table containing the sequences with predicted β-barrels, classified with scores ranging from 0 to 5 (levels of reliability).

MCMBB

Markov Chain Model for Beta Barrels (MCMBB) uses, as the name implies, a Markov Chain model which captures alteration of hydrophilic-hydrophobic residues in the transmembrane β-strands of outer membrane proteins, while providing a low number of false positives. The input is limited to 1000 sequences and the output is the probability of existence of a β-barrel, for each sequence. Positive values denote a higher likelihood of a given sequence being a outer membrane protein with β-barrels.

partiFold-TMB

partiFold-TMB uses a set of algorithms for computing the Boltzmann partition function (32), estimating β-strands residue interaction probabilities and performing individual β-contact prediction. This tool accepts FASTA files as input and has a wide sort of customizable configurations for the output files.

TMBpro

TMBpro is a method based in neural networks, which predicts secondary structures, β-contacts and tertiary structures of TMB proteins. TMBpro accepts only one sequence for each submission and the required length of each sequence is between 100 and 800 amino acids. The results are sent by e-mail.

TMB-KNN

TMB-KNN uses a k-nearest neighbor strategy, discriminating TMB and non-TMB proteins based on the Weighted Euclidean distance. This method is gradually improved by including homologous sequences and searching for a set of residues and di-peptides for calculating the distance.

BetAware

BetAware is a machine-learning tool developed for detecting and predicting the topology of outer-membrane β-barrels in prokaryotes. For detection, this software is based in N-to-1 network encoding (91) and Extreme Machine Learning (60) training algorithms , while for topology prediction a Grammatical Restrained Hidden Conditional Random Field (GRHCRF) (37) model is applied. BetAware only accepts one sequence for each submission and the next submission is only accepted when the first submission ends. This tool returns the number of β-strands for the given sequence and their exact location, when TMBs are predicted.

*BOCTOPUS*2

BOCTOPUS2 is an improvement of the previous prediction tool BOCTOPUS (52). Unlike its predecessor, this tool can now predict lipid/pore-facing orientation of residues in the transmembrane β-strands, using this information to improve topology predictions. This tool also exploits the dyad-repeat feature of bacterial TMBs to identify transmembrane β-strands. This software uses the Viterbi algorithm (41), Hidden Markov Model and Support Vector Machines (SVM) to perform predictions. BOCTOPUS2 only accepts 5 sequences for each submission and returns the number of β-strands for each TMB predicted and a detailed chart with all predicted topologies and SVM scores.

*PRED-TMBB*2

PRED-TMBB2 (132) is an updated version of the PRED-TMBB (10). PRED-TMBB2 is able to predict the topology of β-barrel outer membrane proteins, while simultaneously differentiating them from water-soluble proteins. This method is based on Hidden Markov Models, trained according to the Conditional Maximum Likelihood criterion and uses a database of β-barrel outer membrane proteins (OMPdb) (131) as reference to reduce running time. It also incorporates evolutionary information for better results. PRED-TMBB2 accepts up to 5 000 sequences as input, offers the user the option of running signal-peptide predictions and to use a library of characteristic profile Hidden Markov Models (pHMMs) for β-barrel regions derived from OMPdb, before performing predictions. As output, a table containing the signal-peptide location, β-barrel, OMPdb family, number of β-strands, reliability and some charts illustrating the locations of each β-stand, is presented.

2.5 *merlin*

merlin is a user-friendly Java™ application, built on top of the AIBench software development framework (45), which performs the reconstruction of genome-scale metabolic models for every organism that has its genome sequenced (31). It performs several steps of the reconstruction process, including the functional genome annotation, using homology similarity search tools such as BLAST (3) and the profile HMMER (34). Functional information is retrieved for all homologous genes and the annotations are automatically scored, allowing the user to change the automatic selection, dynamically (re-)annotating the genome (31).

merlin includes a module specifically developed for the identification and semi-automatic annotation of genes encoding transport proteins, and also for generating reactions for those carriers (*TRIAGE* (29)). Furthermore, *merlin* allows integrating *TRIAGE*'s data into existing and draft metabolic models.

2.6 METHODS FOR AUTOMATIC ANNOTATION OF TRANSPORTER SYSTEMS

Despite the exponential growth of sequenced genomes in recent years, the number of reliable tools available for automatic annotation of transporter systems to develop high-quality reconstructions is still low (51). Some databases that provide information about these transport systems, are available in table 6.

Table 6.: Some of the available transport databases.

Database	Description	Reference
TCDB	A database containing structural, functional, mechanistic, evolutionary and disease/medical information about transport systems, creator of the Transport Classification (TC) system, the only system adopted by the International Union of Biochemistry and Molecular Biology (IUBMB) for the organization of transport proteins. A free access database, currently with 17 399 entries and constantly increasing this value, with cross-references to several other databases, including UniProt. Is has also available tools like BLAST to perform a rapid identification of transporters using homology.	(117)
TranportDB 2.0	TranportDB 2.0 is a completely updated version of the TransportDB database (109) containing the transporters regarding a wide set of organisms, and also capable of identify transporters in genomes submitted by the users. This platform is also capable of performing comparisons between a set of genomes. The transporters comprised in this database are organized accordingly to the TC system. Currently, there are available 2 760 organisms in this database, being 164 of archaea and 39 of eukaryota.	(35)
ARAMEMNON	A database of 9 plant species. Besides information regarding membrane proteins, this database also has information of non-membrane proteins. It provides BLAST option for homology search.	(125)
YTPdb	A wiki database containing the information of 298 transporters regarding yeast Saccharomyces cerevisiae. Although organized according to the TC system, it does not provide TC numbers, and by being a wiki website, it all pages are editable by the user.	(17)
ABCdb	A database devoted to the ABC transporters for prokaryotic genomes, organized according to the TC system.	(105)

Lee et al. (79) developed a method for inferring and constructing transport reactions associated with transporter proteins, based on the organism's genome annotation. The Transport Inference Parser (TIP) analyses the name of each protein, inferring the transport reaction promoted by these proteins, as shown in Table 7.

Table 7.: Simple example adapted from (79) representing TIP's input and the final result for each protein. Note: '[ext]' stands for 'extracellular'

Input: protein function	**Output: Inferred transport reaction**
predicted ATP transporter of cyanate	$cianate[ext] + H_2O + ATP = cyanate + phosphate + ADP$
putative phosphate ABC transporter, ATP binding subunit	$phosphate[ext] + H_2O + ATP = 2phosphate + ADP$
putative potassium channel	$K^+[ext] = K^+$
sodium / proline symporter	$Na^+[ext] + L - proline[ext] = Na^+ + L - proline$
lactose transport system permease protein	$H^+[ext] + lactose[ext] = H^+ + lactose$

Other toolboxes, like Reconstruction, Analysis and Visualization of Metabolic Networks (RAVEN) (1) or SuBliMinaL (129), generate transport reactions based on information retrieved from databases. These reactions, both for internal and external transporters, are also integrated into metabolic models. RAVEN automatically retrieves information from KEGG and BRENDA databases regarding existing models that are closely related to the studied organism, based in the assumption that related organisms share metabolic capabilities. Nevertheless, it also allows the user to add transport reactions manually and fill gaps using KEGG Orthology (KO) IDs (68) to ensure a functional network. SuBliMinaL optionally provides to the user a default set of transporters, retrieved from BiGG, not relying on genomic information.

Regarding the annotation of transporters, ModelSEED and Pathway Tools (69) can perform this task using RAST functional annotations to develop models, adding external transport information based on the information from the genome, adding spontaneous reactions to fill in pathways when necessary, but just for prokaryotic organisms.

A new webserver was released in 2017, the Transporters via Annotation Transfer by Homology (TransATH) (4), which claims to be able of automatically annotating transporter systems, returning information like TC family, transported substrates, subcellular location, and prediction of transmembrane segments. The authors claim that this software makes use of Milton Saier's (Saier's lab - TCDB) protocol, automating it to annotate transporters. Their method performs BLAST against the entire set of records present at TCDB, and infer TC families through homology. Aplop and Butler manually pre-processed the substrates available in TCDB and the groups to which these substrates belong, to perform the identification of which substrates should be carried by each transport system. Furthermore,

the team plans to extract the manual annotations present in *TRIAGE*'s curated database to improve their annotations. As the tool is still in beta version, the prediction of subcellular location is not yet implemented but will be carried out in the future by two different external webservers: TM-Coffee (21) and LocTree3 (46). The prediction of transmembrane segments is also performed by an external webserver: HMMTOP. When all calculations and predictions are complete, TransATH displays the results in table format with an option for downloading a .csv file, containing all aforementioned information. According to the authors, TransATH takes up to 100 minutes to annotate a "typical fungal genome", and when the job is complete, notifies the user via e-mail, providing a web link to the results, however this feature is not yet available.

Nevertheless, besides *TRIAGE*, none of the aforementioned tools can identify membrane transport proteins, annotating these with reactions.

2.7 TRANSPORT REACTIONS ANNOTATION AND GENERATION - TRIAGE

TRIAGE is a Bioinformatics tool available in *merlin* (31) that identifies and classifies transporter proteins, based on the identification and classification of genes encoding transmembrane proteins. *TRIAGE* is entirely developed in Java™ and uses H2 Database Engine (H2) (92) or MySQL™ databases to store any retrieved or generated information. Although TCDB is the main source, data is retrieved from five different databases. UniProtJAPI (98) is used to retrieve phylogenetic data regarding TCDB's transport systems entries, whereas KEGG, ChEBI (28), and semantics SBML 2.0 (80) are used to gather additional information for metabolite identification and characterization.

An internal database to help performing the TCG identification was also created and filled with data retrieved from TCDB with the following fields (when available):

- UniProt accession number;
- protein name;
- organism;
- taxonomy;
- TC number;
- TC family;
- transported metabolite;
- direction;
- reversibility;
- reacting metabolites;
- and equation.

The field TC number (116) refers to a system analogous to the Enzyme Commission (EC) system (11) used in enzymes classification. The TC system, unlike the EC system incorporates both functional and phylogenetic information, and bases its classification in five parameters, corresponding each one to the five numbers or letters, separated by a dot, present in the TC number. Thus, using the example provided by TCDB in its website, the five parameters can be described as follows ***V.W.X.Y.Z***:

- ***V*** - (a number) the transporter class;
- ***W*** - (a letter) the transporter subclass;

• ***X*** - (a number) the transporter family (sometimes actually a superfamily);
• ***Y*** - (a number) the subfamily in which the transporter is found;
• ***Z*** - (a number) corresponds to a specific transporter with a particular substrate or range of substrates transported.

The class parameter (**V** in the example), is composed by 7 categories, namely:

- 1.*.*.*.* **- Channels/Pores:** This category consists in α-helical or β-strand-type spanners that may be specific for a particular molecular species or molecules. This transport system catalyzes the facilitated diffusion;
- **2.*.*.*.* - Electrochemical Potential-driven Transporters:** Transporter systems included in this category use uniport, antiport, or symport systems to drive species between compartments against the concentration gradient;
- 3.*.*.*.* **- Primary Active Transporters:** Transport against a concentration gradient with consumption of a primary source of energy (chemical, electrical, and solar);
- 4.*.*.*.* **- Group Translocators:** In this type of transportation an organic molecule is transported across the membrane while being chemically modified.
- 5.*.*.*.* **- Transmembrane Electron Carriers:** Systems responsible for catalyzing the electron flow across a biological membrane.
- 8.*.*.*.* **- Accessory Factors Involved in Transport:** This category is constituted by transporter proteins that are complexed to known transport proteins. Auxiliary transport proteins that do not actually participate in the transport process represent other example, such as regulatory or energy coupling proteins.
- 9.*.*.*.* **- Incompletely Characterized Transport Systems:** Transport proteins with unknown classification and putative transport proteins are assigned to this category. Proteins from this class can be later moved to one of the aforementioned classes after appropriate classification.

TRIAGE's first assumption when performing the assignment of transporter systems to genes is that transporter proteins are in membranes. Hence, the initial step is to remove genes without Transmembrane Domains (TMD), to identify the TCGs (29). Only genes with at least n TMD are classified as TCGs. This step, performed using either Phobius or TMHMM, is crucial as a gene not predicted to have any transmembrane α-helices is automatically excluded. TMHMM and Phobius are tools that use the amino acids' sequence to predict the number of α-helices. As described in Tables 3 and 4, both tools are available through web services and standalone packages, thus users can run the tools locally.

However, the rationale for identifying TCGs should include transmembrane β-barrels. Currently, *TRIAGE* is not using any tool to determine the presence TMBs. Thus, *TRIAGE*'s algorithm migth be missing transport systems associated with TMBs. These transporters act as mediators in the process of moving metabolites from one compartment to another (2, p. 656).

After identifying TCGs present in the genome's sequences, an alignment against TCDB, aiming at finding similarities with known transporter systems, is performed. Biojava (104) was used to implement the Smith-Waterman algorithm (128), performing local alignments to guarantee optimality and high sensitivity when sequences are homologous. A similarity threshold based on the maximum score of the alignments can be defined by the user (10% as default). However, as TCDB is a small database, when a minimum of 5 α-helices is identified in the sequence, the threshold is lowered, justified by the strong evidence of a transporter role. For every extra α-helix, the threshold is reduced 0.5% until it reaches half of the original threshold value.

After completing the alignments and retrieving the information for the most similar genes in TCDB, a score that determines which metabolites m will be assigned to each gene g is calculated. This is done by weighting the frequency of each metabolite m within the homologous protein records. Moreover, as shown in equation 2 it is assumed that related organisms will thrive in similar environments rather than dissimilar organisms. The balance between the frequency score ($Score_A$) and the taxonomy score ($Score_B$) is provided by the parameter α, which takes values between 0 and 1.

$$Score = \alpha \times Score_A + (1 - \alpha) \times Score_B \tag{2}$$

The sum of the similarities of each homologous TCDB record that transports metabolite m is divided by the sum of the similarities of all homologous proteins, to calculate the frequency score (equation 3). Thus, S_i refers to the similarity to the i^{th} TCDB record, H is the total number of hits, and Vm_i is a binary variable described in equation 4.

$$Score_A = \frac{\sum_{i=1}^{H} S_i \times Vm_i}{\sum_{i=1}^{H} S_i} \tag{3}$$

$$Vm_i = \begin{cases} 1, & if\ metabolite\ m\ is\ in\ record\ i \\ 0, & otherwise \end{cases} \tag{4}$$

The taxonomy score (equation 5) is calculated by dividing the taxonomy frequency by the maximum taxonomy M_T multiplied by the frequency of the genes that transport the metabolite, though the score may be subject of a potential penalty. This is used to penalize the score for metabolites that are associated to a low number of similar proteins, which might result from an incorrect assignment, in which β is a penalty parameter with a value

between 0 and 1, set by default as 0.05, and p_m is the metabolite penalty described in equation 6. t_i is the number of common taxa between the organism to which record i belongs and the target organism.

$$Score_B = \frac{\sum_{i=1}^{H} t_i \times Vm_i \times (1 - p_m \times \beta)}{M_T \times \sum_{i=1}^{H} Vm_i} \tag{5}$$

$$p_m = \begin{cases} 0, & if \ \sum_{i=1}^{H} Vm_i \geq Min_{Hits} \\ Min_{Hits} - \sum_{i=1}^{H} Vm_i, & otherwise \end{cases} \tag{6}$$

The algorithm used to classify the metabolites is also used to classify how a metabolite is transported. The same metabolite can be transported by carriers such as uniport, symport or antiport, and the carrier chosen to the metabolite is based on which type has the highest score. Nevertheless, if two types have the same score, both types are selected for that metabolite.

TRIAGE also assigns sub-cellular locations to the identified transporters. This task involves using the tools WoLF PSORT (58) or PSORTb 3.0 (146). Whereas WoLF PSORT is able to predict compartments only for eukariotic organisms, PSORTb 3.0 only does it for Bacteria and Archaea. As none of these tools provide a web Application Programming Interface (API) available to retrieve results, this section is implemented in a way that the users submit their requests in the tool's respective website, and when the results are ready, users provide the file's results to *merlin*. Then, the information is parsed, processed and stored in the database. When generating the transport reactions, the secondary compartments (if any) can be taken into account whenever their scores differ less than a user-defined percentage, regarding the primary compartment. In addition, the user can also set which compartments should be ignored, by writing in *merlin* the names of the compartments to be ignored separated by ';'.

After the metabolites and transport type selection, and the identification of the compartments, the transport reactions can be generated. The whole process of TCGs identification is briefly described in Figure 8.

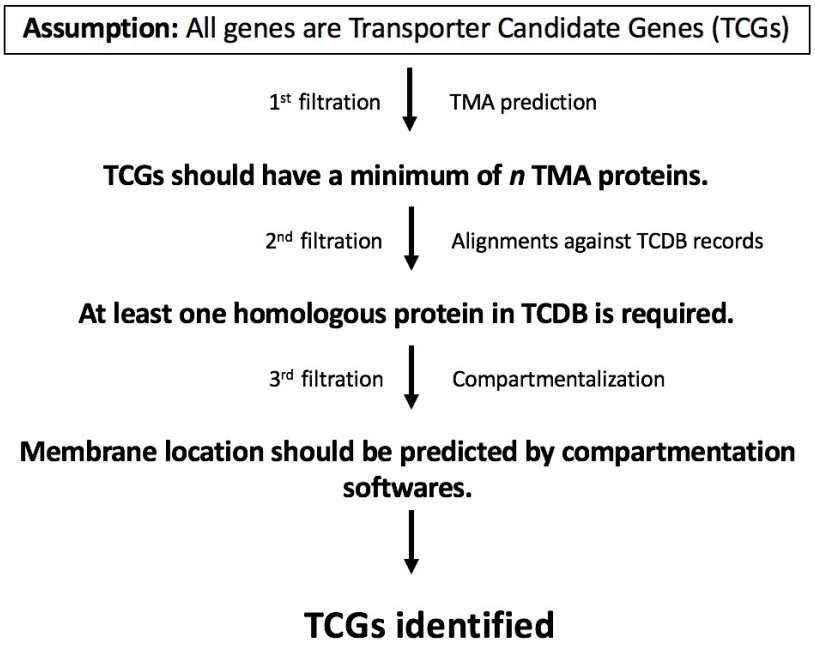


Figure 8.: Process of TCGs identification.

2.8 KBASE

The Department of Energy Systems Biology Knowledgebase (KBase) is an online platform, developed to solve the biggest problems of systems biology, prediction and design of biological functions for small and large-scale genomes (5). KBase increases the efficiency of large-scale analyses by providing to the user an open source of information and tools. Thus, users do not need to access data from different sources or learn several systems to develop and run systems biology workflows. KBase allows creating narratives, in a user-friendly graphic interface, in which users can download or upload data from other sources. Users can employ tools from KBase's wide catalog, divided in 10 categories, to perform the analysis of such data:

- Read Processing
- Genome Assembly
- Genome Annotation
- Sequence Analysis
- Comparative Genomics
- Metabolic Modeling
- Expression
- Microbial Communities
- Utilities
- Uncategorized

Ultimately, KBase allows the user to share and publish their narratives and conclusions. KBase is available at https://kbase.us/.

2.9 NEO4J

Neo4j https://neo4j.com/ is an open-source graph database with breakthrough performance in the analysis of large amounts of data. Unlike relational databases, graph databases allow maintaining performance, independent of the size of the database. The performance of relational databases"join" queries, tends to decrease with the increase of the database size. In graph databases, the performance is not affected by the size of the overall database, as the execution time is only proportional to the size of the database that satisfies the query (113, p. 8). Relational databases lack relationships, as these are designed to encode tabular structures, and are not prepared to process connections found in real world data (113, p. 11). Thus, graph databases allow finding patterns and clusters of the data that may not be obtainable in other resources. As graph databases can be represented in visual format in a completely different way, these patterns can be found visually.

A graph is a set of vertices (nodes) and edges (relationships), in which several nodes can be connected by different relationships. Edges can connect vertices in any possible way with no restrictions, and an edge must have a start and an end vertex (example in figure 9). This representation allows storing information for all varieties of fields, such as supply-chains, rockets engineering, geoespatial systems, social networking, biological data, etc (113, p. 1).

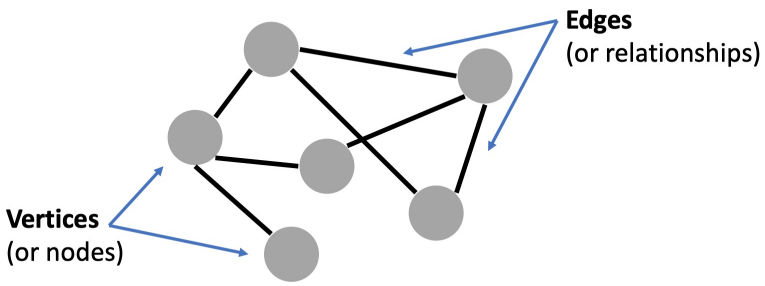


Figure 9.: Graph example.

Next Generation Sequencing and other high-throughput technologies generate vast amounts of data, in a cheap and scalable format, which were not previously accessible. However, the availability of this information also depends on cost-effective ways to store data, and how to efficiently make it public and easily queryable (97). Approaches like Bio4j are currently undergoing to deal with this problem. Bio4j is an open-source framework powered by Neo4j (90). The purpose of this project is to aggregate data available in different sources, into a single one. This intends to solve problems like high performance access to data, integration, and expression of the data (97). Bio4j is available at https://github.com/bio4j/bio4j.

2.10 DOCKER

Docker https://www.docker.com/ is a software that performs virtualization at Operating System (OS) level, allowing multiple isolated virtualized servers to run on a single physical server, also known as containerization. Docker applications are built in "containers" that are related to underlying "Docker images", containing the system binaries and libraries required by the applications to run. On the other hand, Virtual Machines (VMs) build and isolate system resources, as well as entire working environments, running their own OS to run other applications, whereas Docker isolates the applications themselves.

Docker provides an environment that ensures that applications work identically, independently of the host OS installed in the infrastructure. Docker images usually have tens of megabytes, unlike VMs that can take up to tens of gygabytes. Running on top of the host OS, Docker shares with the host the resources of the infrastructure though the "Docker Daemon", which is the layer that manages differences between software and host OS/infrastructures, providing a uniform environment for the applications. On the other hand, in VMs, the hypervisor is the entity responsible for creating and running VMs, allocating Central Process Unit (CPU) and Random Access Memory (RAM) resources from the infrastructure to the virtual environment. This can be seen as running several machines in the infrastructure of a single machine, largely decreasing the available resources to the host OS. Hypervisors can be divided in two different classes: type 1 (or bare-metal hypervisor) and type 2 (or hosted hypervisor). The difference between these classes is that type 1 hypervisors run directly on host's infrastructure, while in type 2 hypervisors, a host OS is required to run on top of it. The differences between Docker and VMs are shown in figure 10.

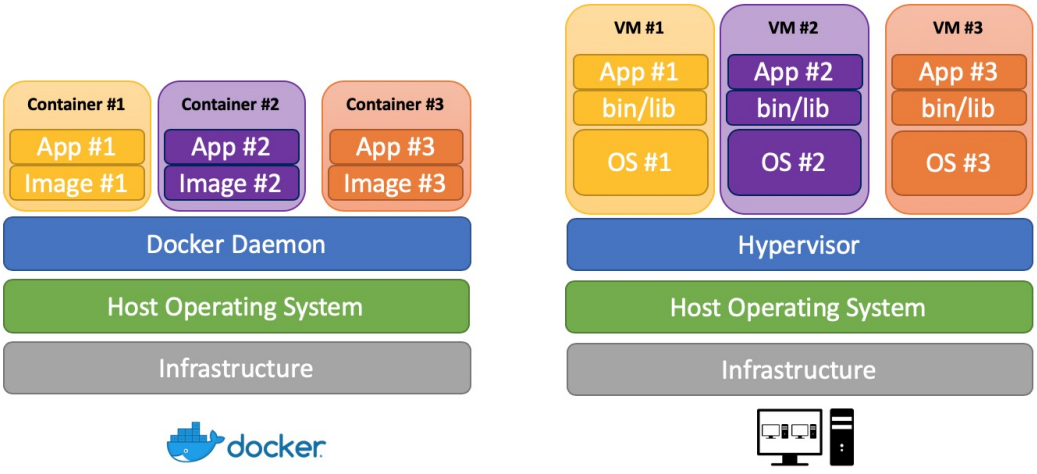


Figure 10.: Comparison between Docker and a Virtual Machine. In case of Type 1 VMs, host OS level does not exist.

2.11 CASE STUDIES

Escherichia coli (*E. coli*) model *i*AF1260 (38) was used for the validation of *TranSyT*'s results. *E. coli* is a well-known gram-negative, rod-shaped, facultative anaerobic Bacteria. It is commonly used as model organism in microbiology studies, and biological engineering, for genetic manipulation and production of sub-products (77; 141). Most of the *E. coli* strains are harmless to public health; however a few serotypes have been associated with food poisoning outbreaks (101; 112). The *i*AF1260 model is highly cited by the scientific community, currently counting 1 310 citations, according to the Google Scholar search engine. This model encompasses:

- 1 260 genes;
- 2 077 reactions;
  - 1 387 metabolic reactions (67%);
  - 690 transport reactions (33%);
- Gene-Protein-Reaction associations;
  - 1 294 metabolic reactions (93%);
  - 625 transport reactions (91%);
- 1 039 metabolites;
- 3 compartments;

3

METHODS

3.1 *TranSyT*'s architecture

After assessing the limitations of the current versions of *merlin* and *TRIAGE*, an outline of the requirements that *TranSyT* should meet was performed. Planning was idealized as follows:

1) Conduct a study to access the best method to find transmembrane β-barrels.

2) Perform a comparison between *TRIAGE*'s internal database and TCDB, validating the new TCDB information regarding the metabolites transported by each system.

3) Design and implement a method to automatically web scrape all information available in TCDB, for each transporter system.

4) Implement a method to assign identifiers to metabolites retrieved in the previous step.

5) Develop a strategy to identify relationships between compounds (hierarchical ontology).

6) Use retrieved information to generate transport reactions for all transporters described in TCDB.

7) Design and implement a suitable database to store all generated information.

8) Develop a strategy to associate genes with TCDB records.

9) Implement a method to perform the reactions' compartmentalization.

With all planning delineated, the guideline foundations of how to implement the software were also listed as follows:

- User's interaction with the software should be only when initiating the software and using controlled input (less interference as possible, to avoid human error);

- Short running time, to allow implementation in KBase services;
- Use only the genome and respective taxonomy identifier as input to generate the output;
- Design an implementation of modular software, allowing the modules to be easily replaceable and easier to update and debug;
- Software should run standalone without requiring external webservices;
- Develop a strategy using the results of the predictions of proteins localization and presence of TMD, to assign confidence levels to *TranSyT*'s output.
- Keep a backup record of all data generated to trace possible errors.

Following the aforementioned criteria, the software should encompass two different components:

- the graphical interface with which the user will interact,
- the service component allocated in a server.

The graphical user interface is the module of *TranSyT* responsible for associating the transport reactions to the genes submitted by the user. Whereas, the gathering of online information, storage, and time/resources consuming tasks will be executed by the service component. The service is responsible for creating/updating *TranSyT*'s internal database and communicates with the graphical interface when required. The service component will also be installed in a Docker component, as this facilitates the integration of the software with both *merlin* and KBase. *TranSyT*'s architecture is illustrated in figure 11. The development of the software was performed using Java™ SE 10.

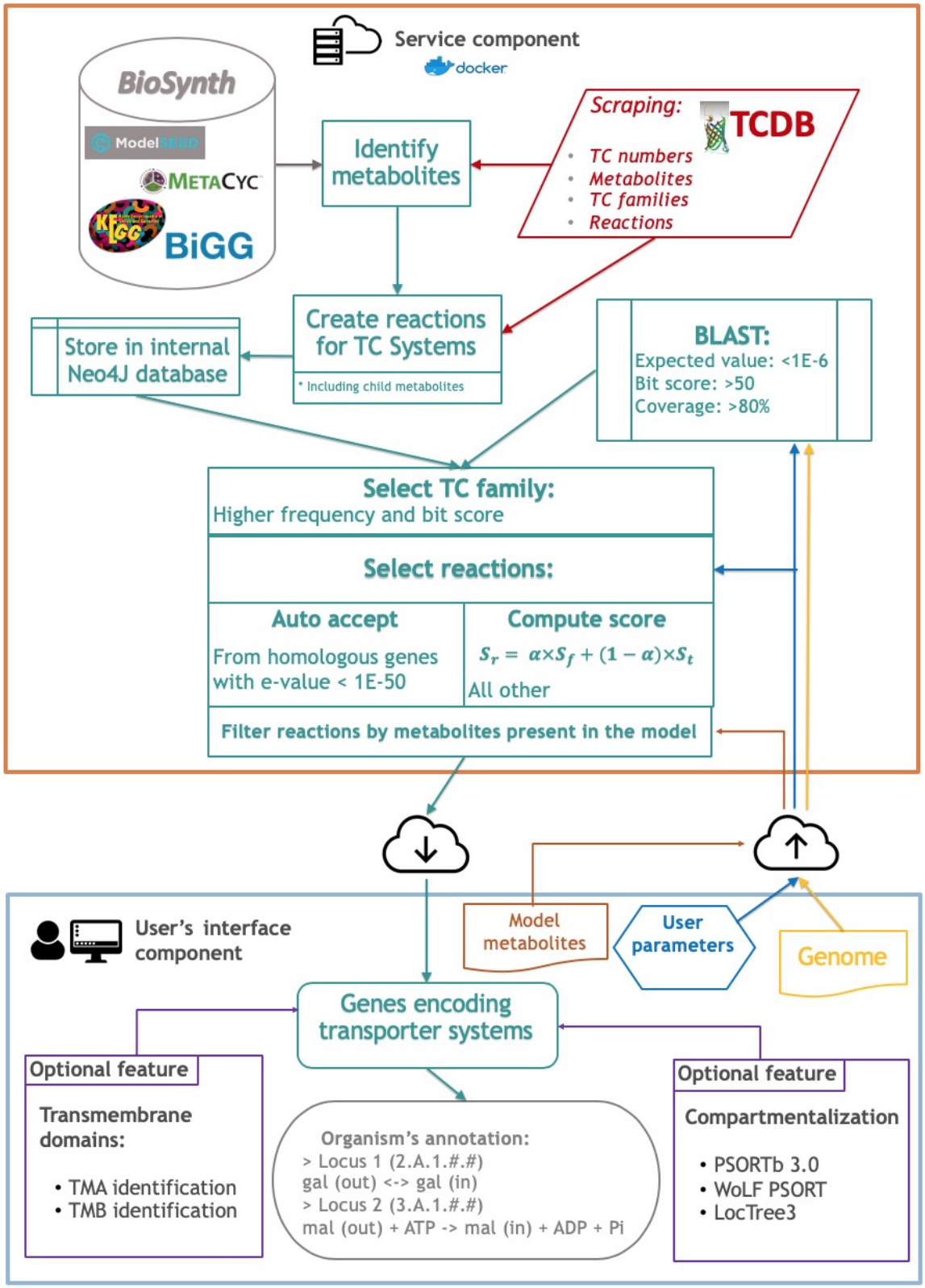


Figure 11.: Architecture of *TranSyT*.

3.2 Assessment of available methods to predict transmembrane β-barrels

Currently, *TRIAGE* does not include in its algorithm tools, nor report parsers, to include transmembrane β-barrels prediction results. Hence, one of the objectives of this work, is to improve the process of TMD identification, therefore refining the annotation results. The tools described in Table 5, were submitted to comparison tests to assess which tool should be integrated in the framework of *TranSyT*. Thus, eight tools that use different methods for prevision were analysed: BOMP, MCMBB, BetAware, PRED-TMBB2, parti-Fold, BOCTOPUS2, TMBpro, TMB-KNN. The remaining tools presented issues in the webservers that did not allow obtaining results (ConBBPRED, TMBETA-NET, TMB-Hunt, PredβTM).

Regarding tools using the same prediction method, only the latest ones were considered. As the output of each tool is slightly different and not all of them predict the number of β-strands, all outputs were converted to a Boolean (yes/no) value for each sequence.

3.2.1 *Construction of the dataset*

A dataset encompassing 25 proteins, unavailable in the tools' training datasets, was assembled using information from UniProt and OPM database to find the most reliable and efficient tools for β-barrel predictions. OPM is a curated source of information about proteins' three-dimensional structure in the lipid bilayer, structural classification, topology and intracellular localization, which also provides the Positioning of Proteins in Membranes 2.0 webserver (PPM 2.0) (83) for calculating the spatial positions of proteins in membranes. OPM was used to select the proteins to be included in the database, because unlike OMPdb, this database is not used by any of the assessed software and is manually curated.

Although some of these tools were designed specifically to predict β-barrels in the outer membrane of gram-negative bacteria, three distinct organisms were selected for the construction of this dataset, namely *Homo sapiens*, *Escherichia coli* and *Arabidopsis thaliana*, with the purpose of testing the versatility of the tools regarding different organisms and membranes. The datasets used for training the tools with machine-learning based methods were taken into account for the construction of the test dataset. Thus, none of the sequences used for training were selected for the test dataset, avoiding biased results.

The *Homo sapiens* and *Escherichia coli* proteins' datasets encompassed 10 proteins, for each organism: 5 β-barrels, 3 α-helical proteins and 2 β-stranded proteins (non-β-barrels). Regarding *Arabidopsis thaliana*, only 5 proteins were available in the aforementioned databases and thus used. Such proteins included: 1 β-barrels, 2 α-helical proteins and 2 β-stranded. The presence of α-helical proteins in the dataset has the objective of testing the capability of each tool to differentiate between TMB and TMA proteins. The β-stranded sequences are

meant to assess which tools find the correct tertiary structure of the protein (barrel) and not only the presence of β-sheets.

3.3 Validation of TCDB's information

Due to the lack of organized information in TCDB regarding the metabolites carried by each described transporter, there was the need to create an internal database for *TRIAGE* that was manually curated by several contributors. Here, for each entry, it was possible to find TCDB's information in a structured format, including transporter description, generic transport reactions, type of transport (antiport, symport, uniport, *etc*), direction, reversibility, and the metabolites being transported. The task of determining which metabolites were being carried was performed manually, through the revision of published articles cross-referenced on TCDB's web page for each transport system, as well as the assembly of the generic transport reaction in a predefined format. Thus, there was a need to develop a method that performs this task automatically.

Recently, TCDB included a "substrates"' field in each transport system's entry. However, the method used to gather this information was not fully disclosed by TCDB. Hence, there was the need to validate this information against *TRIAGE*'s internal curated database.

Therefore, a novel methodology that retrieves all online information available in TCDB was developed. This method accesses the public FASTA file available at the mapping files option in a static Uniform Resource Locator (URL) (http://www.tcdb.org/public/tcdb). Then, a parser will be specifically developed to read this FASTA. This parser will allow to identify each entry by the pair accession number and TC number. After listing all entries, a web scraper able to find the web page relative to a specific transporter entry (accession number, TC number) and retrieve all information, will be developed. The following URL was used to access these pages: "http://www.tcdb.org/search/result.php?acc=ACCESSION&tc=TCNUMBER", in which "ACCESSION" and "TCNUMBER" are replaced by the accession number and TC number to search, respectively. It is important to search using both fields, as different TC numbers are associated to one or more accessions and vice-versa. Whenever one TC number has several accessions, this is evidence that such transport system is composed of distinct subunits. This can be observed in TC number 3.*D*.5.1.1, which requires 7 accessions to form the Na-NDH enzyme complex. Alternatively, an accession can also be present in several TC numbers, which unlike the previous case, is not as trivial to understand. Milton Saier, in a personal communication via email, explained that this situation represents the same protein participating in two different systems. For instance, entries with accession P94360, ATPase encoded by gene MsmX is used to energize at least two systems, 3.*A*.1.1.26 and 3.*A*.1.1.34, and must therefore be included in both cases.

After retrieving all data, a preliminary analysis revealed that a restricted number of metabolites were misspelled in either TCDB or *TRIAGE*'s internal database. Moreover, the same metabolites were occasionally written with different synonyms (i.e. sodium, Sodium ion, Na, Na+, Na1+...). This led to the creation of a manually curated dictionary of "synonyms and incorrections". This dictionary has a keyword and synonyms or common misspells associated with that key. Examples of such entries are listed below:

- **Ca2+** =ca2+; calcium(2+); calcium; calcium2+; calciumions; ca+;
- **Endolysin** =endolysin; mureinhydrolase; lysin;
- **Phosphatidyl serine** =phosphatidylserine; ps;
- **NO2-** =nitrite; no2; nitriteanion; nitriteion; nitrousacid; nitrit; [no2]; no(2-); itrite; itriteanion; itriteion; itrousacid; nitrit.

The words were written without white spaces, as metabolite names can have different orthography variants, regarding white spaces.

4

SOFTWARE DEVELOPMENT

4.1 *merlin*'s IMPROVEMENTS

Releasing a new and improved version of *TRIAGE*, i.e. *TranSyT*, is a complex endeavour; hence, urgent improvements/updates were performed in the parents framework's source code, to allow the release of *merlin*'s 3.5-beta version. This task allowed to deepen the knowledge regarding both software's source code and to determine which should be the major improvements and how these should be implemented. Below, some of these improvements.

4.1.1 *Database services*

Although *merlin* currently uses MySQL™ or H2 as internal relational database management system, the Hibernate framework (http://www.hibernate.org/) is currently being implemented to map java objects to database tables. However, this project is still ongoing and will not finish before *TranSyT* is available. All SQL queries were clustered according to its "module", to simplify the implementation of Hibernate in *merlin*.

For instance, for *TRIAGE*, all queries regarding the transporters module were moved to the Java™ class "TransportersDatabaseServices", and for the compartments module to the "CompartmentsDatabaseServices", replacing the queries by generic methods that will temporarily continue to use these queries while Hibernate is not implemented in *merlin*. This step, besides expediting the transition to Hibernate, also revealed how important it is to create a modular program, to allow the easy replacement of outdated software components. This concept was taken into account when creating *TranSyT*.

4.1.2 *Compartments parsing and integration*

For compartments predictions, as aforementioned, *merlin* was able to parse the results provided by PSORTb 3.0 and WoLF PSORT reports. Nevertheless, due to updates in both software's output files, existing parsers had to be updated and new parsers had to be

created (WoLF PSORT). While refactoring these parsers, a new one was created to offer support for LocTree3, a software tool able to predict compartments in Eukaryotes, Bacteria, and Archaea. The results loading process was also updated, and in the new version the user has to provide the web link for WoLF PSORT and LocTree3 results. This option was not available for PSORTb 3.0, as this software sends the results file by email, restricting the direct access to the user itself.

As shown in figure 8, TCGs should to be located in membranes. After the confirmation that a gene encodes a transport protein, the sub-cellular location is used to determine between which compartments a compound is carried. Manually annotated reactions retrieved from *TRIAGE*'s internal database are generic and the compounds are allocated either to the inside (in) or outside (out) of a putative membrane. An example of such reaction is $H^+(in) \longleftrightarrow H^+(out)$, in which "*in*" represents the inner compartment of the membrane that will receive the transport protein, and "*out*" the outer compartment of the same membrane.

While generating transport reactions, whenever an error occurred when a compartment was not recognized, *TRIAGE* would assume the default compartments (in - cytoplasm, out - extracellular). As the inner and outer compartments were inferred separately, eventually transport reactions that carried metabolites from a compartment to itself were created (i.e. $H^+(Cytoplasm) \longleftrightarrow H^+(Cytoplasm)$), which is obviously not a transport. This error was overcome by creating a controlled dictionary and a controlled structure that determine the inner and outer compartments of a specific membrane. In table 8, a small example of such structure is shown.

Table 8.: Example of inner and outer compartments for the indicated membranes.

Membrane	Outer Compartment	Inner Compartment
Cytoplasmic Membrane	Periplasm*	Cytoplasm
Vacuole Membrane	Cytoplasm	Vacuolar lumen
Outer Membrane	Extracellular	Periplasm
Mitochondrial Membrane	Cytoplasm	Mitochondrial lumen
Cell Wall	Extracellular	Cytoplasm
Default compartments	Extracellular	Cytoplasm

*Periplasm for gram-negative bacteria, Extracellular otherwise

Another major improvement was the upgrade of *merlin*'s *TRIAGE* compartments annotation panel, making it more appealing and less ambiguous. Now, to select the desired threshold for secondary compartments, the user selects one of the values available in the drop-down list. Thus, the secondary compartments (if any) can be taken into account whenever scores differ less then the user-defined percentage when compared to the primary com-

partment. This threshold can be set from 0% to 50% in increments of 10% (more than 50% is not available because a single secondary compartment does not have a score difference superior to this value). Likewise, the option "compartments to ignore", associated with the compartments integration process was replaced by a different method. The process of integration allows to assign the compartments predictions to the reactions encoded in the genome. Thus, currently, when the button "integrate to model" is clicked, a new window is opened asking the user which, if any, of the compartments available in the compartments predictions are supposed to be ignored during integration. This is an important feature as often these predictions return compartments such as "unknown". In these cases, the user can ignore such compartments, and a default compartment is assumed.

4.1.3 *SamPler*

Besides all aforementioned improvements, a new feature was added to *merlin*. SamPler (26), a semi-automatic method for annotation of enzymes, was developed and integrated to *merlin*. When *merlin*'s homology search is complete, a score is assigned to each EC number using an equation similar to Equation 2 to each annotation using taxonomy score, frequency score and a parameter α. However, the best value for the parameter α is not automatically discovered and has to be calculated manually or using SamPler.

To determine the best value of α, the user has to manually annotate a small sample of the genome (5% to 10%), allowing *merlin* to accept this sample as its standard of true. Next, SamPler starts a wide set of calculations that allow determining the best value of α, upper threshold, and lower threshold. Above the upper threshold, all entries are accepted as correctly annotated, and below the lower threshold all entries are rejected as non-metabolic genes. The user must then perform the manual annotation of all genes between the upper and lower thresholds. Despite saving time when compared to the manual annotation of the entire genome, this two stage annotation task can still become a very laborious and time-consuming procedure for larger genomes.

4.1.4 *Annotation workflow*

This project ultimately led to the idea of another project with the purpose of improving *merlin*'s enzymes annotation capabilities and researchers productivity during annotation. Thence, a fully automatic method for annotating enzymes in *merlin* was proposed, in a collaboration project with a first year master's student.

When manually annotating a genome, researchers usually follow a strict workflow individually designed, specifically for the genome in study. The idea for the fully automatic method was that these pipelines can be adapted for automatic implementation in *merlin*, an-

notating all entries using information already available in its internal database, consistently replicating the user's task without the human error factor. A user-friendly Graphical User Interface (GUI) with a maximum of 9 slots was designed and implemented to collect the pipeline's parameters, which allow replicating most of the previously developed workflows. This interface was divided into 4 different fields:

- The **first field** is composed of a drop-down list where the user can select if the step selected will be relative to the species or the genus. If the empty option is selected, the remaining fields become inactive and excluded from the pipeline;
- The **second field** is composed of a drop-down list containing all species or all genus available in *merlin*'s internal database, depending of the option selected in the first field. Also the option "any" is available for both options, in order to accept any species or any genus;
- The **third field** is relative to the E-value threshold. Here, the user can define the maximum threshold above which all annotations must be disregarded;
- The **fourth field** is a Boolean option where the user can select between matches with UniProt reviewed entries or not.

In case that none of the conditions in each slot of the workflow is fulfilled, the user has available an extra Boolean field that allows the entries to accept the default annotation when no matches are found.

The process of assessing the correct annotation, for each gene, starts by searching the best match among the homologous genes found in the similarity search. With this purpose, an entry that meets the requirements of the first slot is sought among all available homologous genes. If no entries match the requirements, the process is repeated for the next slot, until the entire workflow is iterated. If no entries meet the requirements of any slot, the default annotation can be accepted or rejected, depending on the value of the Boolean option "Accept default annotation if no match is found".

To each annotation a confidence level is associated depending on which slot of the workflow the match was found. The confidence level is expressed in a sequential order with alphabet letters, spanning from slot 1 (letter A) to slot 9 (letter I). The default annotation also has a confidence level of "Default". When the evaluation process is complete, *merlin* saves the annotation into its internal database and refreshes the Enzymes Annotation panel to reveal the results of the process. The confidence level of each annotation is shown in column "notes".

4.2 IMPLEMENTATION OF *TranSyT*

After validating TCDB's data and performing the study over the best method for *TRIAGE*'s TMBs identification, the next stage was starting the implementation of *TranSyT*. As the objective is to create a modular software, the implementation will be divided according to the four main functionalities:

1) TCDB scraper and family-specific transport reactions assembly;

2) Metabolites identification and hierarchical ontology;

3) Generation of transport reactions for each transport system;

4) Identification of genes encoding transport systems (genome's transporters annotation);

The modularization principle was also taken into account when developing each of the main systems.

4.2.1 *TCDB scraper and family-specific transport reactions assembly*

The first step was to retrieve the information available in TCDB for each transport system, to generate the family-specific transport reactions. The scraper used to retrieve the information for the validation process was slightly modified and re-implemented at this stage to perform this task. As before, the latest FASTA is used, to guarantee that each time the software is executed, all entries available at TCDB are utilized, and a backup record of the FASTA file is saved. At this stage, the fields "Description", "Accession Number", "Species", and "Substrates" are retrieved and saved in *TranSyT*'s internal database.

The next stage was the implementation of a method to search for the webpages containing the TC family information regarding the transporter systems saved. The URL `http://www.tcdb.org/search/result.php?&tc=TCFAMILY`, in which the "TCFAMILY" was replaced by the TC family number to be retrieved (1.A.1 for TC numbers 1.A.1.1.1 and 1.A.1.2.1, for example), was used To access these pages. Here, information like superfamily, family, and family-specific transport reactions can be retrieved. For instance, TC family 1.A.1, belongs to the "VIC Superfamily" superfamily, whereas its family is "The Voltage-gated Ion Channel (VIC) Superfamily". The generic reaction is $cation\ (out) \longleftrightarrow cation\ (in)$. However, often this information is missing for several TC families. All relevant data is retrieved using a parser developed to process these HTML pages. As the information is unstructured and without a structured format, errors are common during the parsing process. For instance, the superfamily description is always found as it is always in the same position. However, it

is more demanding to retrieve the family description, as frequently the scraper returns more text than it should. Retrieving the transport reactions fields is even more challenging, as the reactions are embedded in the webpage text, increasing the complexity of finding and parsing them. Hence, several alternatives were included into the parser such as seeking for known signs ($+, -, \rightarrow, \leftrightarrow$) or sentences after a colon (":"), as generally these equations follow this punctuation mark. However, the fact that this punctuation mark is used in several other occasions hardens this task. Other issues, like the reaction being so long that it needs to be displayed in two or more separated lines, or the arrows separating the reactants from the products being in different forms of text for the same symbol, or even images, turned this task into an automatic text processing challenge.

After gathering all reactions the objective is to decompose them and assess which are products or reactants, their compartments, reversibility, and transport type. Here, automatic text processing is used again. While retrieving the reactions, the reversibility indication was instantaneously identified and replaced by *TranSyT*'s own reversibility. This token is an unambiguous symbol used by *TranSyT* to indicate the reaction's reversibility. This makes the reactants, products and reversibility identification easier at this point. To find the transport type, the transport between relative compartments needs to be identified first. Implementing this part was, once again, challenging, as compartments information is often between parentheses, but not always. Moreover, in seldom cases, the information between parentheses is not related to the compartmentalization process. Occasionally TCDB is explicit and instead of the required *in* or *out* compartments, it has specific locations throughout the cell. Again, a dictionary similar to the one used during TCDB's validation process was created using the same principles to solve these inconsistencies. A small example of the entries listed in the dictionary is available below:

- **inner_membrane** =innermembrane; innermonolayeroftheplasmamembrane;
- **outer_membrane** =outermembrane; outermonolayeroftheplasmamembrane;
- **lysosome** =lysosome; intralysosomalspace; lysosomallumen; lysosomal;
- **endosome** =endosome; endosomallumen; endosomal;
- **periplasm** =periplasmofgramnegativebacteria; periplasm; innermembraneperiplasmicside.

Then, after identifying the two different compartments between which the transport takes place, the identification of their relative position to each other is determined. This step involved writing an organized array of compartments, which starts with the most outer compartment, and ends with the lumen of the organelles present in the cytoplasm of the cell (most inner compartments). As each position of the array is numerated, the lowest

position is the *out* and greatest the *in*. An example of such structure is shown in table 9. For demonstration purposes, the column "Group" was added. The positions among the members of each group is not relevant, as transport between compartments of the same group are not biologically feasible. For instance, compounds can be transported from the chloroplast to the cytoplasm across the chloroplast membrane, but not between the chloroplast and another organelle, without passing through other structure such as the cytoplasm first.

Table 9.: Structure used to decide the relative positions between compartments.

ID	Compartment name	Group
0	out	
1	cells	
2	extracellular	**A**
3	xylem	
4	phloem	
5	outer_membrane	**B**
6	periplasm	**C**
7	cytoplasmic_membrane	**D**
8	inner_membrane	
9	cytoplasm	**E**
10	endomembrane	**F**
11	endoplasmic reticulum_membrane	
12	peroxisomal_membrane	**G**
13	chloroplast_membrane	
14	mitochondrial_membrane	
15	chloroplast_intermembrane_space	**H**
16	mitochondrial_intermembrane_space	
17	mitochondria	
18	chloroplast	
19	intravesicular	
20	endoplasmic reticulum	
21	nucleus	
22	golgi	**I**
23	lysosome	
24	endosome	
25	peroxisome	
26	vacuoles	
27	organelle	
28	in	

The results are then saved in files encoded in JavaScript Object Notation (JSON) format. At this stage, family-specific reactions are yet to be generated, as seldom *and/or* rules can be present in the reaction string. For instance, the below reaction:

- $SO42^{-}\ or\ CrO42^{-}\ (out)\ +\ nH^{+}\ (out) \longrightarrow SO42^{-}\ or\ CrO42^{-}\ (in)\ +\ nH^{+}\ (in)$,

will return the following reactions:

◇ $SO42^{-}\ (out) + nH^{+}\ (out) \longrightarrow SO42^{-}\ (in) + nH^{+}\ (in)$

◇ $CrO42^{-}\ (out) + nH^{+}\ (out) \longrightarrow CrO42^{-}\ (in) + nH^{+}\ (in)$

Thus, the objective is to find all rules using automatic text processing, and infer all family-specific reactions.

The next step is to infer the type of transport of the family-specific reactions. At this stage, all compartments were already converted into *in* or *out*. Thus, the identification of the type of transport can be performed taking into account the following rules, when reactants and products have the same metabolites and the following conditions on each side of the reaction:

- one compartment and one metabolite $\Longrightarrow$ **Uniport**
- two or more metabolites and one compartment $\Longrightarrow$ **Symport**
- two or more metabolites and two compartments $\Longrightarrow$ **Antiport**

Other transport systems that do not comply with these rules are classified according to information collected from TCDB, as for example, transporters belonging to TC family 3.*A*.1, which are known as ABC transporters. Despite this classification, there are cases in TCDB that have several family-specific transport reactions, including different type of transport. For such cases, automatic text processing is used to search in the description of the transporter system for evidences of the correct type of transport. For instance, keywords like "symport(er)", "uniport(er)", and "antiport(er)" are sought in the description, and ultimately, for cases in which no evidences are found, in the subfamily description. At this stage errors from previous processes are solved. A file containing exceptions was created for this purpose. All errors not solvable through programming methods should be added to this file. Then, all entries that have a corresponding match in this file are replaced by it. Examples of such exceptions are the following cases that can be found for TC families 3.*A*.8, 5.*A*.1, and 4.*C*., respectively, in which the family-specific transport reactions are exhibited as follows:

1) $protein\ (cell\ cytoplasm)\ protein \longrightarrow (mitochondrial\ matrix\ or\ membrane)$

2) $2\ e^{-}\ _{cytoplasm} \longrightarrow 2\ e^{-}\ _{periplasm}$

3) $Acyl - CoA \longrightarrow FattyAcid\ Coenzyme\ A$

In the first case it is possible to observe that the substrate "protein" that should be present in the products is displayed in the wrong side of the equation. The second case has the compartments written in subscript and not between parenthesis. As *TranSyT* seeks parenthesis for the compartmentalization, these compartments would be considered as substrates. For the third case, the fact that the products of the reaction are not separated by the character "+", will lead to the interpretation that such string represents a single metabolite named "Fatty Acid Coenzyme A". Known exceptions like the ones described here and others, occur for 56 TC families in a universe of 1 176, which corresponds to less than 5%. Below, examples of the correction of the reactions manually set in the exceptions file.

1) $protein\ (cell\ cytoplasm) \longrightarrow protein\ (mitochondrial\ matrix\ or\ membrane)$

2) $2\ e^{-}\ (cytoplasm) \longrightarrow 2\ e^{-}\ (periplasm)$

3) $Acyl - CoA \longrightarrow Fatty\ Acid\ +\ Coenzyme\ A$

Solving these inconsistencies programmatically is not feasible, as the creation of universal exceptions within the code to deal with similar cases, would create conflicts. Correct entries would also be caught by these exceptions and rendered incorrect.

Another example is that occasionally TCDB does not displays the reversibility of a reaction (e.g. $cation\ [out]\ cation\ [in]$). These occurrences have to be manually analyzed. Finally, the results are once again saved in a file with JSON encoding, and the generation of family-specific transport reactions is complete. A simple schematic of all stages of the process is illustrated in figure 12.

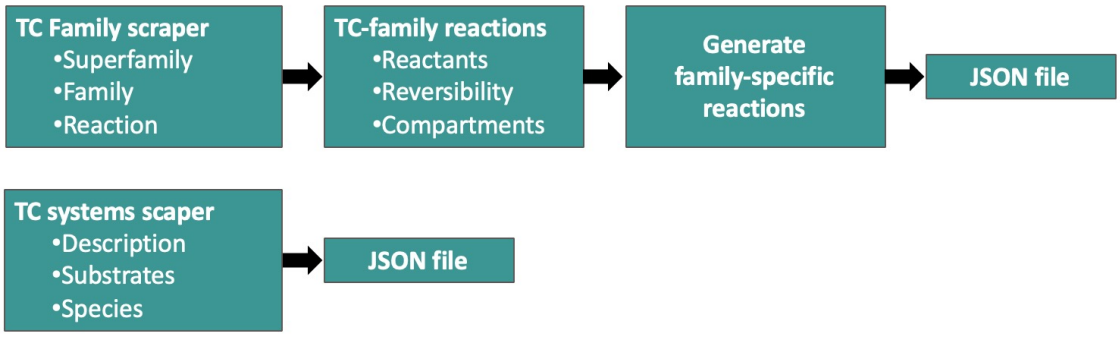


Figure 12.: TCDB scraper and family-specific transport reactions assembly process

4.2.2 *Metabolites identification and hierarchical ontology*

The metabolites retrieved during the previous phase are not annotated with database identifiers, as TCDB does not provide cross-references for the substrates described. Hence, it is of paramount importance to provide identifiers to the metabolites present in reactions. Without identifiers, these reactions cannot be integrated with metabolic reconstructions. As stated by Thiele and Palsson at step 14 of the protocol for reconstructing GSM models, if the metabolites do not possess identifiers of known databases, they cannot be recognized by other scientists and software tools (130).

Biosynth (81), an open-source Bioinformatics software developed within the BioSystems group at Centre of Biological Engineering (CEB) by Filipe Liu and available at https://github.com/Fxe/biosynth-framework, was used for this task. Biosynth is capable of retrieving information from several online data sources and creating links between all information, based on cross-references available at each source, creating a local Neo4j graph database. Among all features present in Biosynth, only the module to collect compounds information was used. Thus, Biosynth was used to retrieve compounds information, from the following databases: KEGG, BiGG, MetaCyc and ModelSEED. Unlike *TRIAGE* that uses ChEBI, *TranSyT* uses MetaCyc to determine the metabolites hierarchical ontologies. The reason for this transition is that ChEBI is too extreme when establishing relationships between metabolites, creating unwanted relationships for modelling purposes, which often leads to cyclic relations. Tis characteristic is not observed in MetaCyc hierarchies. odelSEED was also one of the choices as the objective is to integrate *TranSyT* in KBase. esides KEGG identifiers, also ModelSEED and Biochemical, Genetic and Genomic (BiGG) identifiers are planned to be supported by *merlin* in its future versions.

Biosyn th creates relationships between metabolites that are previously cross-referenced in the source databases, thus, in some cases links between data might be missing. For example, if compound-X from MetaCyc does not have references to compound-X in KEGG, these entries will be regarded as distinct compounds. Hence, when gathering information from the graph database, instead of listing the metabolites by ID, these were listed according to their names and aliases. Nevertheless, in seldom cases the same name is associated with different metabolites. For instance, in MetaCyc, both Fructose and D-Fructose, although being different entities, have the alias "fructose" in common. Although being related, an incorrect selection at this stage will lead to different outcomes when searching the related metabolites in the hierarchical ontology.

Therefore, a method that iterates over all nodes present in Biosynth's metabolites names was developed to organize the data, avoiding collision between metabolites with the same aliases. This method looks for all metabolites associated with the name being iterated based on a default order of selection (KEGG, ModelSEED, BiGG, and finally MetaCyc). When

the search is complete, all identifiers found in the search process are associated with the name. Special cases, such as the duplication of metabolites in ModelSEED are addressed by selecting the identifier with the lowest numeric value in the identifier, as those are the commonly used ones. For instance, the same D-Glucose has two different identifiers with the same chemical formula: cpd00027 and cpd26821. In figure 13 an hypothetical scenario of the previous situations is represented.

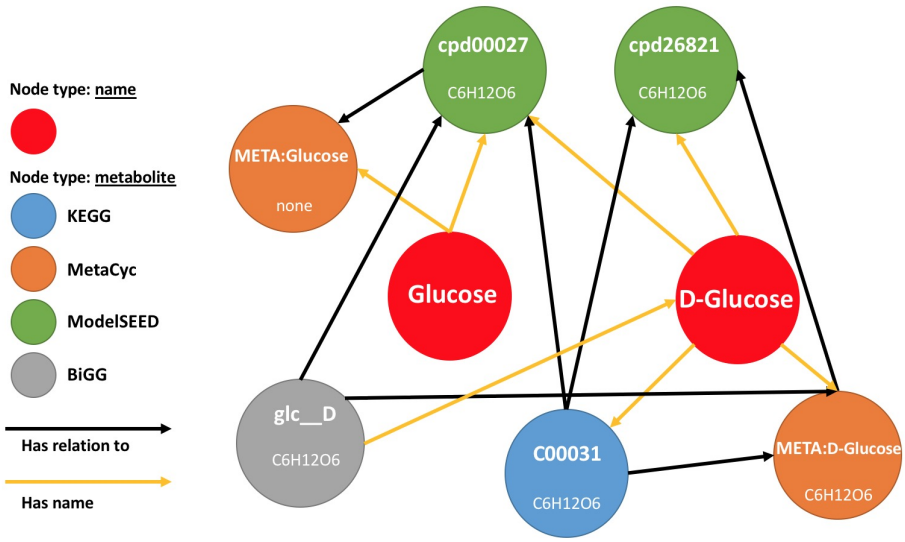


Figure 13.: Example of an hypothetical relationship that can be found in Biosynth's graph database.

In this hypothetical scenario, it is possible to observe the relationships between the metabolite identifiers of the four different databases and two names. According to the previous rules, the annotation would be as follows:

- **Glucose -** cpd00027; C00031; glc__D; META:Glucose.
- **D-Glucose -** C00031; cpd00027; glc__META:D-Glucose.

This process lists all names and aliases, including identifiers, that describe metabolite in Biosynth's database (currently over 154 000 names). The family-specific transport reactions, as well as the substrates associated with each transport system were retrieved from TCDB in the previous stage. Hence, the next step involves assigning database identifiers to all metabolites retrieved from TCDB. However, not all metabolites will have a direct match, thus an algorithm that performs automatic text processing was developed. For every metabolite described in TCDB, matches are sought in the following logical sequence, until a match is found. When a match is found, the metabolite is then removed from the set of metabolites to search.

1. **Direct match -** identification through direct, case sensitive, match. This allows cases like CO (carbon monoxide) not be associated with Co (cobalt).
2. **Remove stoichiometric coefficients -** often metabolites retrieved from reactions present in TCDB have stoichiometric coefficient. By removing the stoichiometric coefficient, direct matches are possible.
3. **Case insensitive match -** a case insensitive search of the compounds name allows assigning identifiers to some metabolites.
4. **Non-alphanumeric characters removal -** metabolites names may contain white spaces or other characters that impair automatic matching. Removing these non-alphanumerical characters usually solves the problem for several entries.
5. **Add prefix "D-" or "L-" to enantiomers -** such as sugars and amino acids. For instance, mycarose, requires either prefix L- or prefix D- before the name, otherwise matches are not found for this case.
6. **Case insensitive and non-alphanumeric characters removal -** a combination of methods 3 and 4.

When the matching process ends, metabolites from TCDB have the corresponding ID associated. A summary of the entire process of metabolites identificatio is described in figure 14.

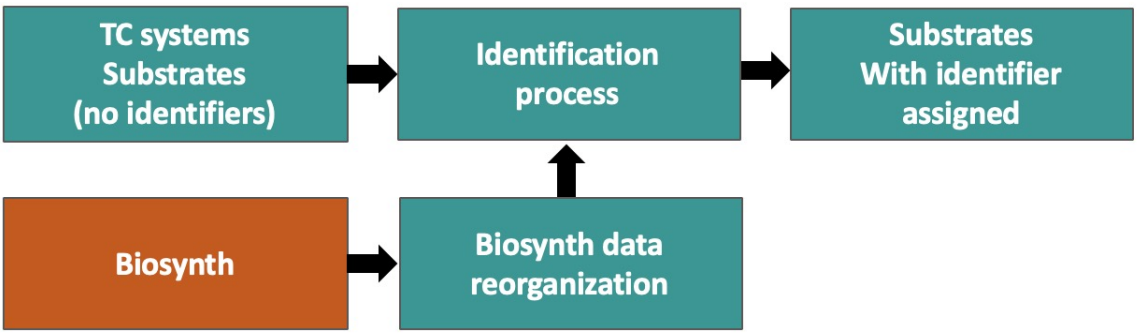


Figure 14.: Metabolites identification process.

4.2.3 *Generate system-specific reactions*

The next stage encompasses associating the substrates retrieved from each transport system, annotated with database identifiers in the previous stage, with the family-specific reactions. This process allows creating the final substrate specific transport reactions, which will be eventually assigned to each transport system. The metabolites assigned to each transport system are used to replace the appropriate metabolites in the respective family-specific transport-reaction, retrieved from TCDB and assembled in section 4.2.1. During this step, the ontological descendants of the substrate are sought in Biosynth's graph database. Although MetaCyc provides the hierarchical ontology, the descendant metabolites will be assigned with the identifier selected by the user, when available. Selected cases, such as "lipid" (mostly for generic metabolites), encompass thousands of descendants. Thus, *TranSyT* allows specifying the maximum number of generations to selected metabolites. The default value of -1, does not impose any limit, 0 is the metabolite itself, and higher than 0, the desired limit. Metabolites not listed as special cases are assumed as unlimited (-1). Descendant metabolites will provide reactions similar to their parent (replacing it in the equation) and will be associated with the same transport system. Using the metabolites molecular formulas, the created reactions are tested for mass balance, and only balanced transport reactions will be accepted. However, in known cases, the reactions require balance correction due the origin of the annotation. For instance, ABC transporters with metabolites formulas retrieved from MetaCyc, BiGG, and ModelSEED will be lacking a proton. Alternatively, when using KEGG data, the reaction will require a molecule of water. Identified cases are automatically corrected when the reactions are generated. *TranSyT* generates its reactions based in evidences of the TC family reaction equation and also of the description of the system, subfamily, family, and superfamily. If no evidences are found, the simplest mechanism is assumed (reversible uniport). If symport or antiport evidence is found but no default reaction is available, the co-transport of the metabolites described in the system is performed with a proton.

When the reaction is approved, a hash code of the reaction is generated and sought in *TranSyT*'s internal persistent database. If a match is found, the same ID is used to maintain database integrity throughout versions. Otherwise, a new ID is generated and the reaction hash code stored.

To generate the Transport Reaction identifier (TR identifier)s, a mechanism that easily allows to understand the reactions encoded by the hash was developed and implemented. The only common element to all identifiers is the sequence of characters "TR", initials that stand for "Transport Reaction", that can also be preceded by the letter "i" in case of irreversible reactions, and followed by the abbreviation of the type of transport. Next, all reactants are alphabetically sorted and concatenated, followed by the respective compart-

ment("i" for "in", "o" for "out", or none of them in case of no compartment evidence), separated by the character "_". Below, examples of identifiers using metabolite's ModelSEED IDs and the respective encoded reactions:

- **TRUni__cpd001170** - $D-Alanine\,(out) \longleftrightarrow D-Alanine\,(in)$
- **iTRUni__cpd00117i** - $D-Alanine\,(in) \longrightarrow D-Alanine\,(out)$
- **TRSym__cpd00034i_cpd00067i** - $Zn^{2+}\,(in) + H^{+}\,(in) \longleftrightarrow Zn^{2+}\,(out) + H^{+}\,(out)$
- **iTRAnt__cpd00034i_cpd00067o** - $Zn^{2+}\,(in) + H^{+}\,(out) \longrightarrow Zn^{2+}\,(out) + H^{+}\,(in)$
- **iTRabc__cpd00002_cpd001170_cpd00001** - $ATP + D-Alanine\,(out) + Water \longrightarrow ADP + D-Alanine\,(in) + Pi + H^{+}$

The last step of this module is to store the information in a suitable database. Neo4j graph database was selected for this step, due to its performance over relational databases when dealing with densely connected data (55). Thus, all identifiers and TC numbers are the nodes of the graph, and information like descriptions, names, formulas, and reactions are properties of the nodes, and the edges (e.g. "has_tc", "has_reaction" or "has_metabolite") provide the type of relationship between nodes. All relationships are unidirectional, with an accession number having none, one, or multiple TC numbers associated, and one TC number owned by one or multiple transport reaction identifiers.

Before initiating the storage process, the taxonomy relative to each accession number is also retrieved using UniProt API, due to the importance of the phylogenetic information inherent to each transporter system. This is not used at this stage but it is important for the last module of *TranSyT*. This search is performed while the server is creating or updating the database, and this data will be available for users, during the identification stage. For the storage process, multi-threading was not used because two threads can be creating the same nodes or relationships at the same time. In a laptop running Windows 10 64-bit, processor Intel(R) Core(TM) i7 first generation CPU 1.60 GHz, and 8 GB of RAM, this process takes approximately 6 hours due the amount of information and relationships to store (58 623 different reactions currently).

A small example illustrating the behavior of the graph database for 5 different accessions is shown in figure 15. This figure allows visualizing the 3 subunits that integrate the transport system with TC number 3.*A*.1.9.1. This figure also illustrates the aforementioned example, where one single accession is associated with two different transport systems (TC numbers). The high density of relationships is evident between reactions and metabolites, as 10 out of the 11 reactions displayed share common metabolites.

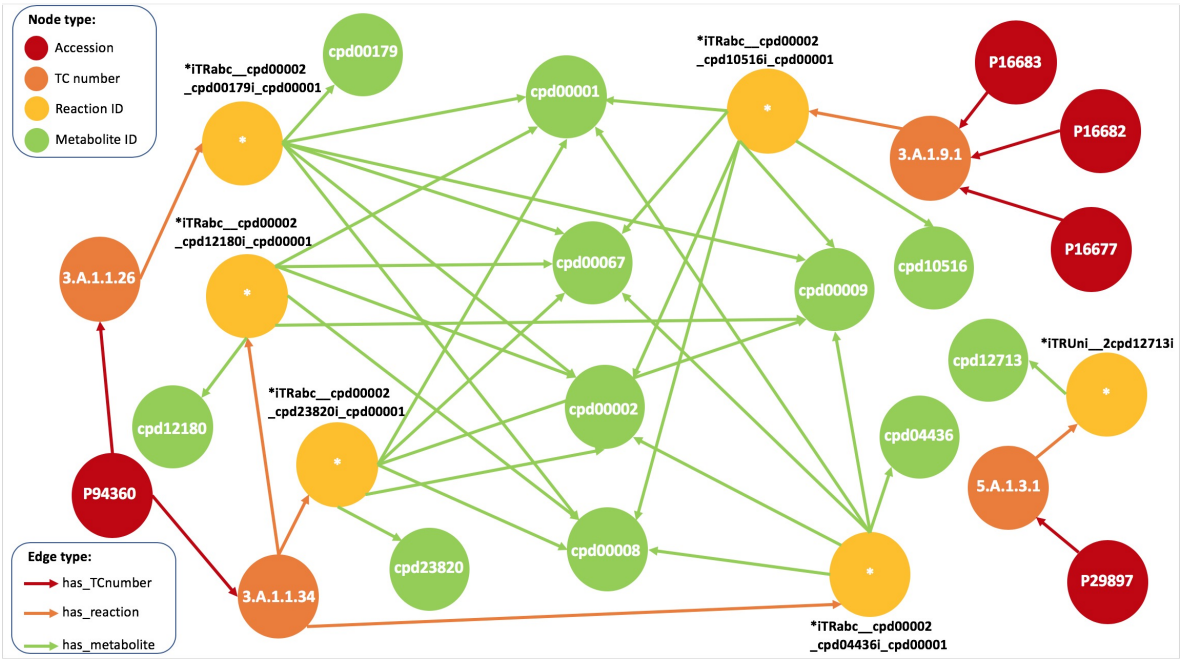


Figure 15.: Representation of *TransSyT*'s internal database for 5 accessions.

4.2.4 *Identification of genes encoding transport systems*

TranSyT's last module is the one with which the final users will interact. The inputs for performing a request to *TranSyT* are the genome and the taxonomy identifier of the organism being analyzed. As in TCDB itself, *TranSyT* uses BLAST to identify genes encoding transporter systems. BLAST was implemented using the plugin recently developed for *merlin* that uses the command line software BLAST+ (19) available for download in NCBI at `ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST/`, currently in version 2.7.1. Having BLAST+ installed in the machine that is running *TranSyT* is not a prerequisite, as *TranSyT* will run remotely in a server and send the results to the computer that made the request. If all conditions are fulfilled, *TranSyT* proceeds and performs the BLAST using the input genome against all sequences present in TCDB's latest downloaded FASTA file. This will be the the most time-consuming process of the last module, as the running time depends on the performance of the machine and size of the genome.

TranSyT sets the following BLAST configurations, by default: scoring matrix BLOSUM62, E-value threshold = 1E−10; query coverage threshold = 80%; bit score threshold = 50. The default BLAST parameters were set, taking into account the Pearson's study in *An Introduction to Sequence Similarity ("Homology") Searching* (100), in which it is stated that bit scores of 40 and E-values $\leq$ 1E−3 are significant for databases with less than 7 000 entries. As the E-value is dependent of the size of the database, an increase of 10 to the bit score increases the significance in a factor of 2^{10}, thus a bit score of 50 would be significant for a database

with less than 7 million entries (TCDB's FASTA file currently contains less than 18 000 entries). Hence, a low E-value, a high bit score and 80% coverage of the query sequence when querying a database as small as TCDB, can be regarded as a conservative approach. This strategy will increase the number of false negatives rather than the number of false positives. Nevertheless, all values are configurable.

After performing the alignments, the first step is to determine which TC family should be assigned to a given entry. This task involved devising an algorithm that calculates the score for each family among the BLAST hits for each entry. The family score is calculated according to equation 2, that takes into account the frequency of the family ($Score_A$) and the similarity ($Score_B$) of each hit, balanced by the parameter α that can take values between 0 and 1, (default value of 0.4, configurable). The default value of α was attributed to favour the frequency without disregarding the similarity. Equation 2 was used in *TranSyT* for several calculations as it allows taking into account several parameters.

As shown in equations 7 and 8, the frequency score is calculated by dividing the number of hits for the TC family F, by the total number of BLAST hits. Similarly, equations 9 and 10 show that the similarity score is calculated by dividing the sum of the similarities of the hits belonging to family F by the sum of the similarities of all hits.

$$Score_A = \frac{\sum_{i=1}^{H} F_i \times Vf_i}{\sum_{i=1}^{H} F_i} \tag{7}$$

$$VF_i = \begin{cases} 1, & if\ record\ i\ belongs\ to\ family \\ 0, & otherwise \end{cases} \tag{8}$$

$$Score_B = \frac{\sum_{i=1}^{H} S_i \times Vfs_i}{\sum_{i=1}^{H} S_i} \tag{9}$$

$$VFs_i = \begin{cases} 1 \times S_i, & if\ record\ i\ belongs\ to\ family \\ 0, & otherwise \end{cases} \tag{10}$$

In table S1 of the supplemental material, one can find the results of the alignments of UniProt entry S7V9F2, to the whole set of records available in TCDB. A summary of the results is presented in Table 10

Despite having a bit score of 955, the second BLAST hit's (8.*A*.1.6.1) family 8.*A*.1 has only 4 hits, whereas family 2.*A*.6 has 40 hits. The high frequency and similarity scores, allows annotating this protein as belonging to 2.*A*.6 family. Despite this classification, this protein is present in UniProt with a different annotation - family 8.*A*.1; however, this is

Table 10.: Family annotation scores for UniProt entry S7V9F2, calculated using the default $\alpha = 0.4$.

TC Family	Frequency	Freq. Score	Sim. Score	Family Score
2.*A*.6	40	0.85	0.88	0.87
3.*A*.1	3	0.06	0.03	0.04
8.*A*.1	4	0.09	0.09	0.09

an unreviewed entry with 1 point of annotation score. Moreover, UniProt's description is "Efflux transporter, RND family, MFP subunit". According to TCDB, proteins labeled as "MFP subunits" belong to family 8.*A*.1, while the "RND family" belongs to the 2.*A*.6 family. In addition, both families are related, as proteins belonging to the latter family function in conjunction with proteins from family 8.*A*.1, that belong to class 8 "Accessory Factors Involved in Transport".

Therefore, *TranSyT*'s classification seems to be correct as family 2.*A*.6 had 40 BLAST hits against TCDB which contains 139 proteins belonging to this family. Whereas family 8.*A*.1 accounted for a total of 4 hits hits against TCDB which contains 19 entries for such family. This analysis show that the high number of hits is not correlated with the high availability of entries in such family, as the percentage of hits per family is higher in the former case.

The next step of this stage is the association of transport reactions to the genes identified as encoding transport proteins. *TranSyT* selects the reactions to associate to such genes (annotated with TC families), using two different methods: "auto accept" and "compute score".

The first method accepts all transport reactions that fulfill the following conditions:

- Reaction associated with TCDB entry belonging to the annotated TC family;
- TCDB entry hit with an E-value below the automatic acceptance threshold ($1E-50$ by default).

This method does not take the taxonomy of the BLAST hits into account, due to the high similarity with the query sequence. All reactions, available in *TranSyT*'s internal database, associated with TCDB entries that comply with these conditions will be assigned to the respective genes of the organism being annotated.

The second method, follows the same philosophy as in *TRIAGE*'s approach. The advantage of using this method is that reactions not selected by the first method can still be associated with the gene, if the BLAST hits' proteins belong to organisms with taxonomic classifications close to the case study. This score is calculated using 2 for each reaction associated hits. Thus, for each gene g the frequency of each reaction r within the homologous protein records is calculated ($Score_A$). The common taxonomy between query and subject sequences is also considered ($Score_B$). Leverage between score is provided by the parameter

α which takes values between 0 and 1. The default value of 0.8 for α was selected taking into account that TCDB has a limited number of organisms represented in its database, mostly belonging to Bacteria. Thus, if the case study organism does not belong to this domain, this value should be reduced.

As shown in equations 11 and 12, the frequency score is calculated by dividing the similarity of all homologous proteins promoting the reaction by the sum of the similarities of all BLAST hits, for each protein encoding gene. Thus, S_i refers to the similarity of the i^{th} BLAST hit encoding the reaction being scored, and H the total number of hits.

$$Score_A = \frac{\sum_{i=1}^{H} S_i \times Vr_i}{\sum_{i=1}^{H} S_i} \tag{11}$$

$$Vr_i = \begin{cases} 1, & if\ reaction\ r\ is present\ in\ record\ i \\ 0, & otherwise \end{cases} \tag{12}$$

Likewise, as shown in equations 13 and 14 the taxonomy score is calculated by dividing the taxonomy frequency t_i by the maximum taxonomy M_T, multiplied by the frequency of the protein hits promoting the reaction and a penalty. This penalty is put forward to penalize reactions that are associated to a low number of homologous genes by multiplying p_r with β, which is the penalty percentage with a value between 0 and 1 and set by default as 0.3. The taxonomy frequency is calculated by counting the common taxa between case study and the organism to which the i^{th} homologous record belongs.

$$Score_B = \frac{\sum_{i=1}^{H} t_i \times Vr_i \times (1 - p_r \times \beta)}{M_T \times \sum_{i=1}^{H} Vr_i} \tag{13}$$

$$p_r = \begin{cases} 0, & if\ \sum_{i=1}^{H} Vr_i \geq Min_{Hits} \\ Min_{Hits} - \sum_{i=1}^{H} Vr_i, & otherwise \end{cases} \tag{14}$$

Reactions with score above the defined threshold (0.5 by default) are associated to the protein encoding gene. All variables present in the calculations are parameterizable in *TranSyT*'s configuration files. To finalize the process, *TranSyT* uses the information regarding metabolites present in the model to filter the reactions. Reactions containing metabolites present in the model are accepted, all others are disregarded.

Unlike *TRIAGE*, *TranSyT* allows performing GPR associations. However, the search for protein complexes formed by multiple subunits encoded by different genes is not straightforward. *TranSyT*'s approach to this problem takes advantage of the information available TCDB regarding protein complexes together with the BLAST search results. The first step

of this methodology is finding genes associated to every subunit of each complex and the respective bit score, as shown in table 11.

Table 11.: Example of the data structure requested by the argument "blastData" in the algorithm 1.

TC Number	Accession	Query Gene	Similarity
3.A.1.9.1	**P16679**	Gene_1	0.7
		Gene_2	0.5
		Gene_3	0.66
	P16682	Gene_4	0.58
		Gene_1	0.25
		Gene_3	0.8
	P16683	Gene_2	0.5
		Gene_3	0.25

This method assigns reactions accounting for similarities between query gene and the subunits available in TCDB. For every TC number with evidence of forming a protein complex of multiple subunits, *TranSyT* searches the genes belonging to the family of the transporter system to create the GPR associations, using the method described in algorithm 1.

Algorithm 1 GPR association process - searchSubunits(arg1, arg2, arg3)

Input: *blastData* ▷ Structure like table 11
invertedMapping ▷ inverted mapping of columns "Accession" & "Query Gene"
assigned ▷ initially empty map

if *assigned*.size() == *data*.size() **then**
  return *assigned*
end if

allRemainingAccessions ← *blastData* keys and remove all keys in *assigned*
accession, *gene* ← **findBestGene**(*blastData*, *allRemainingAccessions*, *invertedMapping*, *assigned*) ▷ algorithm 2

if *invertedMapping*.get(*queryGene*) is empty or *querygene* == null **then**
  return null
end if
assigned.put(*accession*, *gene*)

return searchSubunits(*blastData*, *invertedMapping*, *assigned*)

In this method, for accessions with similarities with only one entry, the association is direct and automatically performed. For cases with several genes per subunit, the algorithm selects the genes with the highest bit score. When multiple hits have the same value of bit score, the lowest E-value is used as tie breaker. When a gene is associated with a subunit, other subunits are discarded from the gene entry. If the total number of genes is less than the total number of subunits, the association is reset, and the discard relations restored. The algorithm 2 will then select the next highest scoring gene for the unannotated subunit, and the previous methodology applied is recursively until all subunit are associated with a different gene, as shown in algorithm 1.

Algorithm 2 Find gene with highest score - findBestGene(arg1, arg2, arg3, arg4)

```
1:  Input: blastData
2:         allRemainingAccessions
3:         invertedMapping
4:         assigned
5:
6:  exclude ← empty set
7:  for each acc in allRemainingAccessions do
8:      if data.get(acc) size == 0 then
9:          return acc, data.get(acc)
10:     end if
11: end for
12:
13: found ← TRUE
14: while not found do
15:     accession, gene ← similarity with highest score not included in exclude
16:
17:     if gene is empty then
18:         return null
19:     end if
20:
21:     conflicts ← remove accession from data and gene from other accessions
22:     if conflicts == TRUE then
23:         data removed is restored
24:         found ← FALSE
25:         exclude.add(gene)
26:         gene ← empty string
27:     end if
28: end while
29: return accession, gene
```

4.3 COMPARTMENTALIZATION AND TMD IDENTIFICATION

The integration of *TranSyT* into *merlin*, will allow the input of two extra options: compartmentalization and TMD identification. The compartmentalization of all generated reactions is straightforward, as the direction of transport of each metabolite was defined beforehand. The integration process will employ the prediction tool already integrated with *merlin* for *TRIAGE*, including LocTree3. Thus, *TranSyT* supports LocTree3, WoLF PSORT, and PSORTb 3.0.

The TMD identification will involve using Phobius for TMAs and PRED-TMBB2 for TMBs. Phobius is already imbeded in *merlin*, but TMBs indentification is still not available. Thus a method that retrieves the predictions from PRED-TMBB2 webserver will be implemented. Before starting the development, an email was sent to the authors querying about the availability of a public API and source code for the predictor, to avoid the limitations of using the webserver. The answer was that, at that moment, no API was available and the source code of the tool was not available for sharing as it was part of a bigger package of tools yet to be published. Despite this answer, the authors agreed in upgrading the webserver to accept batches of 20 000 sequences, instead of 5 000 before. Hence, a method that accesses the webserver and retrieves the results of the identification was developed, as well as a parser for the results.

Proteins with TMAs can be located in membranes. Likewise, the classification of predictions containing TMBs has to be processed carefully, as β-barrels are found in the chloroplast membrane, and mithochondrial membrane of Eukaryotes and outer membrane of gram-negative Bacteria. The results of both types of predictors (compartments and TMD) can be combined to determine which entries have high evidences of being transporters, without considering the homology results. This analysis will allow assigning confidence levels represented by alphabetic characters spanning from "A" to "C", as shown in table 12. This method of classification allows annotating a system with the highest confidence (level A), if both predictions are in agreement. If α-helices or β-barrels are associated with location predictions in the correct membranes, and belong to the correct organisms, maximum confidence is assigned. However, if proteins are predicted to be in other compartments, confidence B is assigned. On the other hand, if the location prediction of a given protein is not a membrane, though showing evidences of having TMD, such protein is assigned with the intermediate level of confidence (level B). When proteins do not have evidence of being located in membranes nor having TMD, the lowest level of classification is assigned (level C).

This functionality will not be integrated to KBase version and will only be available for *merlin* users, as this classification requires the input of results retrieved from third party software tools.

Table 12.: Classification of a transport system using TMD and compartments predictions.

Compartment Prediction	TMD Prediction		Confidence Level
Outer membrane Chloroplast membrane Mitochondrial membrane	Alpha-helix		A
	Beta-barrel (Gram-negative Bacteria or Eukaryote?)	yes	A
		no	B
	none		B
Extracellular Xylem Phloem Periplasm Cytoplasm Mitochondria Chloroplast Intravesicular Endoplasmic Reticulum Nucleus Golgi apparatus Lysosome Endosome Peroxisome Vacuole	Alpha-helix		B
	Beta-barrel (Gram-negative Bacteria or Eukaryote?)	yes	B
		no	C
	none		C
Any other membrane	Alpha-helix		B
	Beta-barrel (Gram-negative Bacteria or Eukaryote?)	yes	B
		no	C
	none		B

4.4 THIRD PARTY TOOLS

Table 13 contains the type of license regarding each software/database used in the tool's development, along with a small description of the task performed by each one and source. No terms or conditions of each license are infringed by *TranSyT*.

Table 13.: Third party tools used in the development of *TranSyT*.

Tool	Task	Type of license	Source
Eclipse photon	Development of the software	Eclipse Public License - v2.0	https://www.eclipse.org/downloads/
TCDB	*TranSyT*'s source database	Creative Commons Attribution -Sharealike 3.0 Unported License and the GNU Free Documentation License	http://www.tcdb.org/
PRED-TMBB2	Prediction of TMB	Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License	http://www.compgen.org/tools/PRED-TMBB2
Phobius	Prediction of TMA	GNU General Public License v3.0	http://phobius.sbc.su.se/
Biosynth	Biosynth database integrates the entries of several databases used by transyt, and provides the necessary hierarchical ontology	GNU General Public License v3.0	https://github.com/Fxe/biosynth-framework
Neo4j	Used to generate *TranSyT*'s internal database	GNU General Public License v3.0	https://neo4j.com/
BLAST+ 2.7.1	*TranSyT*'s homology Search	Freely available to the public for use without any restriction on its use or reproduction	ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/
***merlin* (local-alignments)**	Used to gather and process BLAST+ results	GNU General Public License v2.0	https://github.com/merlin-sysbio/local-alignments
MEWorkbench	Read/Write SBML files and used in validation	GNU Lesser General Public License v2.1	https://github.com/MEWorkbench/biocomponents
LocTree3	Prediction of Compartments	—	https://rostlab.org/owiki/index.php/Loctree
PSORTb 3.0	Prediction of Compartments	—	http://www.psort.org/psortb/
WoLF PSORT	Prediction of Compartments	—	https://wolfpsort.hgc.jp/

4.5 *TranSyT*'s validation

As aforementioned, *i*AF1260 model of *Escherichia coli* was used to validate *TranSyT*'s results. The bacterium model uses BiGG identifiers for metabolites and reaction. However, as *TranSyT* is supposed to be integrated into *merlin* and KBase, it was decided that the validation should be performed using ModelSEED identifiers. Hence, KBase *beta* tools for conversion of identifiers were used. The first step was to use "Import model SBML from web" to upload the SBML model to KBase platform. The second step was use "Integrate Imported Model into KBase Namespace" to convert all identifiers from BiGG to ModelSEED. Finally, the new model was exported in the SBML format to proceed the validation process.

An algorithm employing project MEWorkbench, currently available at `https://github.com/MEWorkbench`, was created to expedite and automate the comparison of *TranSyT*'s results with the model. This package provides methods that allow reading and converting SBML files into container objects. Likewise, these containers can also be instantiated with information returned from *TranSyT* regarding transport reactions. After uploading the SBML into the container, the number of reactions with several compartments, which was the rule for defining transport reactions, was 718. Three different approaches for validation were devised for this work: per gene, per reaction, and per metabolite. Note that the results container only comprises reactions with metabolites available in the model, as all other were discarded by *TranSyT*.

Algorithm 3 Validation process

```
1: Input: sbmlPath
2:        transytResults
3:
4: modelContainer ← readSBML(sbmlPath)
5: resultsContainer ← buildContainer(transytResults)
6:
7: validationByMetabolite(modelContainer, resultsContainer)
8: validationByReaction(modelContainer, resultsContainer)
9: validationByGene(modelContainer, resultsContainer)
```

4.5.1 *Validation per gene*

The validation by gene was performed by confirming if the metabolites transported by the reactions associated to a given gene are the same in the model and in *TranSyT*'s reactions. The metabolites were divided in two different sets: retrieved from the model or from *TranSyT*. The validator determines if the metabolites of each set are present in the other set, and the differences between them. When the process is complete, results are exported to an Excel file.

4.5.2 *Validation per reaction*

The purpose of validating by reaction is to find if reactions are assigned to the same gene both in the model and by *TranSyT*. Again, after instantiating both containers, only transport reactions were retrieved from the model container and added to the *TranSyT*'s results container. When a reaction retrieved from the model was already present in *TranSyT*'s results container, the plugin returns an exception stating that such reaction was a duplicate, and the respective gene association is manually verified. All non-matching reactions were exported to an Excel file and semi-automatically analyzed using functions available at Microsoft Excel®. A pseudo-code regarding this validation process is shown in algorithm 4.

Algorithm 4 Validation process by reaction

```
1: Input: modelContainer
2:        resultsContainer
3: duplicated ← empty set
4: missing ← empty set
5:
6: for each reaction in modelContainer.getAllTransportReactions() do
7:     try
8:         modelCointainer.add(reaction)
9:         missing.add(reaction)
10:    catch (ReactionAlreadyExistsException)
11:        duplicated.add(reaction)
12:    end try
13: end for
14:
15: WriteXLSX(missing)
```

4.5.3 *Validation per metabolite*

The purpose of performing a validation by metabolite was to understand if, independently of the gene encoding the reaction, a metabolite can be still be transported even if using different types of transport systems. As *TranSyT*'s TR identifiers can provide all information regarding the reactants of a transport reaction (irreversibility, compartmentalization, metabolites, and type of transport), a comparison between TR identifiers is adequate to understand if the reactions are identical. Thus, TR identifiers were generated for each transport reaction retrieved from the model, using the same algorithms developed for *TranSyT*. Also, the metabolites location was replaced, as *TranSyT* does not include specific compartments information in its TR identifiers. Hence, the model's compartments (external, periplasm, cytoplasm), were converted to the "in/out" compartments used by *TranSyT* as follows:

- **external to periplasm** $\longrightarrow$ outside to inside
- **periplasm to cytoplasm** $\longrightarrow$ outside to inside
- **periplasm to external** $\longrightarrow$ inside to outside
- **cytoplasm to periplasm** $\longrightarrow$ inside to outside

The TR identifiers created for the model's reactions and the ones retrieved from *TranSyT*'s reactions were exported to an Excel file, in which the function "VLOOKUP" was used to:

- find direct matches between TR identifiers;
- find similar reactions in the same type of transport;
- find metabolites transported by any other type of transport;

A summary of the process is shown in algorithm 5.

Algorithm 5 Validation process by reaction

Input: *modelContainer*
  resultsContainer
modelTransportReactions ← *modelContainer*.getAllTransportReactions()
transytReactionsIdentifiers ← *modelContainer*.getAllTransportReactions().getIDs()

modelTransportReactions ← **correctCompartmentalization(***modelTransportReactions***)**
modelReactionsIdentifiers ← **generateTransytIdentifiers(***modelTransportReactions***)**

WriteXLSX(*transytReactionsIdentifiers*, *modelReactionsIdentifiers*)

5

RESULTS AND DISCUSSION

5.1 ASSESSMENT OF AVAILABLE METHODS TO PREDICT TRANSMEMBRANE β-BARRELS

Since TMB-KNN and partiFold webservers were not working (errors were thrown after submitting the sequences) and the e-mail with the results of TMPpro was never received, these three tools were excluded from the analysis. The remaining five tools correctly predicted the existence of 10 β-barrels in the 10 sequences regarding *Homo sapiens* and *Escherichia coli*, but none predicted correctly the only β-barrel available for *Arabidopsis thaliana*, the only false negative for all tools. In BetAware's predictions, one of the sequences (α-helical) failed every prediction attempt and the only answer provided by the webserver, after a while, was fail. BOMP was the only predictor with a false positive in its results, predicting incorrectly the β-stranded *Homo sapiens* protein P20023 (UniProt accession) as β-barrel. As shown in table 5, BOMP, one of the first tools ever developed for the prediction of β-barrels, was specifically designed to predict β-barrels in the outer membrane of gram-negative bacteria, whereas this false positive was predicted for an eukaryotic organism. PRED-TMBB2, BOCTOPUS2 and MCMBB, correctly predicted the absence of β-barrels in the remaining sequences, while being able to identify the difference between β-stranded sequences with no β-barrels from β-stranded sequences with β-barrels. The first two, considered the state-of-the-art tools, predicted the number of β-strands for each β-barrel, with minor differences between them (see column "#TM" in table 14). Although providing similar results, at the time of the comparison, PRED-TMBB2 and MCMBB had a limit of 5 000 and 1 000 sequence for each submission, respectively, while BOCTOPUS2 only allowed 5. Regarding running times, once again BOCTUPUS2 is worse, taking hours for each submission, especially for sequences with no β-barrels. As the accuracy of the predictions for PRED-TMBB2 and MCMBB is identical and the running times are similar, a few minutes for this test dataset, the only difference is the output format.

Despite the fact that BOMP has not been developed for predictions in eukaryotes, the false positive in the results is an indication that its performance is impaired, as most of the other tools were also developed for gram-negative bacteria and yet their predictions were correct. BetAware has also lower performance when compared with PRED-TMBB2,

BOCTOPUS2, and MCMBB because it failed one of the predictions. In conclusion, PRED-TMBB2, BOCTOPUS2, and MCMBB have the same accuracy but not the same efficiency. For example, the running time and the number of sequences per submission are the major problems found for BOCTOPUS2, making this server almost useless for a large dataset. Regarding the MCMBB and the PRED-TMBB2, the first returns the probability of a β-barrels being present, while the second returns a wide range of information for each sequence, such as OMPdb family, number of β-strands and reliability of each prediction, making it the most complete between all the tools analyzed. It is also important to note that PRED-TMBB2 prediction methods are based on Hidden Markov Models which, which according to Bagos et al. 2005, are the most reliable predictors for β-barrels (9).

The study regarding the tools for TMBs prediction selected PRED-TMBB2 as the most reliable for *TranSyT*'s classifications due its overall performance when compared with other tools.

Table 14.: Results of each tool with the only false positive highlighted in yellow and the only error in red.

Entry	Length	Organism	Topology	PRED-TMBB2		BOCTOPUS2		BetAware	BOMP	MCMBB
				Prediction	#TM	Prediction	#TM			
1BXW_A	175	*Escherichia coli*	β-barrel	yes	8	yes	8	yes	yes	yes
1TLY_B	282	*Escherichia coli*	β-barrel	yes	12	yes	12	yes	yes	yes
2BRJ_C	191	*Arabidopsis thaliana*	β-barrel	no	0	no	0	no	no	no
2JK4_A	299	*Homo sapiens*	β-barrel	yes	14	yes	16	yes	yes	yes
2K4T_A	296	*Homo sapiens*	β-barrel	yes	14	yes	14	yes	yes	yes
3GP6_A	166	*Escherichia coli*	β-barrel	yes	8	yes	8	yes	yes	yes
4DCB	301	*Homo sapiens*	β-barrel	yes	10	yes	10	yes	yes	yes
4NTJ_A	473	*Homo sapiens*	α-helical	no	0	no	0	no	no	no
4O6Y_A	234	*Arabidopsis thaliana*	α-helical	no	0	no	0	no	no	no
4R9U_B	338	*Escherichia coli*	α-helical	no	0	no	0	no	no	no
4X5M_A	102	*Escherichia coli*	α-helical	no	0	no	0	no	no	no
5GLI_A	471	*Homo sapiens*	α-helical	no	0	no	0	no	no	no
5JDP_A	290	*Homo sapiens*	β-barrel	yes	14	yes	12	yes	yes	yes
5NIK_K	664	*Escherichia coli*	α-helical	no	0	no	0	FAIL	no	no
5TZY_A	499	*Homo sapiens*	α-helical	no	2	no	0	no	no	no
5XDO_A	300	*Homo sapiens*	β-barrel	yes	16	yes	14	yes	yes	yes
5XPD_A	298	*Arabidopsis thaliana*	α-helical	no	0	no	0	no	no	no
P00441	157	*Homo sapiens*	β-stranded	no	0	no	0	no	no	no
P0A910	352	*Escherichia coli*	β-barrel	yes	8	yes	8	yes	yes	yes
P0A917	174	*Escherichia coli*	β-barrel	yes	6	yes	8	yes	yes	yes
P0ACF8	140	*Escherichia coli*	β-stranded	no	0	no	0	no	no	no
P0ACJ8	214	*Escherichia coli*	β-stranded	no	0	no	0	no	no	no
P20023	1051	*Homo sapiens*	β-stranded	no	0	no	0	no	yes	no
Q39255	163	*Arabidopsis thaliana*	β-stranded	no	0	no	0	no	no	no
Q9SMT7	522	*Arabidopsis thaliana*	β-stranded	no	0	no	0	no	no	no

5.2 VALIDATION OF TCDB'S INFORMATION

While performing the comparison, all words from both databases were sought in the dictionary. When a match was found, the keyword for the match replaced the sought word. The result of this assessment was organized in 12 different classifications, as shown in table 15.

Table 15.: Comparison results and description of the respective classifications. Data retrieved on April 2018.

Classification	Description	Frequency
Same	The same metabolites in both databases.	2 643
Different	Metabolites in both databases are completely different.	1 815
Unknown TCDB	Internal database has results whilst TCDB has "unknown".	569
Unknown Internal database	TCDB has results whilst the Internal database has "unknown".	357
Unknown Both	Both databases have "unknown" as metabolites.	559
Subset TCDB	All TCDB metabolites are present in the wider set of metabolites described by the internal database.	340
Subset Internal database	All IntenalDB metabolites are present in the wider set of metabolites described by TCDB.	331
None TCDB	Cases where TCDB has "none" as transported metabolites.	305
Proton TCDB	The only difference is an extra H+ in TCDB's metabolites.	3
Proton Internal database	The only difference is an extra H+ in internal database's metabolites.	481
None Internal database	No metabolites available in the internal database	158
Not annotated TCDB	Entries without "substrates" in TCDB.	88
TOTAL		7 649

Although the number of entries in TCDB was above 16 000 at the time of the study, only 7 610 entries characterized in the internal database were selected for comparison. Maintaining *TRIAGE*'s manually curated internal database up to date is very demanding and requires numerous person hours; thus, an automated approach was required. This short-

coming led to the development of *TranSyT*'s automatically updatable database, described in this work.

The results described in table 15, show that approximately one third of the metabolites represented in both databases are identical. Also, the "Unknown - Internal database" and "None - Internal database" classifications assume TCDB annotation as correct, as it is not possible to infer metabolites from "unknown" or "none". "Subset - TCDB" is also assumed as correct, as all metabolites described by TCDB are present in the internal database set. "Proton - TCDB" and "Proton - internal database" can also be assumed as correct as the only difference between the entries of both databases is an extra H+. Usually, the presence of a proton in the "substrates" field at TCDB, is associated to symport or antiport mechanisms. However, this metabolite is already described in transporter families general transport reactions. Thus, the comparison can be performed without it. Assuming all these cases as correct, approximately 48% of the classifications remain as "incorrect". The entries of these groups were reviewed manually.

Regarding the classification "Different", it was shown that a major portion of the differences were due to hierarchical differences. For instance, TCDB is often more generic, describing that a sugar is transported instead of sucrose, while the opposite happens with the internal database. This problem will be resolved by *TranSyT* through the search for metabolites hierarchical "descendants". Other major difference was ions being transported along with other substrates, with either one of the databases not including these ions. These situations were ignored and TCDB was considered correct as when a ion is mandatory it is included in the generic transport reaction (as for protons), and also because ions are not involved in metabolic reactions. The same methodology is used for the classification "Subset - Internal database".

Of the total entries, 569 of them were classified as "Unknown TCDB", as only TCDB was missing the metabolites to transport. Likewise, the internal database had 357 entries with unknown metabolites. Both databases described the metabolite associated to the same entries as "unknown" in 559 cases. Metabolites classified as "none" occurred 305 times in TCDB and 158 in the Internal database, with only one of them in common.

Therefore, TCDB was selected as source for the metabolites annotation because: (1) 52% of the information was the same as the annotated in the internal database records; (2) among the 48% "different" data there was no significant differences; (3) TCDB will continuously keep updating the database information, adding new entries to the existing ones. This allows *TranSyT* to keep retrieving data generated from TCDB without the need to curate it. Nevertheless, future errors found at TCDB's substrates annotations can be overcome, by adding exception to *TranSyT*'s annotation workflow.

The file "Validation_TCDB.xlsx" containing the complete data of the analysis performed to assess TCDB information is available at https://nextcloud.bio.di.uminho.pt/s/CKb3JNfCyGFrqy9.

5.3 *merlin*'s improvements

5.3.1 *Compartments parsing and integration*

merlin's new "Compartments annotation panel", including the window that opens when the integration process is launched, is shown in figure 16.

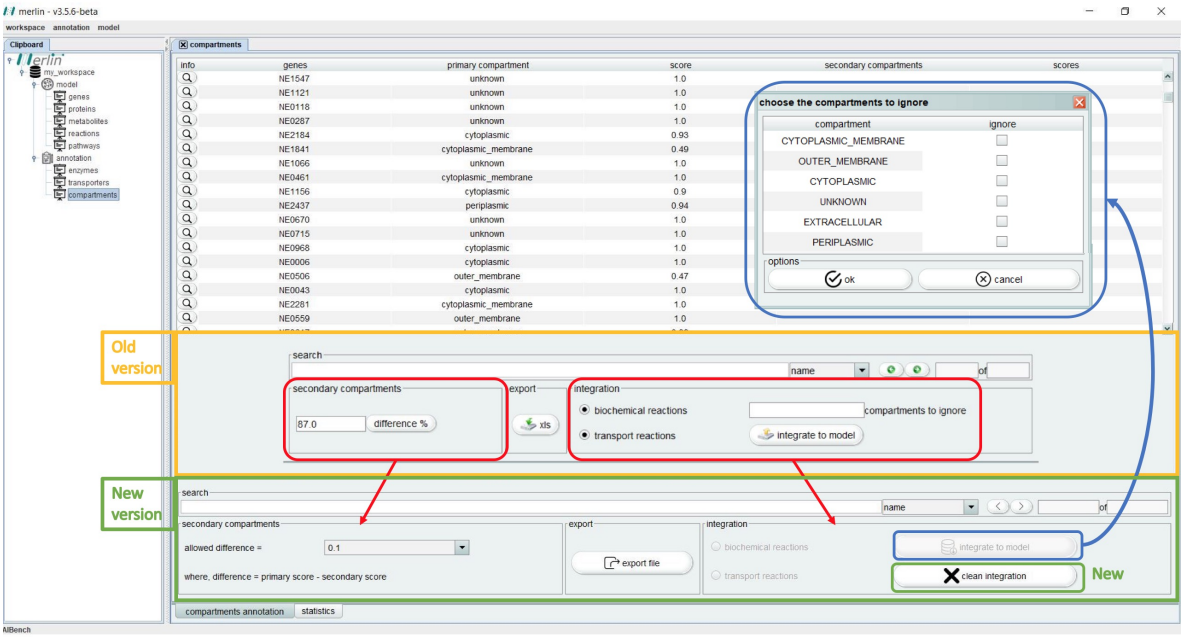


Figure 16.: Representation of the differences between the old and new "Compartments annotation panel". Green: new feature to clean all integrated data or just the selected data; Red: The before (inside the yellow box) and after of some options. Instead of typing values directly in the text boxes, the user can select between options; Blue: After clicking "integrate to model", a new window opens, asking which, if any, compartments the user wants to be ignored.

Replacing the text boxes where the user would have to type text directly, by methods that make available options to select, errors generated by bad input from the user are eliminated. This option was also taken into account while developing *TranSyT*.

A new feature added to the panel is the "clean integration" option. If errors occur during the integration process, it is possible to revert the model comparmentalization. This process, can be individually performed to "biochemical reactions" and/or "transport reactions". A similar option was also developed for the "Transporters annotation panel", which

allows cleaning transporters integration. Before these implementations, integrations were irreversible.

5.3.2 *Annotation workflow*

Regarding the automatic annotation of enzymes, the user will implement the annotation pipeline in the window shown in figure 17. The slots are numbered from 1 to 9 and the partition of workflow in 4 different fields is also obvious.

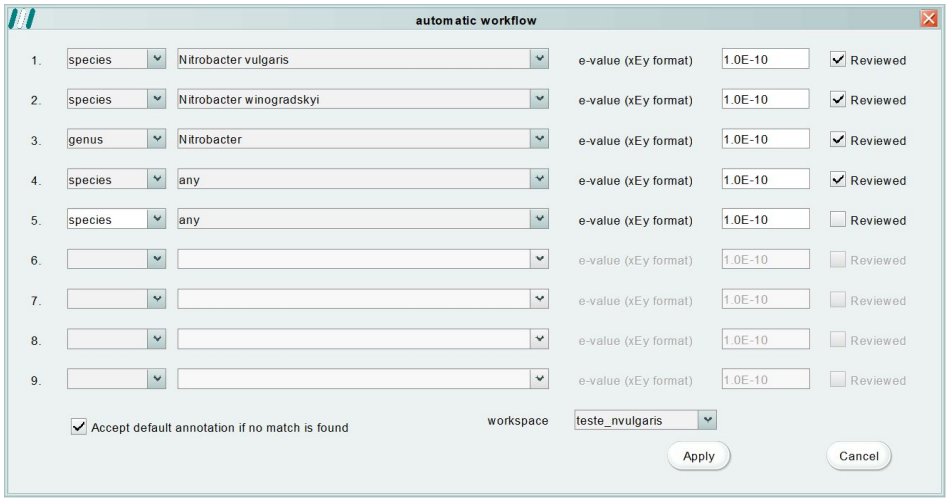


Figure 17.: Representation of an hypotethical workflow for species *Nitrobacter vulgaris*.

When the annotation process is complete, the "Enzymes annotation panel" is refreshed and the confidence level of the annotations is shown in the column "notes", as demonstrated in figure 18.

The improvements made in *merlin* allowed achieving a more user-friendly GUI for compartments integration, which allows to avoid errors from the user. Also the enzymes annotation process was successfully improved, due the achievement of a fully automatic process for the assignment of functions to genes.

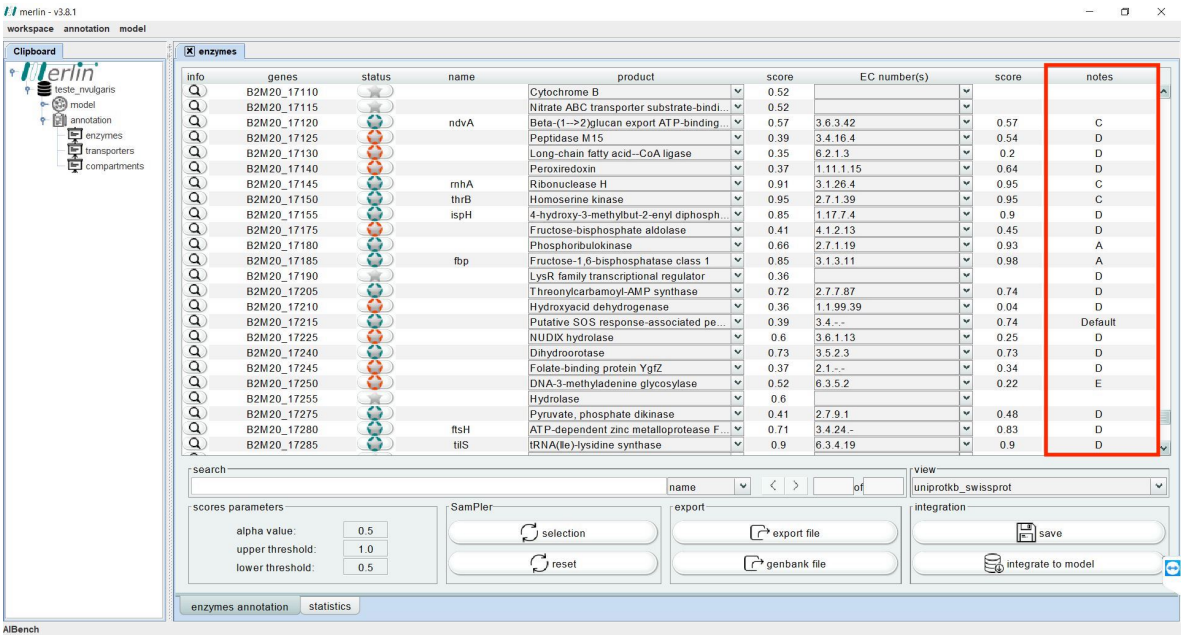


Figure 18.: Results of automatic annotation using workflow represented in figure 17.

5.4 TRIAGE *versus TranSyT*

The main goal of this project is the development of a tool that performs the same tasks as *TRIAGE*, using more efficient methods and overcoming *TRIAGE*'s major restrictions, namely: TMDs identification;

- *TRIAGE*'s rigorous rules in TCGs identification;
- slow running time;
- and inability to automatically follow TCDB's growth.

Figure 19 represents each iteration of the proteins identification process in *TRIAGE*. The total number of genes in the genome is gradually reduced, based on one method at each iteration. Genes predicted to encode TMAs are selected to be aligned against TCDB records, using Smith-Waterman (128) algorithm running in the users' computer. This approach does not take into account proteins encoding TMBs, as no method was used to perform such identification. *TranSyT* addresses this problem using a different approach. Instead of iteratively reducing the number of genes at each stage, it combines the information retrieved from several sources to assign confidence levels. In *TranSyT*, unlike *TRIAGE*, genes can be considered transport systems even if no TMD are predicted on the sequence. Likewise, *TranSyT* does not require the gene to have a location prediction on a membrane. *TranSyT* assesses the homology against TCDB records and these alignments allow annotating genes with TC families.

The TMD and location predictions allow assigning a confidence level to the identification of transporter system. This feature is only available for *merlin*'s version, which uses third-party tools to provide these predictions. KBase does not have tools to perform these predictions and does not rely on third-party tools.

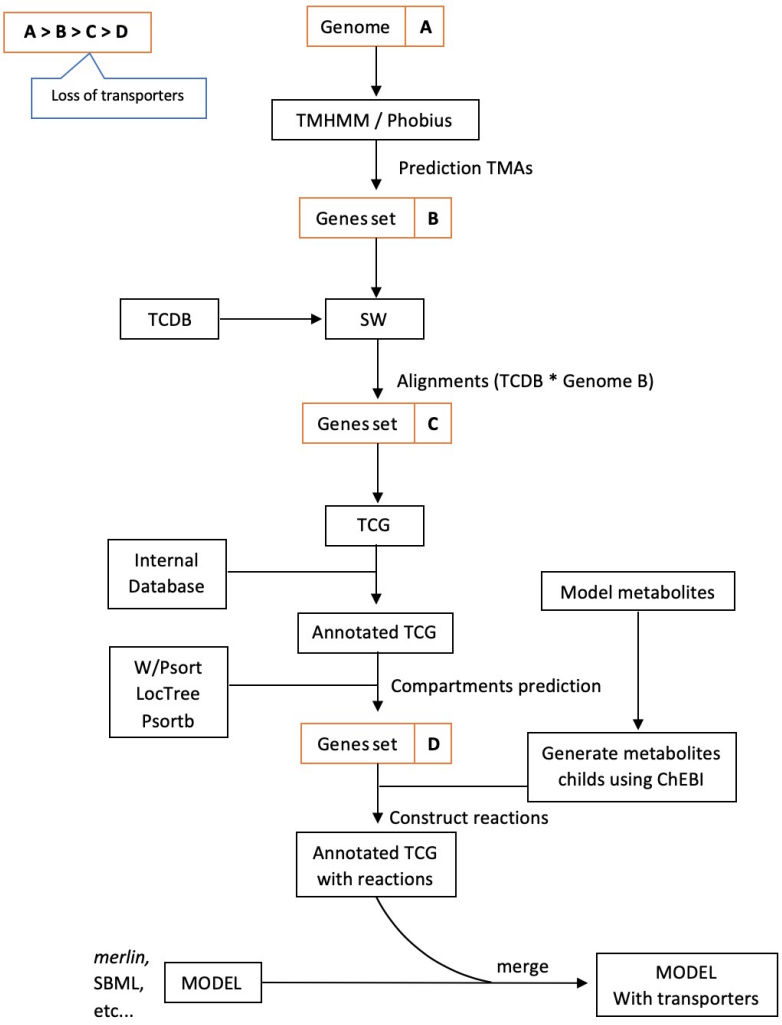


Figure 19.: *TRIAGE* architecture.

The homology search performed by *TranSyT* uses BLAST+ to perform the alignments and several advantages resulted from this decision. The alignments can be performed remotely, which expedites the process, and unlike *TRIAGE*, are performed to the entire genome. Using a machine running Windows 1064-bit Operative System, processor Intel(R) Core(TM) i7 first generation CPU 1.60 GHz, and 8 GB of RAM, alignments can be performed in average in 5 minutes for the genome of *E. coli*, containing 4 140 genes. The same does not happen with *TRIAGE*'s Smith-Waterman algorithm, in which an alignment of a batch of about 500 genes can take several hours.

Another major difference is the fact that *TranSyT* generates all transport reactions beforehand storing them in its internal database. Whereas *TRIAGE*, generates the reactions for every descendant metabolite after the identification of TCGs, increasing the time for annotating the transporters. *TRIAGE* uses its manually curated database to generate transport reactions, currently counting with 7 383 entries (42% of total TCDB entries). Efforts to manually assemble this database are time-consuming and an endeavor for which several researchers have contributed. Despite these efforts, *TRIAGE* still has less than half of the total entries described in TCDB. *TranSyT* automatically retrieves information from TCDB and generates the suitable transport reaction for each metabolite described, storing the information in its internal database, making it available per request. This allows *TranSyT*to always be up-to-date with TCDB's constantly increasing information.

When the process is complete, the output format of both tools is the same. However, *TranSyT* is a standalone software that can be integrated into other tools (currently *merlin* and KBase) or run independently, unlike *TRIAGE* that was built inside *merlin*'s framework. Figure 19 represents the entire process of transporters identification and generation of reactions to be executed by *TRIAGE*. The same process using a different approach is represented in 20, that corresponds to *TranSyT*'s module for identification of genes encoding transporter systems.

TranSyT's architecture allows it to overcome *TRIAGE*'s limitation regarding TMB's identification and running time. Also the internal database can be regularly updated, which is not feasible for *TRIAGE*.

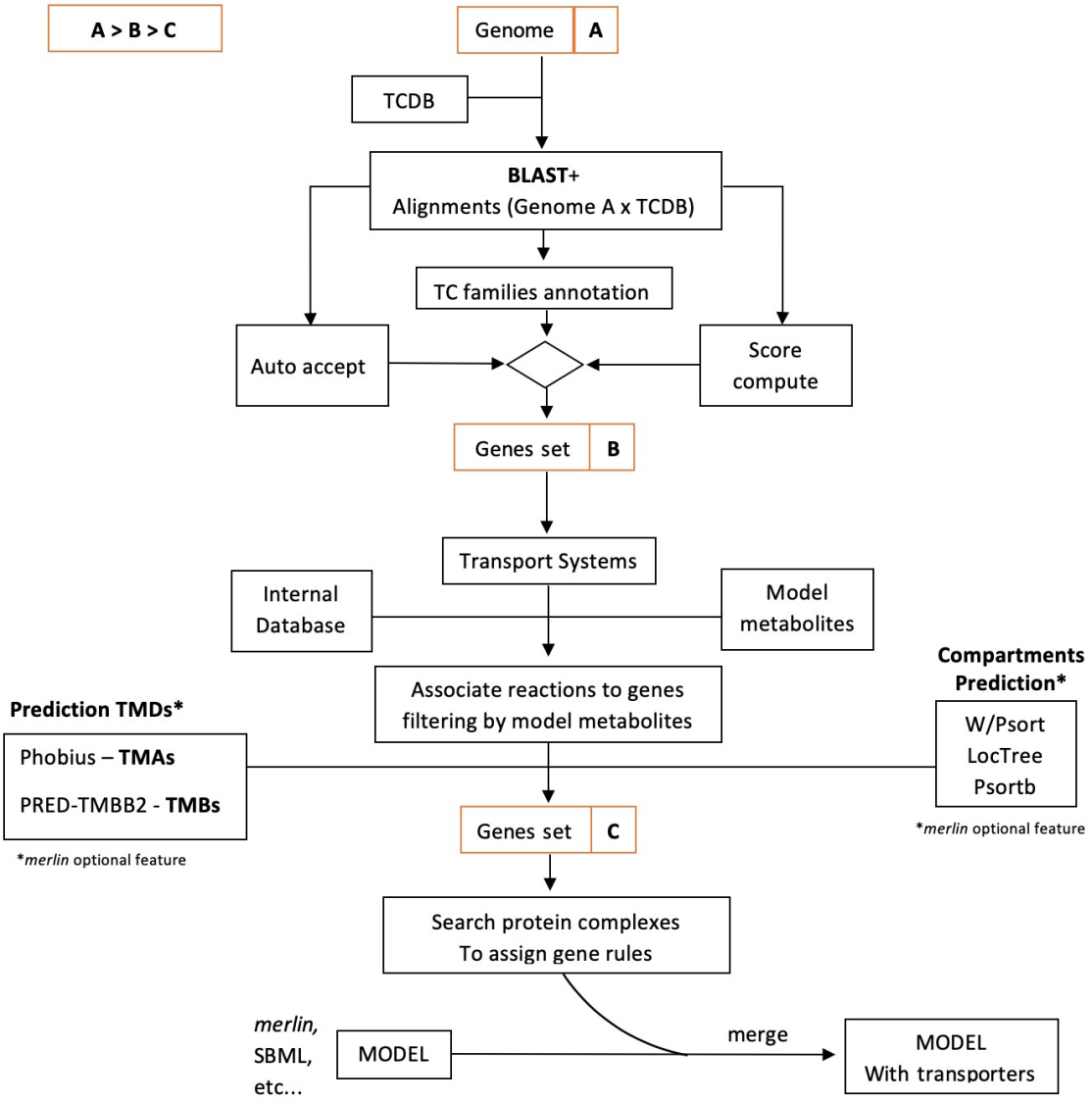


Figure 20.: *TranSyT*'s module for identification of genes encoding transporter systems.

5.5 *TranSyT*'s user interface

With the objective of creating a standalone friendly-user interface, *TranSyT*'s GUI was developed. When the user runs the software, the initial GUI opens and allows the user to input the resources required for the classification of the transporters, and to select the optional features. This GUI is shown in figure 21.

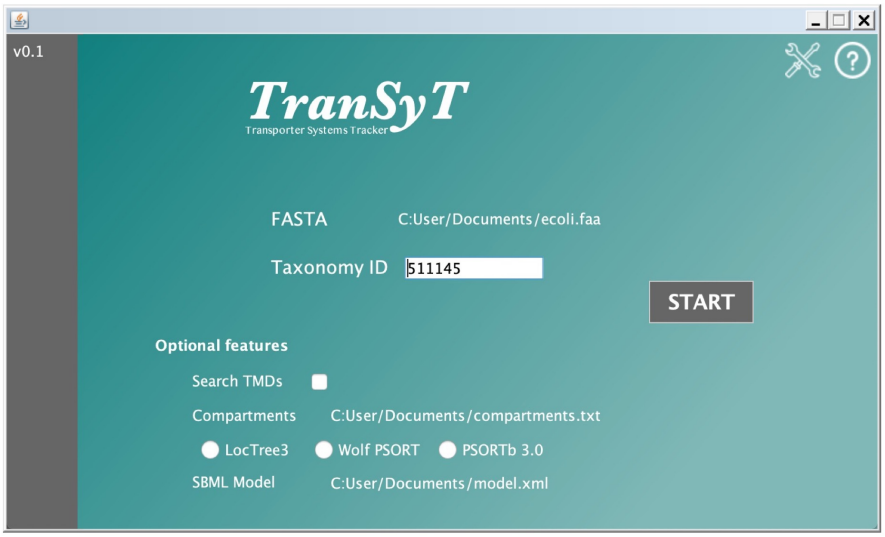


Figure 21.: *TranSyT*'s initial GUI.

TranSyT's parameters configuration is available trough this user interface. Here, users can open a new GUI (figure 22) that allows modifying the set of parameters stored in *TranSyT*'s configuration file, namely:

- BLAST parameters E-value, bit score, and query coverage;
- α of equation 2 for TC families annotation;
- E-value for automatic acceptance of reactions;
- α, β, threshold, and minimum hits of equation 2 for score computing; default database of the identifiers;

After clicking the "start" button in the initial GUI, the results are generated and displayed in the results GUI (figure 23). A small summary of the process is displayed, providing information about the total reactions generated, total metabolites participating in the reactions, and total different TC families annotated. The total number of different transport types generated is shown. The results of the software can then be finnally exported in .csv or SBML formats. Figure 23 illustrates an example of what would be the outcome of the *i*AF1260 model used in the validation process.

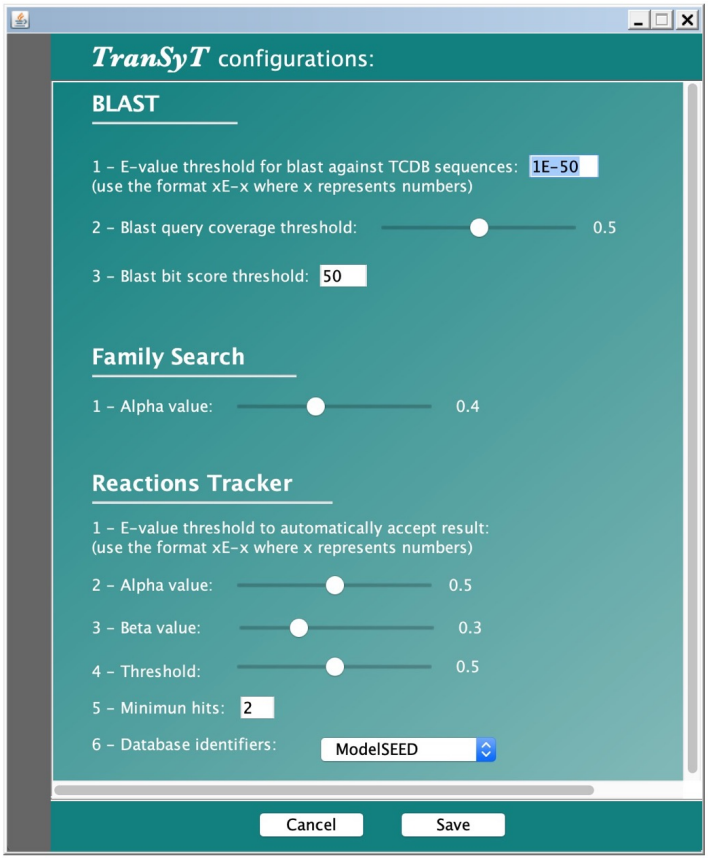


Figure 22.: *TranSyT*'s configurations GUI.

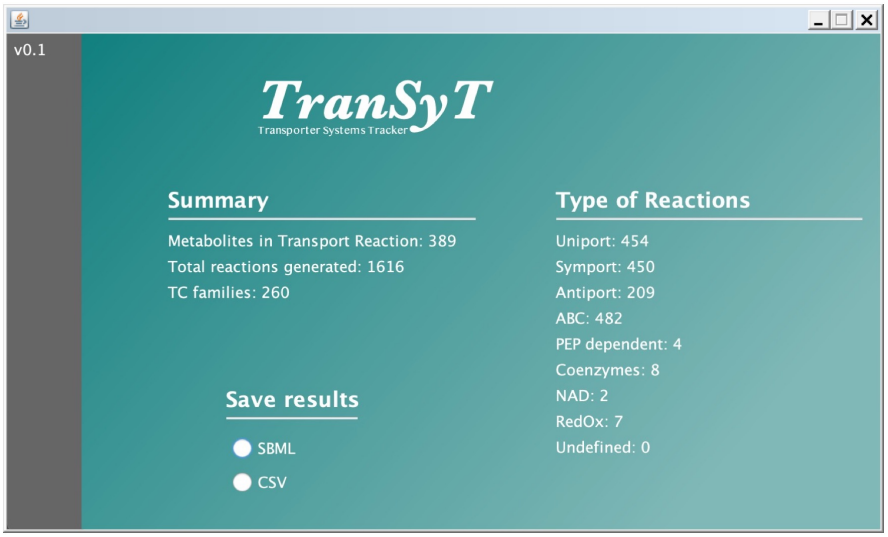


Figure 23.: *TranSyT*'s results GUI.

5.6 *TranSyT*'s internal database

Neo4j was used to develop *TranSyT*'s graph database using the Neo4j Java Driver. This database currently contains information regarding 17 321 transporter systems available in TCDB. Table 16 illustrates the contents stored in the database. Of the total 6 015 metabolites available, 5 002 were associated using hierarquical ontology search. A total of 58 623 reactions were generated for 1 176 families.

Table 16.: Summary of *TranSyT*'s internal database current contents.

Property	Counts
TC systems (TC Number, Accession)	17 321
Accessions	17 221
TC Numbers	13 349
TC Families	1 176
Family-specific reactions	428
Metabolites	6 015
Descendent metabolites	5 002
Reactions	58 623

5.7 *TranSyT*'s validation

5.7.1 *Validation per gene*

The *i*AF1260 *E. coli* model has 406 genes encoding 660 transport reactions and 65 reactions without gene associations. Whereas *TranSyT*, provided 644 genes with significant homology evidence (for an E-value of 1E-50) to transport systems listed in TCDB, and no reactions without encoding genes. Yet, 11% of the genes present in the model were not available in *TranSyT*'s results. These genes include cases in which no homologies with TCDB records were reported, and cases in which homology was reported but no reactions could be generated, due the lack of elements in TCDB that allow generating reliable transport reactions (for instance, no metabolites described for a transport system). A gene with locus tag "s0001" was found in the model, encoding 13 transport reactions with 12 participating metabolites. However, this gene cannot be traced to *E. coli* when sought in NCBI databases, but to *Shigella flexneri* 5a str. M90T, a different organism. As the protein sequence encoded by this gene is not present in *E. coli* genome, a similar result is not possible. When searched in TCDB using BLAST, this protein also had low similarity with the records present in the database.

The model describes 411 metabolites (including several ions) as participating in transport reactions. Whereas *TranSyT* created transport reactions for 389 metabolites. Almost 35% (143) of the metabolites present in the model's reactions are not available in *TranSyT*'s reactions set. Likewise, *TranSyT* also generated reactions for 121 metabolites not available in the model transport reactions set. Analysis per gene, determined that for 59 genes, the set of metabolites participating in transport reactions was exactly the same for the model and for *TranSyT*. While the set of metabolites, participating in transport reactions, from *TranSyT* was wider than the one from the model for 147 genes. Hence, almost 52% of the model genes encoding transport proteins are associated with transport reactions involving metabolites that *TranSyT* automatically associated with the same genes.

On the other hand, cases in which *TranSyT* associates metabolites to genes, where the model associates to a larger set of metabolites (though including *TranSyT*'s set) are also available. Most of these cases are related with genes associated to reactions that carry metabolites not available in TCDB for those entries. Thus, *TranSyT* will not generate reactions with those metabolites.

The average number of metabolites participating in transport reactions is 8 and 10 for the model and *TranSyT*, respectively. These values are not much different and the maximum deviation from this value is 66 for gene b0809 in *TranSyT* due high similarity with 36 systems of the same family (3.*A*.1), and the generation of reactions for every descendant of the metabolites of those homologies. However, in the model, the maximum deviation is 247, a

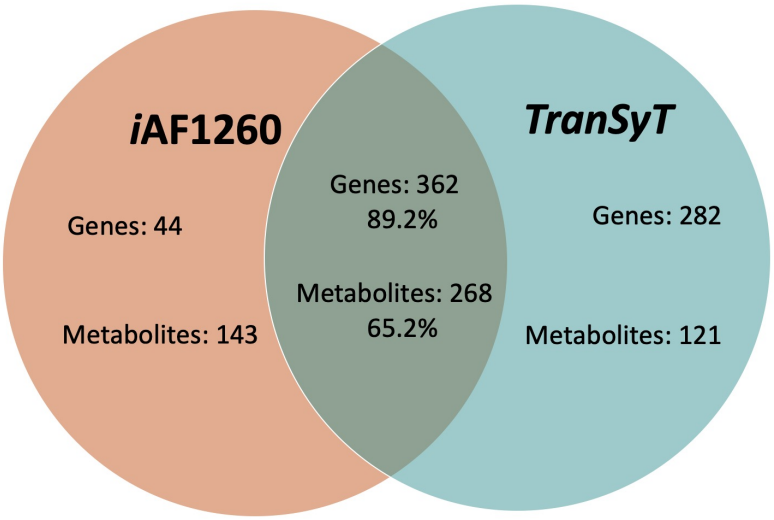


Figure 24.: Number of genes and metabolites in common between the *i*AF1260 model and *TranSyT*.

situation that happens for 4 different genes, b0929, b1377, b2215, and b0241. For these genes, a wide range of metabolites is claimed to participate in the transport reactions associated with such genes. However, several of the metabolites are not related in MetaCyc hierarchical ontologies, although, ions, sugars, nucleotides, amino acids, and several other classes of metabolites are associated with these genes in the model. In TCDB, the annotation for such transport system is a "porin transporting non-specific small molecules", belonging to class 1.*B*.1. However, b0241 is annotated as phosphoporin transporting exclusively phosphate. Another case of wider subset of metabolites per gene in the model is gene b0837. Here, the gene has 5 metabolites in transport reactions. However, the gene has no similarities with entries from TCDB, nor was annotated as a transport system by *TranSyT*. Thus, the model annotation is likely incorrect as, according to TCDB, the proteins encoded by these genes cannot transport such myriad of compounds.

5.7.2 *Validation per reaction*

The algorithm 4 used in the Validation per reaction allowed finding 175 perfect matches with model reactions. The goal was determining if reactions encoded by a gene in the model, are also the same in *TranSyT* for such gene. The automatic matching process finds direct matches, thus the remaining were confirmed manually, to ensure that the reasons of the mismatch were understood. Thus, using this strategy, information from UniProt, ModelSEED, MetaCyc, and TCDB, was assessed to provide a correct classification for each case.

In 91 cases, the reactions present in *TranSyT* were similar to the ones described in the model. The metabolites being transported were the same. However, the type of transport might not be the same. For instance, the model represents a Symport for a metabolite *M*, but TCDB's TC family reaction describes the transport of *M* as Uniport or Antiport. Identical reactions with distinct reversibility were classified as similar. *TranSyT* generates its reactions based in evidences of the TC family reaction equation and in the description of the system, subfamily, family, and superfamily. Cases where no evidences were found, simple transport, symport with proton or antiport with proton was generated, which can explain part of the similar cases.

In 20 cases, reactions from the model were different from *TranSyT*. Cases like this take place when the gene has no homologies with TCDB entries (example aforementioned b0837) or when the type of transport is completely different. For instance, the model represents reactions associated with genes b0731, b0697, b1621, b2429, b2167, and b2169 as phosphoenolpyruvate (PEP) dependent. Nevertheless, TCDB does not contain any TC family reaction for family 4.*A*.2 to which these genes belong in such database. Thus, *TranSyT* will not generate a similar reaction for those genes. The same can happen with ABC transporters or RedOx dependent reactions for example.

For 54 reactions, it was not possible to find the metabolite transported in the model's reactions within *TranSyT*'s output, although the metabolite belonged to the class of the metabolites described in TCDB for such entry because such relationship was not available in MetaCyc's ontology. This issue is associated to the lack of metabolites in MetaCyc's hierarchy for the described class of metabolites, or to the missassignment of identifiers when searching TCDB metabolites in Biosynth. Several reactions (65) were not associated to genes, and could not be traced in *TranSyT*. In 295 cases, the metabolites (or the class to which those metabolites belong) described in the model's transport reactions, were not available in the TCDB entries identified as homologous of the genes associated with such reactions, thus *TranSyT* will not generate such reactions. "Undefined" cases (4) are associated to transport reactions in the model for metabolites whose formula is not available in ModelSEED. Hence, reactions were generated by *TranSyT* for comparison. Finally, cases with different types of reactions, having clear indications in TCDB of the TC family reaction and transported metabolites, but with different representation in the model were classified as "Wrong in model". Cases were all genes encoding the reactions had no similarities with TCDB were classified as such. This classification was attributed 14 times. Table 17 summarizes the counts of all different classifications.

As for almost half the entries the metabolites were not described in TCDB, it is difficult to assess the performance of *TranSyT* in this validation. The absence of these metabolites in TCDB's *E. coli* records can be associated with missing annotations in TCDB or incorrect gene-protein-reaction rules in the model. In the former case *TranSyT*'s capabilities of gener-

Table 17.: Results of Validation per reaction.

Classification	**Counts**
Perfect match	175
Similar	91
Different	20
Wrong in model	14
Metabolite missing for gene(s) homology in MetaCyc hierarchy	54
Metabolite not available in homologous TCDB entries	295
No gene in model	65
Undefined	4
TOTAL	718

ating transport reactions for such systems is impaired. Whereas the latter case is associated with errors in the model. Nevertheless, *TranSyT*'s predictions should be validated with wet-lab experiments to assess robustness.

5.7.3 *Validation per metabolite*

Validation per metabolite was performed to determine whether metabolites transported in the model participate in any of the reactions generated by *TranSyT*. In this approach, TR identifiers were generated for all reactions in the model and then compared to the ones generated by *TranSyT*, regarding:

1) perfect match;

2) a match in the same type of transport without compartmentalization;

3) a match in the same type of transport without compartmentalization and without reversibility;

4) in any other type of transport.

Compartmentalization was excluded because of reactions with the same characteristics but with reversed compartments. However, reversibility is important; thus, it was excluded just in the next level of classification of the aforementioned workflow.

Using this approach, it was possible to identify 44% (316) perfect matches. The exclusion of compartments allowed to match almost 2% more metabolites; however, disregarding also

the reversibility allowed identifying 9% more matches in the same transport system, which is more visible in symport and PEP-dependent reactions. The next step was to exclude the type of transport and comparing the metabolites participating in the transport. This enabled identifying nearly 20% more matches. Nevertheless, 25% of the metabolites of the model do not participate in any of the reactions generated by *TranSyT*. A possible justification seems to be associated with the fact that several of these metabolites were classified in the validation per reaction as "Metabolite not available in homologous TCDB entries".

Table 18.: Summary of the counts for validation per metabolite.

	Mechanism					
	Same	**Same (no Compartments)**	**Same (no Compartments no Reversibility)**	**Other**	**No match**	**TOTAL**
ABC	12.1%(87)	0%(0)	0%(0)	1.1%(8)	4.5%(32)	17.7%(127)
Symport	8.4%(60)	0.1%(1)	4.7%(34)	5.6%(40)	2.5%(18)	21,3%(153)
Uniport	21.2%(152)	1.1%(8)	2.2%(16)	9.2%(66)	14.3%(103)	48.1%(345)
Antiport	1.8% (13)	0%(0)	1.5%(11)	1.8%(13)	3.6%(26)	8.8%(63)
PEP	0%(0)	0%(0)	0.7%(5)	1.5%(11)	0%(0)	2.2%(16)
CoA	0.6%(4)	0.4%(4)	0%(0)	0.1%(1)	0.3%(2)	1.4%(10)
NAD	0%(0)	0%(0)	0%(0)	0.4%(3)	0.1%(1)	0.6%(4)
TOTAL	44%(316)	1.7%(12)	9.2%(66)	19.8%(142)	25.4%(182)	100%(718)

TranSyT was able to identify transport reactions for nearly 75% of the metabolites transported in the model. All files containing the validation data of the three methods of validation used are available in separate Excel documents https://nextcloud.bio.di.uminho.pt/s/CKb3JNfCyGFrqy9.

5.7.4 *GPRs validation*

As observed during the validation per gene stage, *TranSyT* is able to associate reactions to genes. Nevertheless, using the method described in Algorithm 3, *TranSyT* is also able to find protein complexes composed by several subunits. Using this method, 293 complex systems were found and associated to 454 different transport reactions. *TranSyT* was able to successfully create GPR associations for systems as long as 13 subunits, such as:

- 3.*D*.1.1.1 - b2276 and b2277 and b2278 and b2279 and b2280 and b2281 and b2282 and b2283 and b2284 and b2285 and b2286 and b2287 and b2288.

The complete model contains 88 complexes associated to 133 different reactions. However, to validate *TranSyT*'s GPR rules, the 316 perfect matches of the validation per metabolite were compared to the model rules. Regarding these reactions, *TranSyT* found 95 different complexes for 86 reactions, whereas the model described 40 different complexes for only 60 reactions. *TranSyT*'s rules were in agreement 52 times. When not in agreement, minor differences were found 3 times, with only one subunit being different. The remaing 5 different rules were completely dissimilar, with *TranSyT* creating rules with a greater number of subunits than the ones in the model. *TranSyT* was able to create gene rules for 21 reactions assigned with just one gene, and 5 reactions without any gene rule, in the model. A graphical representation of the comparison is illustrated in figure 25.

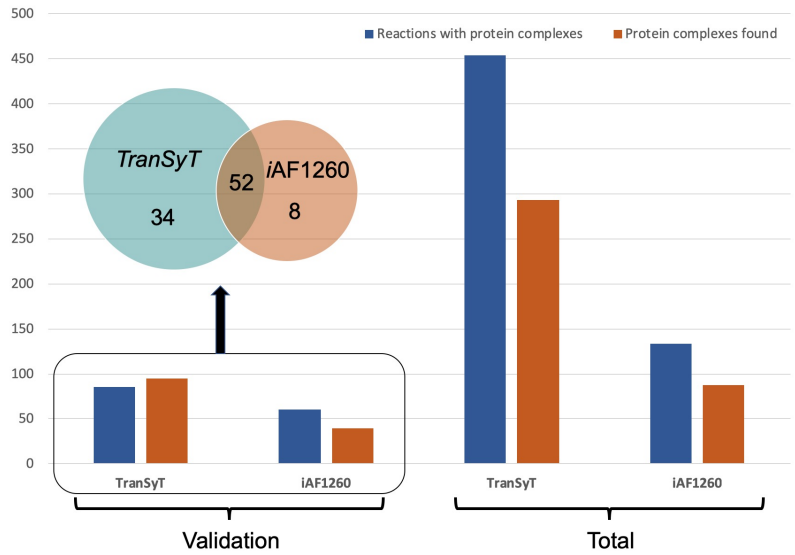


Figure 25.: Comparison between the number of protein complexes found in *i*AF1260 and *TranSyT*.

The validation of *TranSyT*'s classifications using the *E. coli* model *i*AF1260, shows that *TranSyT* assigns transport reactions to 89% of the genes described as transporters by the model, and generated transport reactions for almost 75% of the metabolites participating in transport reactions in the model. Unlike the model, *TranSyT* assigned gene rules to all reactions generated, including complexes formed by multiple genes when a reactions is encoded by several subunits. All GPRs generated and data used in the validation process are available https://nextcloud.bio.di.uminho.pt/s/CKb3JNfCyGFrqy9.

6

CONCLUSION

6.1 CONCLUSIONS

The work presented in this thesis aimed to develop a software that could identify, classify, generate transport reactions, and annotate the genome of any organism. *TranSyT* is the outcome of that goal, the next iteration of *TRIAGE*, set to overcome its limitations. Unlike *TRIAGE* whose internal database was updated once every year through the effort of several researchers, *TranSyT* is automatically updatable, having the ability to scrape TCDB more frequently to follow its growth and changes. *TranSyT*'s architecture allows it to be fast, generating results in a few minutes for a bacterial genome of the size of *E. coli*.

A process for the identification of transmembrane β-barrels was developed and implemented in *TranSyT* to determine the reliability of the annotation, along with the predictions of α-helices and compartments. This feature will only be available in *merlin*'s integrated version and in *TranSyT*'s standalone version. To date, besides *TRIAGE* and *TranSyT*, no other software is known to perform the classification of transporter systems while generating the respective transport reactions.

The validation of the software shows that *TranSyT* was able to successfully generate reactions for nearly 75% of the metabolites transported in the *i*AF1260 model, matching the metabolites involved in the reactions for 52% of the genes. Moreover, it allowed identifying genes encoding transport reaction without homologies with any of the records of TCDB, and genes that encode reactions not able of carry metabolites as described in the model. Almost 90% of the genes encoding transport reactions in the model were also classified as transporters by the software, and *TranSyT* allowed to find a gene not related to the organism in the model and reactions not encoded by any gene. *TranSyT*'s capabilities of generating GPRs allowed assigning 293 complexes to 454 reactions, including the assignment of GPRs to reactions with no gene rule in the model. The validation also allowed noticing that *TranSyT* reliance on TCDB might impair some predictions, as various TC families are lacking reactions and almost 5 000 transporter systems are not associated with metabolites. This represents a limitation, as homologies with these systems, will not be associated to reactions.

However, *TranSyT* is consistent in its classifications and uses the same methods to generate automatically reactions for all transporter system, with virtually no human interaction.

All objectives of this project were accomplished, nevertheless the integration with KBase is still an ongoing task. A standalone version of *TranSyT* is freely available at https://gitlab.bio.di.uminho.pt/TranSyT.

6.2 Prospect for future work

The search for reliable alternatives to TCDB is of paramount importance, as the expansion of the database will ultimately increase the quality of the classifications, while decreasing the dependency in one single source of information. Moreover, the search of more efficient predictors for the optional features is a priority, as the levels of confidence are a good base to assess *TranSyT*'s classifications. Current prediction tools, besides not providing instances to be run locally in *TranSyT*'s webserver, slow down the whole process, reason why an alternative must be sought.

6.3 Outcomes

During the development of this thesis the following outcomes were accomplished:

- Cruz, F.; Lagoa, D.; Mendes, J.; Rocha, I.; Ferreira, E. C.; Rocha, M.; Dias, O. *SamPler* a Novel Method for Selecting Parameters for Annotation Routines, submitted (under revision), 2018
- Lagoa, D; Liu, F.; Ferreira, E.C.; Faria, J.; Henry, C.; Dias, O. Towards a genome-wide transport systems encoding genes tracker. 5th Conference on Constraint-Based Reconstruction and Analysis, Seatle, USA, 2018 (Poster)

BIBLIOGRAPHY

[1] 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.

[2] B. Alberts. *Molecular biology of the cell*. Garland science, 2017.

[3] 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, 1990.

[4] F. Aplop and G. Butler. Transath: Transporter prediction via annotation transfer by homology. 2017.

[5] 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.

[6] P. Artimo, M. Jonnalagedda, K. Arnold, D. Baratin, G. Csardi, E. De Castro, S. Duvaud, V. Flegel, A. Fortier, E. Gasteiger, et al. Expasy: Sib bioinformatics resource portal. *Nucleic acids research*, 40(W1):W597–W603, 2012.

[7] R. K. Aziz, D. Bartels, A. A. Best, M. DeJongh, T. Disz, R. A. Edwards, K. Formsma, S. Gerdes, E. M. Glass, M. Kubal, et al. The rast server: rapid annotations using subsystems technology. *BMC genomics*, 9(1):75, 2008.

[8] P. G. Bagos, T. D. Liakopoulos, and S. J. Hamodrakas. Finding beta-barrel outer membrane proteins with a markov chain model. *WSEAS Transactions on Biology and Biomedicine*, 2(1):186–189, 2004.

[9] P. G. Bagos, T. D. Liakopoulos, and S. J. Hamodrakas. Evaluation of methods for predicting the topology of β-barrel outer membrane proteins and a consensus prediction method. *BMC bioinformatics*, 6(1):7, 2005.

[10] P. G. Bagos, T. D. Liakopoulos, I. C. Spyropoulos, and S. J. Hamodrakas. Pred-tmbb: a web server for predicting the topology of β-barrel outer membrane proteins. *Nucleic acids research*, 32(suppl_2):W400–W404, 2004.

[11] A. Barret et al. Enzyme nomenclature: Recommendations of the nomenclature committee of the international union of biochemistry and molecular biology. *Academic, San Diego, CA*, 1992.

[12] S. A. Becker, A. M. Feist, M. L. Mo, G. Hannum, B. Ø. Palsson, and M. J. Herrgard. Quantitative prediction of cellular metabolism with constraint-based models: the cobra toolbox. *Nature protocols*, 2(3):727, 2007.

[13] F. S. Berven, K. Flikka, H. B. Jensen, and I. Eidhammer. Bomp: a program to predict integral β-barrel outer membrane proteins encoded within genomes of gram-negative bacteria. *Nucleic acids research*, 32(suppl_2):W394–W399, 2004.

[14] J. Boele, B. G. Olivier, and B. Teusink. Fame, the flux analysis and modeling environment. *BMC systems biology*, 6(1):8, 2012.

[15] P. Bogaerts, K. M. Gziri, and A. Richelle. From mfa to fba: Legitimating objective function and linear constraints. *IFAC-PapersOnLine*, 49(7):460–465, 2016.

[16] N. R. Boyle, A. A. Shastri, and J. A. Morgan. Network stoichiometry. In *Plant Metabolic Networks*, pages 211–243. Springer, 2009.

[17] S. Brohée, R. Barriot, Y. Moreau, and B. André. Ytpdb: a wiki database of yeast membrane transporters. *Biochimica et Biophysica Acta (BBA)-Biomembranes*, 1798(10):1908–1912, 2010.

[18] 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, 2003.

[19] C. Camacho, G. Coulouris, V. Avagyan, N. Ma, J. Papadopoulos, K. Bealer, and T. L. Madden. Blast+: architecture and applications. *BMC bioinformatics*, 10(1):421, 2009.

[20] R. Caspi, H. Foerster, C. A. Fulcher, P. Kaipa, M. Krummenacker, M. Latendresse, S. Paley, S. Y. Rhee, A. G. Shearer, C. Tissier, et al. The metacyc database of metabolic pathways and enzymes and the biocyc collection of pathway/genome databases. *Nucleic acids research*, 36(suppl_1):D623–D631, 2007.

[21] J.-M. Chang, P. Di Tommaso, J.-F. Taly, and C. Notredame. Accurate multiple sequence alignment of transmembrane proteins with psi-coffee. *BMC bioinformatics*, 13(4):S1, 2012.

[22] A. R. Choudhury and M. Novič. Predβtm: a novel β-transmembrane region prediction algorithm. *PloS one*, 10(12):e0145564, 2015.

[23] U. Consortium. The universal protein resource (uniprot). *Nucleic acids research*, 35(suppl_1):D193–D197, 2006.

[24] N. R. Coordinators. Database resources of the national center for biotechnology information. *Nucleic acids research*, 44(Database issue):D7, 2016.

[25] D. Croft, A. F. Mundo, R. Haw, M. Milacic, J. Weiser, G. Wu, M. Caudy, P. Garapati, M. Gillespie, M. R. Kamdar, et al. The reactome pathway knowledgebase. *Nucleic acids research*, 42(D1):D472–D477, 2013.

[26] F. Cruz, D. Lagoa, J. Mendes, I. Rocha, E. C. Ferreira, M. Rocha, and O. Dias. *SamPler* a novel method for selecting parameters for annotation routines. *Submitted for publication*, 2018.

[27] W.-D. Deckwer, D. Jahn, D. Hempel, and A.-P. Zeng. Systems biology approaches to bioprocess development. *Engineering in life sciences*, 6(5):455–469, 2006.

[28] K. Degtyarenko, P. De Matos, M. Ennis, J. Hastings, M. Zbinden, A. McNaught, R. Alcántara, M. Darsow, M. Guedj, and M. Ashburner. Chebi: a database and ontology for chemical entities of biological interest. *Nucleic acids research*, 36(suppl_1):D344–D350, 2007.

[29] O. Dias, D. Gomes, P. Vilaça, J. Cardoso, M. Rocha, E. C. Ferreira, and I. Rocha. Genome-wide semi-automated annotation of transporter systems. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 14(2):443–456, 2017.

[30] O. Dias and I. Rocha. Systems biology in fungi. *Molecular Biology of Food and Water Borne Mycotoxigenic and Mycotic Fungi. CRC Press, Boca Raton*, pages 69–92, 2015.

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

[32] Y. Ding and C. E. Lawrence. A statistical sampling algorithm for rna secondary structure prediction. *Nucleic acids research*, 31(24):7280–7301, 2003.

[33] L. Dobson, I. Reményi, and G. E. Tusnády. Cctop: a consensus constrained topology prediction web server. *Nucleic acids research*, 43(W1):W408–W412, 2015.

[34] S. R. Eddy. Profile hidden markov models. *Bioinformatics*, 14(9):755–763, 1998.

[35] L. D. Elbourne, S. G. Tetu, K. A. Hassan, and I. T. Paulsen. Transportdb 2.0: a database for exploring membrane transporters in sequenced genomes from all domains of life. *Nucleic acids research*, 45(D1):D320–D324, 2016.

[36] A. Elofsson and G. v. Heijne. Membrane protein structure: prediction versus reality. *Annu. Rev. Biochem.*, 76:125–140, 2007.

[37] P. Fariselli, C. Savojardo, P. L. Martelli, and R. Casadio. Grammatical-restrained hidden conditional random fields for bioinformatics applications. *Algorithms for Molecular Biology*, 4(1):13, 2009.

[38] 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):121, 2007.

[39] X. Feng, Y. Xu, Y. Chen, and Y. J. Tang. Microbesflux: a web platform for drafting metabolic models from the kegg database. *BMC systems biology*, 6(1):94, 2012.

[40] D. R. Flower. The lipocalin protein family: structure and function. *Biochemical Journal*, 318(1):1–14, 1996.

[41] G. D. Forney. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

[42] T. C. Freeman Jr and W. C. Wimley. Tmbb-db: a transmembrane β-barrel proteome database. *Bioinformatics*, 28(19):2425–2430, 2012.

[43] A. G. Garrow, A. Agnew, and D. R. Westhead. Tmb-hunt: a web server to screen sequence sets for transmembrane β-barrel proteins. *Nucleic acids research*, 33(suppl_2):W188–W192, 2005.

[44] A. Gevorgyan, M. E. Bushell, C. Avignone-Rossa, and A. M. Kierzek. Surreyfba: a command line tool and graphics user interface for constraint-based modeling of genome-scale metabolic reaction networks. *Bioinformatics*, 27(3):433–434, 2010.

[45] D. Glez-Peña, M. Reboiro-Jato, P. Maia, M. Rocha, F. Díaz, 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, 2010.

[46] T. Goldberg, M. Hecht, T. Hamp, T. Karl, G. Yachdav, N. Ahmed, U. Altermann, P. Angerer, S. Ansorge, K. Balasz, et al. Loctree3 prediction of localization. *Nucleic acids research*, 42(W1):W350–W355, 2014.

[47] E. Grafahrend-Belau, C. Klukas, B. H. Junker, and F. Schreiber. Fba-simvis: interactive visualization of constraint-based metabolic models. *Bioinformatics*, 25(20):2755–2757, 2009.

[48] M. M. Gromiha, S. Ahmad, and M. Suwa. Tmbeta-net: discrimination and prediction of membrane spanning β-strands in outer membrane proteins. *Nucleic acids research*, 33(suppl_2):W164–W167, 2005.

[49] M. M. Gromiha, Y. Yabuki, S. Kundu, S. Suharnan, and M. Suwa. Tmbeta-genome: database for annotated β-barrel membrane proteins in genomic sequences. *Nucleic acids research*, 35(suppl_1):D314–D316, 2006.

[50] M. M. Gromiha, Y. Yabuki, M. X. Suresh, A. M. Thangakani, M. Suwa, and K. Fukui. Tmfunction: database for functional residues in membrane proteins. *Nucleic acids research*, 37(suppl_1):D201–D204, 2008.

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

[52] S. Hayat and A. Elofsson. Boctopus: improved topology prediction of transmembrane β barrel proteins. *Bioinformatics*, 28(4):516–522, 2012.

[53] S. Hayat, C. Peters, N. Shu, K. D. Tsirigos, and A. Elofsson. Inclusion of dyad-repeat pattern improves topology prediction of transmembrane β-barrel proteins. *Bioinformatics*, 32(10):1571–1573, 2016.

[54] 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, 2010.

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

[56] B. Honoré and M. Østergaard. Transcriptomics and proteomics: integration? *eLS*, 2003.

[57] A. Hoppe, S. Hoffmann, A. Gerasch, C. Gille, and H.-G. Holzhütter. Fasimu: flexible software for flux-balance computation series in large metabolic networks. *BMC bioinformatics*, 12(1):28, 2011.

[58] P. Horton, K.-J. Park, T. Obayashi, N. Fujita, H. Harada, C. Adams-Collier, and K. Nakai. Wolf psort: protein localization predictor. *Nucleic acids research*, 35(suppl_2):W585–W587, 2007.

[59] J. Hu and C. Yan. A method for discovering transmembrane beta-barrel proteins in gram-negative bacterial proteomes. *Computational biology and chemistry*, 32(4):298–301, 2008.

[60] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew. Extreme learning machine: theory and applications. *Neurocomputing*, 70(1):489–501, 2006.

[61] M. Hucka, A. Finney, H. M. Sauro, H. Bolouri, J. C. Doyle, H. Kitano, A. P. Arkin, B. J. Bornstein, D. Bray, A. Cornish-Bowden, et al. The systems biology markup language (sbml): a medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–531, 2003.

[62] M. Ikeda, M. Arai, T. Okuno, and T. Shimizu. Tmpdb: a database of experimentally-characterized transmembrane topologies. *Nucleic acids research*, 31(1):406–409, 2003.

[63] S. Jayasinghe, K. Hristova, and S. H. White. Mptopo: A database of membrane protein topology. *Protein Science*, 10(2):455–458, 2001.

[64] D. T. Jones. Improving the accuracy of transmembrane protein topology prediction using evolutionary information. *Bioinformatics*, 23(5):538–544, 2007.

[65] T. Jores and D. Rapaport. Early stages in the biogenesis of eukaryotic β-barrel proteins. *FEBS letters*, 591(17):2671–2681, 2017.

[66] L. Käll, A. Krogh, and E. L. Sonnhammer. An hmm posterior decoder for sequence feature prediction that includes homology information. *Bioinformatics*, 21(suppl_1):i251–i257, 2005.

[67] L. Käll, A. Krogh, and E. L. Sonnhammer. Advantages of combined transmembrane topology and signal peptide predictionthe phobius web server. *Nucleic acids research*, 35(suppl_2):W429–W432, 2007.

[68] M. Kanehisa, S. Goto, S. Kawashima, Y. Okuno, and M. Hattori. The kegg resource for deciphering the genome. *Nucleic acids research*, 32(suppl_1):D277–D280, 2004.

[69] P. D. Karp. Pathway databases: a case study in computational symbolic theories. *Science*, 293(5537):2040–2044, 2001.

[70] Z. A. King, J. Lu, A. Dräger, P. Miller, S. Federowicz, J. A. Lerman, A. Ebrahim, B. O. Palsson, and N. E. Lewis. Bigg models: A platform for integrating, standardizing and sharing genome-scale models. *Nucleic acids research*, 44(D1):D515–D522, 2015.

[71] S. Klamt, J. Saez-Rodriguez, and E. D. Gilles. Structural and functional analysis of cellular networks with cellnetanalyzer. *BMC systems biology*, 1(1):2, 2007.

[72] J. H. Kleinschmidt. Folding and stability of monomeric b-barrel membrane proteins. 2005.

[73] A. Klug. The discovery of the dna double helix. *DNA, Changing Science and Society*, 2004.

[74] D. Kozma, I. Simon, and G. E. Tusnády. Pdbtm: Protein data bank of transmembrane proteins after 8 years. *Nucleic acids research*, 41(D1):D524–D529, 2012.

[75] A. Krogh, B. Larsson, G. Von Heijne, and E. L. Sonnhammer. Predicting transmembrane protein topology with a hidden markov model: application to complete genomes. *Journal of molecular biology*, 305(3):567–580, 2001.

[76] A. Krogh, B. Larsson, G. Von Heijne, and E. L. Sonnhammer. Predicting transmembrane protein topology with a hidden markov model: application to complete genomes1. *Journal of molecular biology*, 305(3):567–580, 2001.

[77] S. Y. Lee. High cell-density culture of escherichia coli. *Trends in biotechnology*, 14(3):98–105, 1996.

[78] S. Y. Lee, S. B. Sohn, H. U. Kim, J. M. Park, T. Y. Kim, J. D. Orth, and B. Ø. Palsson. Genome-scale network modeling. In *Systems Metabolic Engineering*, pages 1–23. Springer, 2012.

[79] T. J. Lee, I. Paulsen, and P. Karp. Annotation-based inference of transporter function. *Bioinformatics*, 24(13):i259–i267, 2008.

[80] W. Liebermeister, F. Krause, J. Uhlendorf, T. Lubitz, and E. Klipp. Semanticsbml: a tool for annotating, checking, and merging of biochemical models in sbml format. 2009.

[81] F. Liu. *Evaluation and development of algorithms and computational tools for metabolic pathway optimization*. PhD thesis, Universidade do Minho, 7 2018.

[82] W.-m. Liu. Shear numbers of protein β-barrels: definition refinements and statistics1. *Journal of molecular biology*, 275(4):541–545, 1998.

[83] M. A. Lomize, I. D. Pogozheva, H. Joo, H. I. Mosberg, and A. L. Lomize. Opm database and ppm web server: resources for positioning of proteins in membranes. *Nucleic acids research*, 40(D1):D370–D376, 2012.

[84] R. Luo, S. Liao, S. Zeng, Y. Li, and Q. Luo. Fluxexplorer: A general platform for modeling and analyses of metabolic networks based on stoichiometry. *Chinese Science Bulletin*, 51(6):689–696, 2006.

[85] D. Machado, R. S. Costa, M. Rocha, E. C. Ferreira, B. Tidor, and I. Rocha. Modeling formalisms in systems biology. *AMB express*, 1(1):45, 2011.

[86] D. Machado, M. J. Herrgård, and I. Rocha. Stoichiometric representation of gene–protein–reaction associations leverages constraint-based analysis from reaction to gene-level phenotype prediction. *PLoS computational biology*, 12(10):e1005140, 2016.

[87] R. Mahadevan and C. Schilling. The effects of alternate optimal solutions in constraint-based genome-scale metabolic models. *Metabolic engineering*, 5(4):264–276, 2003.

[88] E. R. Mardis. The impact of next-generation sequencing technology on genetics. *Trends in genetics*, 24(3):133–141, 2008.

[89] A. McLachlan. Gene duplications in the structural evolution of chymotrypsin. *Journal of molecular biology*, 128(1):49–79, 1979.

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

[91] C. Mooney, Y.-H. Wang, and G. Pollastri. Sclpred: protein subcellular localization prediction by n-to-1 neural networks. *Bioinformatics*, 27(20):2812–2819, 2011.

[92] T. Mueller. H2 database engine, 2006.

[93] S. Mukherjee, D. Stamatis, J. Bertsch, G. Ovchinnikova, O. Verezemska, M. Isbandi, A. D. Thomas, R. Ali, K. Sharma, N. C. Kyrpides, et al. Genomes online database (gold) v. 6: data updates and feature enhancements. *Nucleic acids research*, page gkw992, 2016.

[94] H. Ogata, S. Goto, K. Sato, W. Fujibuchi, H. Bono, and M. Kanehisa. Kegg: Kyoto encyclopedia of genes and genomes. *Nucleic acids research*, 27(1):29–34, 1999.

[95] S. Pabinger, R. Rader, R. Agren, J. Nielsen, and Z. Trajanoski. Memosys: Bioinformatics platform for genome-scale metabolic models. *BMC systems biology*, 5(1):20, 2011.

[96] B. Palsson. Metabolic systems biology. *FEBS letters*, 583(24):3900–3904, 2009.

[97] P. Pareja-Tobes, R. Tobes, M. Manrique, E. Pareja, and E. Pareja-Tobes. Bio4j: a high-performance cloud-enabled graph-based data platform. *bioRxiv*, page 016758, 2015.

[98] S. Patient, D. Wieser, M. Kleen, E. Kretschmann, M. Jesus Martin, and R. Apweiler. Uniprotjapi: a remote api for accessing uniprot data. *Bioinformatics*, 24(10):1321–1322, 2008.

[99] K. R. Patil, I. Rocha, J. Förster, and J. Nielsen. Evolutionary programming as a platform for in silico metabolic engineering. *BMC bioinformatics*, 6(1):308, 2005.

[100] W. R. Pearson. An introduction to sequence similarity (homology) searching. *Current protocols in bioinformatics*, 42(1):3–1, 2013.

[101] N. T. Perna, G. Plunkett III, V. Burland, B. Mau, J. D. Glasner, D. J. Rose, G. F. Mayhew, P. S. Evans, J. Gregor, H. A. Kirkpatrick, et al. Genome sequence of enterohaemorrhagic escherichia coli o157: H7. *Nature*, 409(6819):529, 2001.

[102] C. Peters, K. D. Tsirigos, N. Shu, and A. Elofsson. Improved topology prediction using the terminal hydrophobic helices rule. *Bioinformatics*, 32(8):1158–1162, 2015.

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

[104] A. Prlić, A. Yates, S. E. Bliven, P. W. Rose, J. Jacobsen, P. V. Troshin, M. Chapman, J. Gao, C. H. Koh, S. Foisy, et al. Biojava: an open-source framework for bioinformatics in 2012. *Bioinformatics*, 28(20):2693–2695, 2012.

[105] Y. Quentin and G. Fichant. Abcdb: an abc transporter database. *Journal of molecular microbiology and biotechnology*, 2(4):501–504, 2000.

[106] P. Raman, V. Cherezov, and M. Caffrey. The membrane protein data bank. *Cellular and Molecular Life Sciences*, 63(1):36, 2006.

[107] A. Randall, J. Cheng, M. Sweredoski, and P. Baldi. Tmbpro: secondary structure, β-contact and tertiary structure prediction of transmembrane β-barrel proteins. *Bioinformatics*, 24(4):513–520, 2007.

[108] S. Rao, O. Schmidt, A. B. Harbauer, B. Schönfisch, B. Guiard, N. Pfanner, and C. Meisinger. Biogenesis of the preprotein translocase of the outer mitochondrial membrane: protein kinase a phosphorylates the precursor of tom40 and impairs its import. *Molecular biology of the cell*, 23(9):1618–1627, 2012.

[109] Q. Ren, K. H. Kang, and I. T. Paulsen. Transportdb: a relational database of cellular membrane transport systems. *Nucleic acids research*, 32(suppl_1):D284–D288, 2004.

[110] S. M. Reynolds, L. Käll, M. E. Riffle, J. A. Bilmes, and W. S. Noble. Transmembrane topology and signal peptide prediction using dynamic bayesian networks. *PLoS computational biology*, 4(11):e1000213, 2008.

[111] J. S. Richardson, E. D. Getzoff, and D. C. Richardson. The beta bulge: a common small unit of nonrepetitive protein structure. *Proceedings of the National Academy of Sciences*, 75(6):2574–2578, 1978.

[112] L. W. Riley, R. S. Remis, S. D. Helgerson, H. B. McGee, J. G. Wells, B. R. Davis, R. J. Hebert, E. S. Olcott, L. M. Johnson, N. T. Hargrett, et al. Hemorrhagic colitis associated with a rare escherichia coli serotype. *New England Journal of Medicine*, 308(12):681–685, 1983.

[113] I. Robinson, J. Webber, and E. Eifrem. *Graph databases*. " O'Reilly Media, Inc.", 2013.

[114] I. Rocha, J. Förster, and J. Nielsen. Design and application of genome-scale reconstructed metabolic models. *Microbial Gene Essentiality: Protocols and Bioinformatics*, pages 409–431, 2008.

[115] 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.

[116] M. H. Saier. A functional-phylogenetic classification system for transmembrane solute transporters. *Microbiology and Molecular Biology Reviews*, 64(2):354–411, 2000.

[117] M. H. Saier Jr, V. S. Reddy, B. V. Tsu, M. S. Ahmed, C. Li, and G. Moreno-Hagelsieb. The transporter classification database (tcdb): recent advances. *Nucleic acids research*, 44(D1):D372–D379, 2015.

[118] M. H. Saier Jr, C. V. Tran, and R. D. Barabote. Tcdb: the transporter classification database for membrane transport protein analyses and information. *Nucleic acids research*, 34(suppl_1):D181–D186, 2006.

[119] C. Savojardo, P. Fariselli, and R. Casadio. Betaware: a machine-learning tool to detect and predict transmembrane beta-barrel proteins in prokaryotes. *Bioinformatics*, 29(4):504–505, 2013.

[120] J. Schellenberger, R. Que, R. M. Fleming, I. Thiele, J. D. Orth, A. M. Feist, D. C. Zielinski, A. Bordbar, N. E. Lewis, S. Rahmanian, et al. Quantitative prediction of cellular metabolism with constraint-based models: the cobra toolbox v2. 0. *Nature protocols*, 6(9):1290, 2011.

[121] I. Schomburg, A. Chang, and D. Schomburg. Brenda, enzyme data and metabolic information. *Nucleic acids research*, 30(1):47–49, 2002.

[122] D. Schuenemann, P. Amin, E. Hartmann, and N. E. Hoffman. Chloroplast secy is complexed to sece and involved in the translocation of the 33-kda but not the 23-kda subunit of the oxygen-evolving complex. *Journal of Biological Chemistry*, 274(17):12177–12182, 1999.

[123] G. E. Schulz. Porins: general to specific, native to engineered passive pores. *Current opinion in structural biology*, 6(4):485–490, 1996.

[124] G. E. Schulz. β-barrel membrane proteins. *Current opinion in structural biology*, 10(4):443–447, 2000.

[125] R. Schwacke, A. Schneider, E. van der Graaff, K. Fischer, E. Catoni, M. Desimone, W. B. Frommer, U.-I. Flügge, and R. Kunze. Aramemnon, a novel database for arabidopsis integral membrane proteins. *Plant physiology*, 131(1):16–26, 2003.

[126] D. Segre, D. Vitkup, and G. M. Church. Analysis of optimality in natural and perturbed metabolic networks. *Proceedings of the National Academy of Sciences*, 99(23):15112–15117, 2002.

[127] T. Shlomi, O. Berkman, and E. Ruppin. Regulatory on/off minimization of metabolic flux changes after genetic perturbations. *Proceedings of the National Academy of Sciences of the United States of America*, 102(21):7695–7700, 2005.

[128] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of molecular biology*, 147(1):195–197, 1981.

[129] N. Swainston, K. Smallbone, P. Mendes, D. B. Kell, and N. W. Paton. The subliminal toolbox: automating steps in the reconstruction of metabolic networks. *Journal of Integrative Bioinformatics (JIB)*, 8(2):187–203, 2011.

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

[131] K. D. Tsirigos, P. G. Bagos, and S. J. Hamodrakas. Ompdb: a database of β-barrel outer membrane proteins from gram-negative bacteria. *Nucleic acids research*, 39(suppl_1):D324–D331, 2010.

[132] K. D. Tsirigos, A. Elofsson, and P. G. Bagos. Pred-tmbb2: improved topology prediction and detection of beta-barrel outer membrane proteins. *Bioinformatics*, 32(17):i665–i671, 2016.

[133] K. D. Tsirigos, S. Govindarajan, C. Bassot, Å. Västermark, J. Lamb, N. Shu, and A. Elofsson. Topology of membrane proteinspredictions, limitations and variations. *Current opinion in structural biology*, 50:9–17, 2018.

[134] K. D. Tsirigos, C. Peters, N. Shu, L. Käll, and A. Elofsson. The topcons web server for consensus prediction of membrane protein topology and signal peptides. *Nucleic acids research*, 43(W1):W401–W407, 2015.

[135] G. E. Tusnády, L. Kalmár, H. Hegyi, P. Tompa, and I. Simon. Topdom: database of domains and motifs with conservative location in transmembrane proteins. *Bioinformatics*, 24(12):1469–1470, 2008.

[136] G. E. Tusnady, L. Kalmar, and I. Simon. Topdb: topology data bank of transmembrane proteins. *Nucleic acids research*, 36(suppl_1):D234–D239, 2007.

[137] G. E. Tusnady and I. Simon. The hmmtop transmembrane topology prediction server. *Bioinformatics*, 17(9):849–850, 2001.

[138] H. Viklund, A. Bernsel, M. Skwark, and A. Elofsson. Spoctopus: a combined predictor of signal peptides and membrane protein topology. *Bioinformatics*, 24(24):2928–2929, 2008.

[139] H. Viklund and A. Elofsson. Octopus: improving topology prediction by two-track ann-based preference scores and an extended topological grammar. *Bioinformatics*, 24(15):1662–1668, 2008.

[140] J. Waldispühl, C. W. O'Donnell, S. Devadas, P. Clote, and B. Berger. Modeling ensembles of transmembrane β-barrel proteins. *Proteins: Structure, Function, and Bioinformatics*, 71(3):1097–1112, 2008.

[141] W. B. Whitaker, J. A. Jones, R. K. Bennett, J. E. Gonzalez, V. R. Vernacchio, S. M. Collins, M. A. Palmer, S. Schmidt, M. R. Antoniewicz, M. A. Koffas, et al. Engineering the biological conversion of methanol to specialty chemicals in escherichia coli. *Metabolic engineering*, 39:49–59, 2017.

[142] W. Wiechert, M. Möllney, S. Petersen, and A. A. de Graaf. A universal framework for 13c metabolic flux analysis. *Metabolic engineering*, 3(3):265–283, 2001.

[143] U. Wittig, R. Kania, M. Golebiewski, M. Rey, L. Shi, L. Jong, E. Algaa, A. Weidemann, H. Sauer-Danzwith, S. Mir, et al. Sabio-rkdatabase for biochemical reaction kinetics. *Nucleic acids research*, 40(D1):D790–D796, 2011.

[144] J. Wright and A. Wagner. The systems biology research tool: evolvable open-source software. *BMC systems biology*, 2(1):55, 2008.

[145] N. Y. Yu, M. R. Laird, C. Spencer, and F. S. Brinkman. Psortdban expanded, auto-updated, user-friendly protein subcellular localization database for bacteria and archaea. *Nucleic acids research*, 39(suppl_1):D241–D244, 2010.

[146] 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, et al. Psortb 3.0: improved protein subcellular localization prediction with refined localization subcategories and predictive capabilities for all prokaryotes. *Bioinformatics*, 26(13):1608–1615, 2010.

A

SUPPORT MATERIAL

A.1 BLAST RESULT EXAMPLE TO SELECT TC FAMILY

Table S1.: Blast output of the alignment of entry UniProt entry S7V9F2 against TCDB records.

Begin of Table S1

Accession	TC Number	Description	Bit Score	E-value
Q6VV69	2.A.6.2.23	Periplasmic linker protein - Burkh...	960	e-126
P0AE06	8.A.1.6.1	Acriflavine resistance protein A -...	955	e-126
Q6V6X9	2.A.6.2.18	Membrane fusion protein - Pseudomo...	949	e-125
P52477	2.A.6.2.6	Multidrug resistance protein MexA ...	930	e-122
Q9WWZ9	2.A.6.2.9	Toluene efflux pump periplasmic li...	900	e-117
Q93K41	2.A.6.2.49	AcrA protein OS=Klebsiella pneumon...	879	e-114
Q9F241	2.A.6.2.42	Putative membrane fusion protein O...	876	e-114
Q0VQY5	2.A.6.2.46	Multidrug/solvent RND membrane fus...	878	e-113
Q8GC84	2.A.6.2.19	EefA lipoprotein precursor - Enter...	858	e-111
Q93PU5	2.A.6.2.11	Toluene efflux pump periplasmic li...	859	e-111
Q8Y3G9	2.A.6.2.45	Probable acriflavin resistance lip...	845	e-109
Q2FD95	2.A.6.2.29	RND family drug transporter - Acin...	840	e-108
Q9KWV5	2.A.6.2.10	Toluene efflux pump periplasmic li...	824	e-106
P37636	8.A.1.6.3	Multidrug resistance protein MdtE ...	797	e-102
Q2AAU4	2.A.6.2.26	Membrane fusion protein VmeA - Vib...	778	e-99
Q9RBY9	2.A.6.2.41	SmeA OS=Stenotrophomonas maltophil...	768	e-97
Q67GM1	2.A.6.2.4	AdeD OS=Acinetobacter sp. 4365 GN=...	699	e-87
P43505	2.A.6.2.5	Membrane fusion protein mtrC - Nei...	694	e-86
Q2FD71	2.A.6.2.40	AdeA membrane fusion protein OS=Ac...	686	e-85
O87935	2.A.6.2.24	Putative membrane fusion protein A...	673	e-83

Continuation of Table S1

Accession	TC Number	Description	Bit Score	E-value
Q9ZNG9	2.A.6.2.21	MexX - Pseudomonas aeruginosa.	563	e-67
B2FLY3	2.A.6.2.14	Putative multidrug efflux protein,...	458	e-52
Q9HY86	2.A.6.2.34	Probable Resistance-Nodulation-Cel...	423	e-47
Q8ZRG8	2.A.6.2.25	Putative cation efflux pump - Salm...	421	e-46
C5IZH0	2.A.6.2.47	Acriflavin resistance protein AcrA...	419	e-46
Q79MP5	2.A.6.2.31	Membrane fusion protein SdeA - Ser...	409	e-45
Q2FD82	2.A.6.2.44	Putative RND family drug transport...	398	e-43
Q9I3R2	2.A.6.2.35	Probable Resistance-Nodulation-Cel...	384	e-41
Q9I0V5	2.A.6.2.39	Probable Resistance-Nodulation-Cel...	382	e-41
Q8RTE5	2.A.6.2.22	CmeA - Campylobacter jejuni	371	e-40
P76397	8.A.1.6.2	Multidrug resistance protein mdtA ...	329	e-34
Q4VSJ3	2.A.6.2.20	Probable RND efflux membrane-fusio...	230	e-21
Q9HWH5	2.A.6.2.32	Probable Resistance-Nodulation-Cel...	227	e-21
A0Q8A4	2.A.6.2.28	Membrane fusion protein OS=Francis...	229	e-20
Q1LCD7	2.A.6.1.6	Membrane fusion protein (MFP-RND) ...	222	e-20
Q84BQ3	3.A.1.122.12	Putative periplasmic protein OS=Ps...	219	e-19
Q9I6X6	2.A.6.2.27	Probable Resistance-Nodulation-Cel...	215	e-19
P75830	3.A.1.122.1	Macrolide-specific efflux protein ...	202	e-17
Q9I6X5	2.A.6.2.27	Probable Resistance-Nodulation-Cel...	194	e-16
Q2FD52	3.A.1.122.18	Efflux transporter, RND family, MF...	192	e-16
Q1D664	2.A.6.1.10	Putative cation efflux system prot...	181	e-14
Q1CVN0	2.A.6.1.11	Putative cobalt-zinc-cadmium resis...	175	e-14
P37973	8.A.1.2.1	Nickel and cobalt resistance prote...	166	e-13
Q88RT5	2.A.6.1.5	Cobalt/zinc/cadmium efflux RND tra...	166	e-13
P13510	2.A.6.1.2	Cobalt-zinc-cadmium resistance pro...	156	e-11
Q44585	2.A.6.1.12	Nickel-cobalt-cadmium resistance p...	153	e-11
Q1CZ64	2.A.6.3.7	Efflux transporter, RND family, MF...	147	e-10

End of Table S1