GENOME-SCALE METABOLIC MODEL RECONSTRUCTION AND VALIDATION

by

Leila Rezaei

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

at

Dalhousie University
Halifax, Nova Scotia
December 2023

To my parents, Shahdokht and Alireza, and my siblings, Shahrokh, Sharareh, Mohammadreza, Shiva, and Ali

Table of Contents

List of Tables

List of Figures

Abstract

Microorganisms play a fundamental role in production of valuable products. Comprehending the complex metabolism of microorganisms is essential for enhancing the bioprocesses, leading to a profitable increase in both the quality and quantity of products. The construction of genome-scale metabolic models enables us to explore the metabolism of cells and their behaviour under different environmental conditions. This thesis presents a comprehensive exploration of genome-scale metabolic model for a recently sequenced microorganism, the thraustochytrid strain T18, with a particular focus on variation in carbon source (glucose/xylose) in the culture media and the investigation of the production of docosahexaenoic acid (DHA). The first genome-scale metabolic model for T18 was constructed with 2252 reactions and 1952 metabolites. The analysis of T18 revealed that the fundamental difference between growth on glucose and xylose was the utilization of cofactors such as NADPH and NADH. Furthermore, we identified 148 reactions essential for growth on xylose that were not required for growth on glucose. However, due to the broad incompleteness of T18 genomic data, approximately 40 percent dead-end metabolites were present in the model, and no amount of gap-filling was sufficient to simulate the production of fatty acids such as DHA. Consequently, to identify the ideal tool for gap-filling in our model, a comprehensive and systematic assessment of available gap-filling tools was conducted. We developed our own straightforward evaluation framework, which enabled us to demonstrate that gap-filling is primarily model-dependent. Even with widely used tools like COBRApy, Meneco, and CarveMe, only about 50 percent of essential reactions removed/gaps across all 108 published models available on the BiGG database could be identified. Furthermore, we developed and successfully implemented a novel ranking approach in the Python-based SBPRank package. SBPRank incorporates network topology properties, including betweenness and proximity, as well as phylogenetic information, similarity, to rank reactions for the gap-filling process. Our innovative trimming approach efficiently narrows down a large pool of reaction database to a smaller and more specific subset, significantly improving the speed of the gap-filling process and enhancing model validity. The results indicate that when 10-30 percent of reactions are missing from a model, searching the top 5 percent of the ranked universal pool can be sufficient for gap-filling. Moreover, in cases with 10-85 percent of missing reactions, exploring only the top 20 percent of the ranked universal pool can identify suitable reactions. This reduction in the size of the universal database enables more manageable simulation times while still achieving effective and accurate gap-filling results.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisor, Dr. Stanislav Sokolenko, for his unwavering support, invaluable guidance, and constant encouragement throughout my PhD journey. His expertise, mentorship, and dedication have been instrumental in shaping my research and academic growth.

I am also thankful to the members of my thesis committee, Dr. Suzanne M Budge, and Dr. Wendy Gentleman, for their insightful feedback, and generous investment of time in evaluating my work. Likewise, I am grateful to Dr. Brian Ingalls for the time and effort he dedicated to reviewing and evaluating my work as the external examiner for my thesis.

In addition, I would like to appreciate Dr. David Woodhall and Dr. Roberto Armenta from Mara Renewables Corporation for provision of data and their collaborative efforts.

I extend my heartfelt appreciation to my labmates Michelle White, Theodore Street, and Kathy Isaac for their valuable friendship and support. My gratitude extends to Paula Colicchio and Julie O'Grady for their administrative support.

Finally, I would like to thank my friends, Jenna Choi and Aryan Samadi, for their constant support, understanding and encouragement.

Chapter 1

Introduction

The manufacture of many of today's complex molecules and simple chemicals relies on microorganisms and regardless of the final product, the underlying goal in all bioprocesses is to achieve high product quality and quantity while minimizing cost[1]. Achieving this goal requires an in-depth understanding of cellular behaviour. Metabolism, a complex network of chemical reactions inside cells, plays a key role in investigating microorganisms' behaviour and using them to our advantage. One expression of this understanding is through the development of metabolic models, which serve to incorporate available data into a single framework for the purpose of describing and eventually evaluating cellular function. Such models can later be used to guide process optimization. For instance, by identifying feeding supplements or targets for genetic engineering, a large number of combinatorial gene alterations can be tested. Furthermore, these models can offer the best metabolic pathways or routes to achieve the metabolic goals.

Modern high-throughput techniques have enabled the detailed characterization of whole-genome sequencing and annotation (genomics), the measurement of the messenger RNA molecules that are synthesized under specific condition (transcriptomics), the quantification protein abundance, interactions, and functional states (proteomics) and the measurements of the presence and concentration of metabolites (metabolomics), which referred together as omics data, in the study of living organisms. Although the analysis of each of the mentioned data sets have been the center of attention in recent decades, such lists only provide basic information and limited insight into the cellular processes without additional biological context. Therefore, the incorporation of omics data and creation of a single holistic model which represent cells as a system and the relationship between various compartments rather than an individual cellular component is essential.

1.1 Technical definitions

1.1.1 Genome-scale metabolic models

The reconstruction of Genome-Scale Metabolic Models (GEMs) has made it possible to gain a comprehensive and holistic understanding of cell metabolism. A GEM is a mathematical representation of the metabolism for an organism that relates metabolites and reactions information within a cell. All the biochemical information that is hypothesized to be present in the GEM of an organism is derived from the annotated genome of the target organism. The information about metabolites and reactions is then complied into a matrix known as stoichiometric or S matrix. The S matrix presents the stoichiometric coefficient of every single metabolite in each reaction, with each column and row representing a reaction and a metabolite of this matrix, respectively. Furthermore, other types of reactions such as biomass reaction, exchange reactions and transport reactions should be incorporated into the S matrix.

- Biomass reactions: An abstract reaction that is described in terms of major groups of biomolecules including proteins, carbohydrates, lipids, DNA, RNA and free fatty acids. The biomass, a generic term referring to cell growth, sums the mole fraction of each precursor required to produce 1 gram of dry cell weight.

- Exchange reactions: A set of reactions defines the specific medium on which the model organism grows. Exchange reactions are added to simulate the compounds that should be taken up from the outside environment into the cell, and those should be secreted from the cell (waste products). For example, uptake of glucose or amino acids, and excretion of byproducts like lactate or CO_2. Exchange reactions typically involve passive or facilitated diffusion or active transport processes. The directionality of the exchange depends on concentration gradients or active transport mechanisms.

- Transport reactions: A collection of reactions that transport metabolites into the cell from extracellular space. For instance, transport of metabolites into or out of organelles such as the mitochondria, endoplasmic reticulum, or nucleus. Transport reactions often involve specific transport proteins or carriers that facilitate the movement of metabolites across membranes. These carriers may use energy (e.g., ATP) to actively transport metabolites against a concentration gradient.

1.1.2 Flux Balance Analysis(FBA)

Flux balance analysis (FBA) is the most popular method for evaluating genome-scale metabolic models[2]. In FBA, a flux balance is formulated around every single metabolite hypothesised to be in the organism under steady state condition and a set of constraints including mass and energy balance along with thermodynamic feasibility (such as directionality of reactions) are considered. In other words, the reactions are simply assumed to be input and output fluxes, and there is no net accumulation of metabolites in the cell other than generic cellular growth. Since most metabolites participate in many metabolic reactions, the number of metabolites is smaller than the number of reactions in a network[3, 4, 5]. This formulation results in a system of linear equations that represent mass or energy balance around the chemical reactions to be underdetermined since most metabolites participate in many metabolic reactions, the number of metabolites is smaller than the number of reactions in a network.

Therefore, there is an infinite number of flux vectors that can satisfy the steady state conservation of the mass equations[6]. Hence, to calculate the physiologically relevant flux values, which often form the basis for cellular growth, an optimization problem should be performed[7]. This optimization problem involves specifying an objective function, typically the biomass reaction. Eventually, the solution to this optimization problem is a feasible flux space that satisfies all the constraints and the calculated flux vector support the optimal state[8]. It should be noted that steady state assumption is acceptable due to the fact that metabolism is a rapid process in comparison with the other cellular processes like cell division or regulation. The following linear program can be used to represent the general FBA problem:

$$maxz = \sum_{j=1}^{n} c_j \nu_j \tag{1.1}$$

$$\text{subject to} \sum_{j=1}^{n} s_{ij} \nu_j = 0, i = 1, ..., m \tag{1.2}$$

$$\nu_j(LB) \leq \nu_j \leq \nu_j(UB), j = 1, ..., n \tag{1.3}$$

where c_j is the objective function coefficient of reaction flux ν_j defining the objective function z as a linear combination of the reaction fluxes. Equation 2 represents a system of linear equations where s_{ij} is stoichiometry coefficients of metabolite i in reaction j, while Equation 3 specifies the upper ($\nu_j(UB)$) and lower ($\nu_j(LB)$) bounds on the flux variables. Also, m and n represents metabolites and reactions, respectively. FBA simulations took the form as seen in Equations 1-3 where the flux of the biomass reaction was selected as the objective[2, 5]. Furthermore, an overview of the Flux Balance Analysis (FBA) steps is presented for an example involving only three metabolites in Figure 1.1.

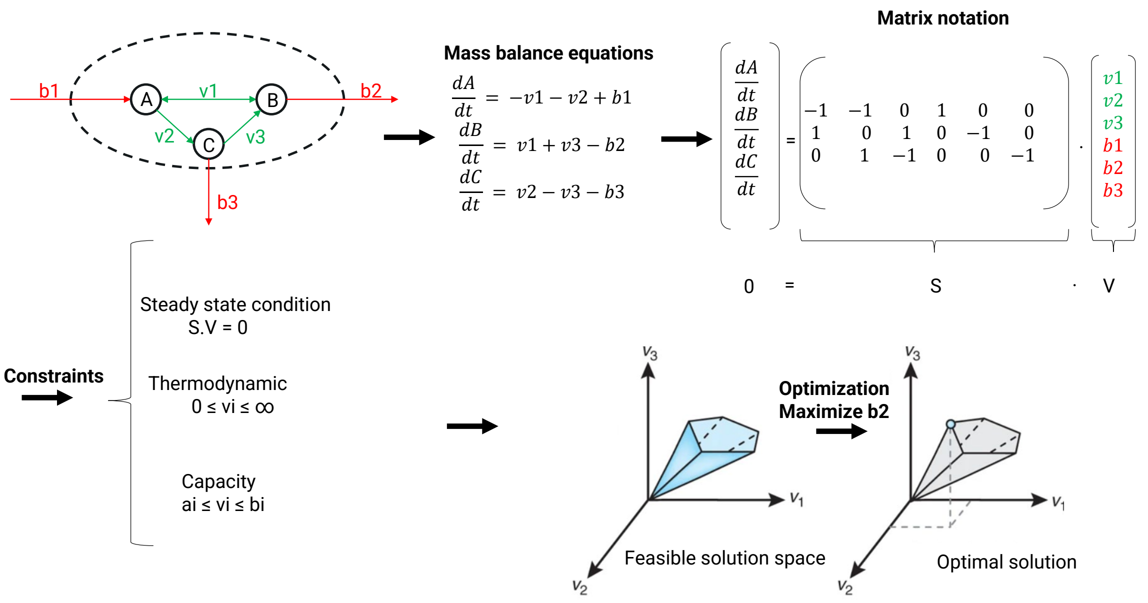

Figure 1.1: An overview of the Flux Balance Analysis (FBA) steps for a small model including three metabolites. b1, b2, and b3 represent boundary/exchange reactions, while v1, v2, and v3 denote internal reactions. The objective function is the maximization of b2.

1.1.3 General model inconsistencies and gap-filling

Although GEMs have established themselves as a valuable approach for comprehending cellular behaviour under particular conditions, they often contain gaps. Gaps occur when a reaction that consumes or produces a metabolite is absent from the model, resulting in dead-end metabolites and blocked reactions. The main reasons for presence of gaps are incomplete genome annotation of the target organism, inaccuracies in genome annotation methods or the reaction produces or consumes a specific metabolite has not been discovered yet[9, 10]. All of these factors can affect models' performance and predictions. Since enzymes play a crucial role in the majority of reactions in a metabolic model, the correct prediction of enzymes allocated to genes in a sequenced genome is of paramount importance for the model's performance[11, 12]. Therefore, GEMs often require gap-filling to enhance their analysis and interpretation. Gap-filling involves identifying and addressing gaps in the metabolic network by incorporating candidate reactions from biochemical databases that could complete the metabolic network. This process can be tedious and time-consuming when done manually, while automated methods have considerable limitations[13].

1.2 Thesis objectives

1.2.1 Objective 1: Genome-scale metabolic model reconstruction

The primary objective of this research project was to reconstruct a comprehensive genome-scale metabolic model for the recently sequenced thraustochytrid strain (T18) from local biotechnology company (Mara Renewables Corporation), which serves as an important source for fatty acid production. This objective encompasses the following specific aims:

- Gather and assess all available data such as genomic and proteomic data for T18, ensuring data quality and completeness.
- Generate a genome-scale metabolic model for T18 that accurately represents the metabolic capabilities and constraints.
- Explore the metabolic capabilities of T18, including its pathways for utilizing xylose as a low-cost carbon source, as well as its fatty acid synthesis pathways.

1.2.2 Objective 2: Addressing gaps in the genome-scale metabolic models

The second objective of this thesis is to develop strategies and methodologies to address the challenges and gaps encountered during the reconstruction of the genome-scale metabolic model. This objective involves the following specific aims:

- Evaluate the performance of current gap-filling tools in order to identify the ideal gap-filling tool, highlighting areas of uncertainty and potential inaccuracies and limitations.

- Develop a framework to enhance the GEM's accuracy and speeding up the gap-filling process by narrowing down a large pool of reactions, a database, to a smaller, specific, and relevant set that can be effectively utilized during gap-filling. An approach that can especially be valuable for research on organisms with limited available data, as they frequently have many gaps in their models.

In this thesis, we will delve into a comprehensive exploration of the genome-scale metabolic model reconstruction and validation. The research objectives are organized into distinct chapters, each contributing to a deeper understanding of the subject matter. Chapter 2 provides a thorough review of the relevant literature, establishing the framework and context for our study. Chapter 3 outlines the process of genome-scale metabolic reconstruction for thraustochytrid strain T18 including data collection, analysis, and outcomes. Chapter 4 presents the results of our systematic assessment of available gap-filling tools. In Chapter 5, we introduce our novel framework, SBPRank, to estimate the required universal pool size to fill gaps in the metabolic model. Finally, Chapter 6 summarizes the key findings, highlights the contributions to the field, and suggests potential avenues for future research.

Chapter 2

Literature Review

2.1 Introduction

A genome scale metabolic model (GEM) is a computational representation of the entire set of metabolic reactions that occur within an organism's cell or a specific cellular compartment[6, 14]. GEMs are constructed using genomic information and biochemical knowledge for an organism and are amenable to simulation, enabling the prediction of metabolic fluxes, thereby facilitating various systems-level studies of complex metabolism. There has been significant progress in the development and simulation of GEMs across a wide range of organisms including bacteria, archaea, and eukaryotes[15, 16]. Gu et al.,[17] reported that GEMs for more than 6239 organisms including 5897 for bacteria, 127 for archaea, and 215 for eukaryotes have been constructed since the reconstruction of the first GEM for *Haemophilus influenzae* in 1999. The types and numbers of applications for GEMs have steadily grown over the past two decades. These applications encompass discovering the physiology of the organisms under different conditions and understanding of their metabolic pathways, metabolic engineering of organisms for production of chemicals, optimization of industrial processes, studying human diseases and metabolic malfunctions, investigating metabolic interactions within microbial communities, and potential drug developments. In this chapter, we review the current landscape of reconstructed GEMs and explore their diverse applications as well as the challenges and limitations associated to GEMs.

2.2 Application of genome-scale metabolic models

2.2.1 Optimization of bioprocesses and production of chemicals

Microorganisms have been widely used in bioprocesses in recent years. Production of a range of chemicals, fuels, pharmaceuticals, and food ingredients depends on organisms. Thus, studying organisms at the system-level to optimize organisms and subsequently bioprocesses is beneficial. To be able to understand how metabolism works and interpret cell's behavior at system-level, omics data for the organism of interest needs to be combined. Reconstructing and utilizing GEMs provides the required data integration framework for characterizing metabolism[18]. GEMs represent the sum of all metabolic reactions that occur within the metabolism and provide a comprehensive insight. Through constraint-based modeling approaches, GEMs elucidate the complicated interplay of metabolic pathways and provide guidance for rational design strategies. They achieve this by simulating the metabolic behavior of microorganisms under various environmental conditions and genetic modifications, allowing the identification of suboptimal pathways, essential enzymatic targets for genetic manipulation, and potential metabolic bottlenecks which are the limiting steps or reactions in metabolic pathways. This predictive capability significantly reduces the time and resources required for bioproduction optimization[19, 20].

GEMs facilitate the identification of metabolic bottlenecks and potential targets, as well as the prediction of an organism's metabolic capabilities, particularly in pathways related to the synthesis of desired compounds. These predictions are founded on numerous experimental data sources, including gene expression levels, protein abundances, and metabolite concentrations[21]. GEMs provide a thorough foundation for prediction by systematically integrating annotated metabolic reactions. These models are employed to identify optimal metabolic engineering strategies for product yield and assess the effects of genetic modifications on metabolism. Furthermore, experimental validation of these predictions allows for iterative model refinement and improvement[1, 22].

Moreover, GEMs also play a pivotal role in the industrial production of biofuels and other valuable compounds. They enable the rapid exploration of metabolic pathways, providing comprehensive predictions of metabolites and facilitating the identification of potential targets for genetic manipulations that enhance product yields, reduce by-product production, and maximize overall bioproduction efficiency[23, 24]. Additionally, metabolic engineering guided by genome-scale models is not limited to increasing production yield but can also enhance product quality, strain viability, and reduce waste production. These models are crucial for the development of new microbial cell factories by predicting the behavior of genetically modified strains. In summary, genome-scale metabolic models provide deep insights and optimization strategies for industrial biological processes, making them indispensable tools for efficient and sustainable industrial production[25, 26].

Two of the most popular organisms, whose GEMs have undergone frequent updates and extensive usage, are *Escherichia coli* and *Saccharomyces cerevisiae. E. coli* is a gram-negative bacterium[27, 23, 28, 29]. Its genome sequence was released in 2000 and makes it possible to reconstruct the first GEM for strain K-12 MG1655 named iJE660. After that, iJR904 and AF1260 models were developed by adding more information such as periplasmic space and thermodynamic constraints to enhance the predictive ability of the model[30]. *E. coli* metabolic models have been the dominant standard model for a wide range of studies including the biofuels (e.g., ethanol), food and feed additives (e.g., lactic acid, amino acids, vanillin), and pharmaceuticals (e.g., Hepatitis B virus Human)[31, 32]. Furthermore, the yeast, *Saccharomyces cerevisiae*, is the popular model for eukaryotic organisms. *Saccharomyces cerevisiae* have been used for fermentation processes since the ancient times and its application have been growing for the production pharmaceutical and biochemical products[22, 33]. Since it has a high tolerance in harsh industrial situations, it has been a standard organism in industrial biotechnology. For example, *Saccharomyces cerevisiae* have been widely used for over production of bioethanol and various kinds of recombinant proteins[34, 24]. The first GEM for *Saccharomyces cerevisiae*, IFF708, was reconstructed in 2003 and since then it has been continuously updated and various models have been generated for that[35]. The latest comprehensive GEM for *Saccharomyces cerevisiae* is Yeast 8. Figure 2.1 shows the evolution of models for Saccharomyces cerevisiae. Besides these two organisms, wider range of organisms

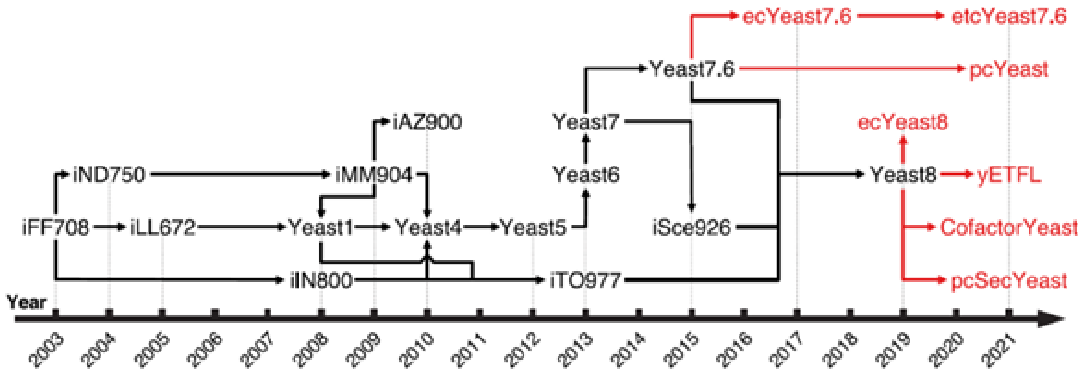

Figure 2.1: The revolution in the generation of Genome-Scale Metabolic Models (GEMs) for *Saccharomyces cerevisiae*, one of the most popular organisms in industrial biotechnology, from 2003 to 2021. The GEMs shown in black represent the classical models, while the ones in red are GEMs integrated with proteome constraints[36].

such as *Bacillus subtilis, Zymomonas mobilis, Pichia pastoris, Chlamydomonas reinhardtii, Caenorhabditis elegans, C. glutamicum*, and *Aspergillus niger* have gained attention and the GEM for them have been constructed to design and optimize their metabolic ability to produce valuable compounds such as recombinant proteins, biofuels, amino acids, and organic acids, etc[37, 21]. Several studies have investigated increase in the production of products such as lycopene, lactic acid, malic acid, polylactic acid, isobutanol, lysine, hyaluronic acid, etc[38]. For example, by using iJE660a model for *E. coli*, a triple knockout target was identified which showed a 40 percent increase in lycopene production and using iJKO04 model showed an increase to 1.75

g/L in lactic acid production[39, 40] On a genetically modified parent strain that exhibits enhanced productivity. Table 2.1 provides application examples of genome-scale metabolic models (GEMs) for various organisms.

Table 2.1: Application examples of genome-scale metabolic models (GEMs) for various organisms.

Organisms	**GEMs**	**Products**	**Results**	**Reference**
E. coli	iJE660a	Lycopene	40 percent improvement	[39]
E. coli	MBEL979	Lycopene	10.6-fold increase	[41]
E. coli	iJR904	Lactic acid	Production up to 1.75 g/L	[32]
E. coli	MBEL979	Malic acid	Production up to 9.25 g/L	[42]
E. coli	MBEL979	l-valine	Production up to 7.55 g/L	[43]
E. coli	iAF1260	Taxadiene	12-fold improvement	[44]
E. coli	MBEL979	Polylactic acid	Increase to 0.11 g/DCW	[45]
E. coli	iJO1366	Isobutanol	Increase to 8.68 g/L	[46]
E. coli	iJO1366	Succinic acid	Increase to 1.4 g/L	[47]
E. coli	iAF1260	Fatty acids	Increase to 1.7 g/L	[48]
E. coli	iJO1366	AntiEpEX-scFv	2-fold improvement	[49]
E. coli	iML1515	Lysine	Enhance to 193.6 g/L	[50]
S. cerevisiae	iFF708	Vanillin	2-fold enhance	[31]
S. cerevisiae	Yeast 8.0	7-dehydrocholesterol	Enhance to 1328 mg/L	[51]
B. thetaiotaomicron	iKS1119	Butyrate	3.4-fold improvement	[52]
B. subtilis	iYO844	Poly-γ-glutamic	Enhance to 43 g/L	[53]
C. glutamicum	iCW773	Hyaluronic Acid	Enhance to 28.7 g/L	[54]

2.2.2 Optimization of culture media

GEMs have also been widely used to design cell culture media by identifying limiting nutrients and their appropriate concentrations in the media composition as well as suggesting the appropriate additives for the culture media. Furthermore, GEMs are used to investigate the growth of organism on cheaper feedstock such as hydrolysates, lignocellulose to reduce the cost of industrial bioprocesses. For instance, in a study by Huang et al., the GEM of CHO cells (CHO-K1) was employed, and the flux balance analysis (FBA) was performed to predict cell's behaviour and immunoglobulin (IgG) production by identifying the limiting amino acids in the cell culture media. The results of this study showed that an increased concentration of leucine and valine led to a 33 percent increase in IgG productivity[55]. Tejera et al.,[56] reconstructed a

GEM for *Campylobacter jejuni* to understand its metabolism and to design and optimize the culture media for that. By the help of the model, they could discover that *Campylobacter jejuni* is auxotrophic for methionine, niacinamide, and pantothenate. Additionally, they could design a define media containing L-cysteine, L-serine, and L-glutamine as additives and achieved 1.75-fold higher growth rate than the default media (Brucella broth)[56]. In another study, the GEMs of the yeast *Pichia pastoris* such as PpaMBEL1254 and iLC915 were utilized to investigate the production of recombinant proteins by altering the oxygen and nitrogen uptake rates in the culture media[57]. Moreover, the production of human IgG was investigated by addition of 11 wheat hydrolysate lots with different properties and qualities to the media of GS-CHO cells. The results revealed the unique metabolic features like the serine and glycine metabolism could be linked to improved cell growth. Also, the distribution of the intracellular metabolic flux along with central metabolism was greatly influenced by the catabolic pathways of amino acids provided by the wheat hydrolysates[58]. In addition, the GEM for *Yarrowia lipolytica*[59] used to investigate the production of microbial oil referred to as single-cell oil (SCO) from sugarcane bagasse hydrolysates in order reduce the dependency on biodiesel and oleochemical production. The obtained result was 45.15 g/L for SCO, representing the highest in lignocellulose-based bioprocess[60]. Furthermore, a GEM of *Geobacillus thermoglucosidasius*(iGT736), suitable as a biocatalyst in lignocellulosic bioprocesses, was used to investigate its ability in production of a range of products such as ethanol and succinate[61].

2.2.3 Study of human diseases and drug development

Another significant application of GEMs is in the field of medicine and drug discovery. These models are used to create context-specific versions, including cell-specific, tissue-specific, disease-specific, and biomarker-specific models, which enable a wide range of analyses such as the phenotype prediction, rewiring of a metabolic network and the identification of drug targets and disease biomarkers. Two available models used for the investigation of metabolic diseases, drug metabolism, and personalized medicine are RECON1[62] and RECON3D[63]. RECON1 was the first GEM published for human in 2007. It represents the human metabolic network and is based on extensive curation of existing biochemical data. In addition, RECON3D is a more recent GEM was published in 2018 containing more comprehensive information. It includes multi-omics data, such as genomics, transcriptomics, and proteomics, providing a more holistic view of metabolism. RECON3D has been widely utilized in study of host-microbe interactions, microbiome modeling, enabling a deeper understanding of the human microbiome. Furthermore, context-specific GEMs are generated to study a variety of cancers such as lung, breast, prostate, liver as well as chronic diseases like obesity and non-alcoholic fatty liver disease (NAFLD).

2.2.4 GEMs for non-conventional organisms

Moreover, thanks to the significant advancements in sequencing technologies, our knowledge of omics data from various organisms with industrial potential has been continually expanding. Therefore, generation of GEMs for newly annotated organisms (non-conventional organisms) has become possible. One example of high potential microorganisms is the thraustochytrids (our case study organism), marine unicellular protists, which have proven to be a great source for production of fatty acids such as Omega-3 long chain and biofuels[25]. Additionally, *Kluyveromyces lactis* can be utilized for the production of proteins for food and feed industry, as well as pharmaceutical enzymes[21]. Moreover, *Pichia pastoris* has a tightly regulated expression system and can produce pharmaceuticals and industrial enzymes, and *Hansenula polymorpha* can produce heterologous proteins and can ferment ethanol at high temperatures[21, 64].

2.2.5 Thraustochytrid strain T18 (our case study organism)

One example of a non-conventional microorganism with high industrial potential is the thraustochytrid. Thraustochytrids are heterotrophic, oleaginous marine unicellular protists that were first described 80 years ago but have gained significant attention in recent years[65, 66, 67]. They have demonstrated to be a valuable source for the production of biofuels and lipids, including docosahexaenoic acid (DHA), eicosapentaenoic acid (EPA), carotenoids, and squalene. The high levels of DHA and EPA in the total fatty acids produced by thraustochytrids make them ideal candidates for the production of nutritional supplements[68, 69, 70]. They can be found in various marine environments, ranging from tropical to Antarctic waters[71, 72]. However, our case study, the thraustochytrid strain T18 was isolated from Atlantic Canadian waters and is known for its remarkable biomass production, containing up to 82 percent total fatty acids. T18 represents a valuable plant-based source for the production of fatty acids, especially long-chain Omega-3s, which can significantly reduce the reliance on animal-based production for these essential oils[25].

In conclusion, Genome-scale metabolic models (GEMs) are invaluable tools for comprehending the intricate interplay of metabolic processes within an organism. These models provide a comprehensive framework for simulating and analyzing metabolic pathways, thereby aiding in various applications, from understanding cellular physiology to biotechnological advancements[73, 17]. GEMs for various organisms have been developed and their role in different bioprocesses have been established. Figure 2.2 represents the evolution of GEMs.

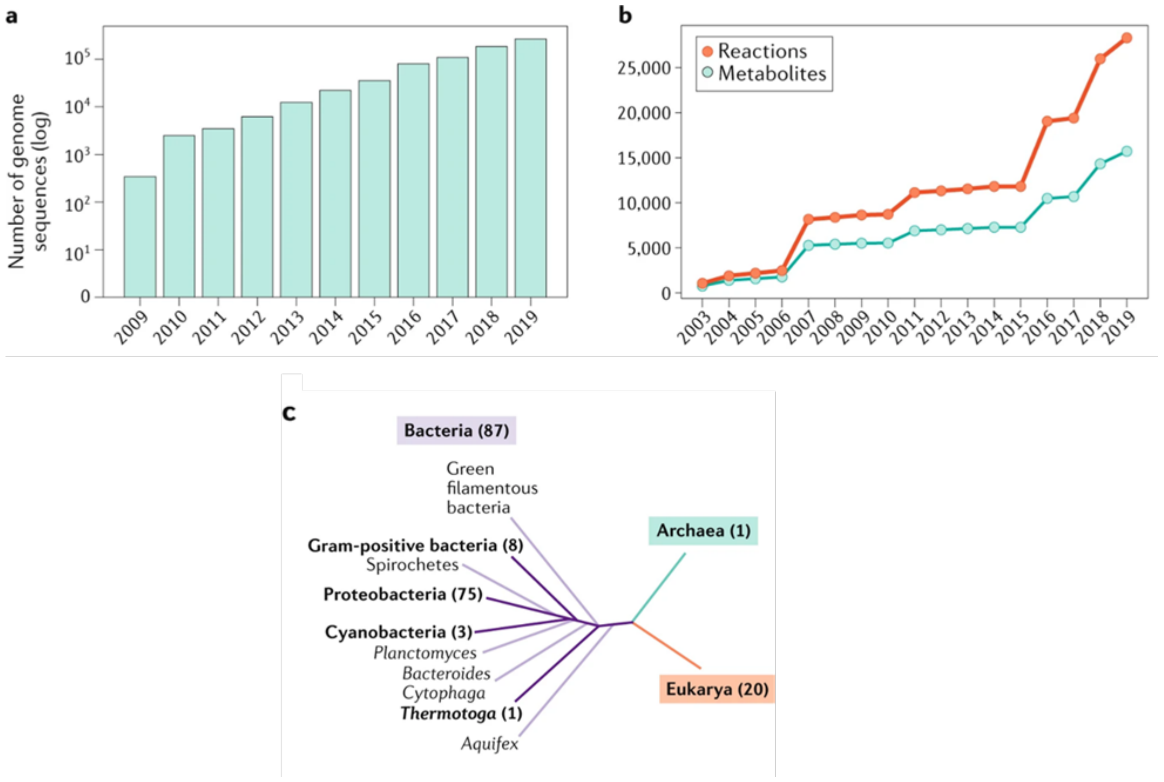

Figure 2.2: The development of GEMs for various organisms with the increase in availability of annotated genome sequences. a) increase in the genome sequences available in PATRIC database. b) development of GEMs and increase in GEMs size for 108 well curated models in the BiGG database. c) The phylogenetic domain of 108 available GEMs in the BiGG database, the numbers show the number of GEMs in each branch and categories with at least one GEM are bolded[74].

2.3 Genome-scale metabolic model reconstruction approaches

2.3.1 Conventional approach

The manual or conventional process for the reconstruction of genome-scale metabolic models is a labor-intensive process that involves meticulous data collection, integration, and curation. The process typically begins by compiling comprehensive information from genomic data, including gene annotations and genomic sequences of the target organism and then extensive literature mining and database searches to identify metabolic reactions associated with the organism of interest. Determining the reversibility and thermodynamic properties of reactions, as well as the subcellular localization of enzymes, becomes an arduous task, often requiring expert knowledge and substantial time investment. Additionally, the manual reconstruction process involves iterative refinement, as gaps and inconsistencies are identified and addressed through a combination of experimental validation and expert curation. While this method provides a high degree of control and accuracy, it is resource-intensive and may not be feasible for less-studied or complex organisms, making it a time-consuming and challenging endeavor in the field of systems biology and metabolic engineering.

2.3.2 Metabolic model reconstruction automated tools

Manual reconstruction of metabolic models is a time-consuming process that necessitates the collection of information ranging from genome sequences to the reversibility of reactions, thermodynamic data, and enzyme localization. With the growth in the various applications of GEMs of organisms, numerous approaches have been developed to automate and speed up the process of reconstruction and to identify and address issues related to gaps. A prominent example of reconstruction tools is COBRA Toolbox which focuses on the construction, analysis, and simulation of genome-scale metabolic models of microorganisms and other cellular systems. It employs constraint-based analysis like Flux Balance Analysis (FBA) and Flux Variability Analysis (FVA) to calculate feasible metabolic flux distributions within the model. Flux Variability Analysis is a computational technique used for GEMs to explore the range of feasible flux values through their metabolic network. FVA calculates the minimum and maximum flux values for each reaction in the network while maintaining an optimal solution for a specified objective function, often the biomass production rate. By identifying the variability in reaction fluxes, FVA provides insights into the flexibility and robustness of metabolic pathways. This information is valuable for understanding the potential metabolic capabilities of an organism, uncovering alternate routes for metabolite production, and assessing the impact of genetic and environmental perturbations on cellular metabolism. FVA plays a crucial role in enhancing the predictive and analytical capabilities of constraint-based metabolic models[75]. These techniques help in predicting cellular growth rates, substrate utilization, and the production of various metabolites under different conditions. COBRApy[75] is a Python-based library for constraint-based modeling of metabolic networks and COBRA.jl[8] is an advanced, high-efficiency, constraint-based modeling and analysis using the Julia programming language. COBRApy and COBRA.jl foundation is the COBRA Toolbox. Similarly, RAVEN, ModelSEED[76] , AuReMe[77] and Pathway tools[10] are other well-known tools that have been developed for mainly draft model reconstruction. ModelSEED is a web-based platform that automates the process of genome-scale metabolic model reconstruction for microbes, microbial communities, and plants. It provides access to a large database of annotated genomes and metabolic reactions, making it easier to generate models for a wide range of microorganisms. ModelSEED also offers tools for model validation and analysis. Pathway Tools is a versatile software platform that allows for the reconstruction, visualization, and analysis of genome-scale metabolic models. It provides a user-friendly interface and supports the integration of various data sources to create detailed metabolic reconstructions. Pathway Tools is widely used in academic and industrial research for its comprehensive capabilities[78]. The RAVEN (Reconstruction, Analysis, and Visualization of Metabolic Networks) Toolbox is a MATLAB-based tool for metabolic model reconstruction and analysis. It provides a flexible framework for creating models, performing flux balance analysis (FBA), and visualizing metabolic networks. RAVEN is particularly useful for researchers who are comfortable with MATLAB programming. CarveMe[79] focuses on creating context-specific metabolic models tailored to specific organisms, conditions, or environments. It considers available genomic and experimental data, enabling researchers to study the metabolic capabilities of an organism in a particular context.

CarveMe automates the process of filling gaps in metabolic models, which typically occur due to missing or incomplete information in genome annotations. It suggests potential reactions that can fill these gaps, enhancing the model's completeness. Table 2.2 represent a complete list of available tools for model reconstruction.

Table 2.2: Summary of available tools for model reconstruction.

Tool	Language	Database for metabolic reactions	Gap-filling	Eukaryote reconstruction	Reference
COBRA	MATLAB	Any reaction database	Yes	Yes	[16]
COBRApy	Python	Any reaction database	Yes	Yes	[75]
COBRA.jl	Julia	Any reaction database	Yes	Yes	[8]
Model SEED	Web, Python	In-house reaction database	Yes	Yes (only plants)	[76]
Pathway Tools	Python	Pathway/Genome Database (PGDB), MetaCyc	Yes	Yes	[15]
RAVEN	MATLAB	KEGG, MetaCyc	Yes	Yes	[80]
AuReMe	Python	KEGG, BiGG Models, MetaCyc	Yes	Yes	[77]
CarveMe	Python	BiGG Models	Yes	No	[79]
KBase	Web	KEGG, BiGG Models, MetaCyc	Yes	Yes (only plants)	[11]
AutoKEGGRec	MATLAB	KEGG	No	No	[81]
Mackinac	Python	Model SEED	No	Yes (only plants)	[82]
FAME	Python	KEGG	No	No	[83]

2.3.3 Limitations of model reconstruction tools

Although numerous frameworks have been established for model reconstruction and data analysis, multiple challenges such as user-unfriendliness, difficulty to install, limited operating system compatibilities and lack of interpretability still exist. Various tools require their own input data format which might be non-standard format and might output data in a format that is incompatible with other frameworks[9, 3]. For example, Pathway Tools is a software environment that supports the creation and curation of GEMs as well as visualization of metabolic pathways in a model. However, its unique input data format (protein gene database; PGDB) and the fact that it is not a free software makes its utilization limited[78, 84]. Additionally, most of the available methodologies have application for prokaryotic organisms. CarveMe is a tool designed by to create GEMs for microbial communities and the created models are ready to use for FBA analysis. The implementation of its own gap-filling algorithm allows this tool to prioritize the incorporation of reactions into the model with higher genetic evidence, however its specificity for prokaryotic organisms and limited number of template models for gap-filling, limit its use in practice. Furthermore, ModelSEED is another popular web resource for genome-scale reconstruction and analysis with useful features like pathway visualization an integration with other available tools. Nonetheless, it can only be used for prokaryotic microorganisms and plants since its content is primarily derived from prokaryotic microorganisms and plants[76, 10].

2.3.4 Gap-filling in metabolic models

The accuracy of GEMs' predictions and functionality largely depends on the quality and completeness of genome annotations[11, 62]. Since the genome annotations are not always complete, generated GEMs contain inaccuracies, missing information, or gaps in knowledge that can affect further simulations of metabolic activities. The incompleteness of genome annotation can either result from the lack of knowledge about the genome of the organism or the due to the errors from genome annotation tools[85, 86]. All the annotation tools rely on public databases, and since some of the sequences in these databases are not experimentally characterized, accurate annotation becomes challenging. Moreover, different tools use different annotation approaches, including sequence-based, structure-based, transcription-based, and pangenome-based annotations, which can all yield diverse outcomes and perhaps introduce errors[87]. Each annotation approach brings unique strengths and limitations to the characterization of genes and proteins. Sequence-based annotation, relying on the genetic code, may miss non-sequence-related functional information. Structure-based annotation, leveraging three-dimensional protein structures, depends on the availability and quality of experimental data. Transcription-based annotation, derived from RNA expression data, offers insights into gene regulation but is context-dependent. Pangenome-based annotation, considering the collective gene repertoire of a species, captures variability but is influenced by criteria for inclusion/exclusion and genome availability[87, 88]. The variability in annotations arises from differences in data quality, algorithmic approaches, and the inherent complexity of biological systems. Challenges include data incompleteness, algorithmic nuances, context-dependent expression levels, and the dynamic nature of biological functions[87, 89]. Other errors in annotation algorithms generally and most related to relativity of the alignment threshold, namely low sensitivity, and specificity, which leads to incorrect or inaccurate annotation[85, 90]. Gaps in the GEMs are the absence of reactions that consume or produce a metabolite, which lead to dead-end metabolites and blocked reactions in the model. As a result, the gap-filling process is necessary to improve the analysis and interpretation of GEMs[91, 89]. Gap-filling involves identifying and addressing gaps in the metabolic network by incorporating candidate reactions from biochemical databases that could complete the metabolic network. Gap-filling can be done manually which is a tedious and time-consuming task or via the current automated tools[6, 15].

2.3.5 Current gap-filling tools

The past two decades have witnessed the development of several automated gap-filling algorithms. These algorithms serve the crucial purpose of completing or refining metabolic network models for various organisms. Among this array of algorithms, we can categorize them into two broad classes based on their underlying principles and methodologies. The first category comprises optimization-based algorithms, such as BoostGAPFILL[92], OptFill[93], GLOBALFIT[29], COBRApy[75], and Model SEED[76]. These algorithms utilize Mixed Integer-Linear Programming (MILP) to identify the minimum number of reactions needed to resolve gaps in a metabolic model. MILP is an extension of linear programming that allows for the inclusion of integer variables in the optimization problem. In the context of gap-filling, MILP is used to identify a minimum set of reactions (integer variables) to add to a metabolic network in order to resolve gaps, typically gaps in biomass production. The objective is to minimize the number of added reactions while ensuring that the modified network is capable of producing the necessary biomass. The inclusion of integer variables allows for discrete decisions, such as whether to include or exclude a particular reaction, making MILP suitable for problems involving binary choices. In contrast, the second category uses likelihood-based algorithms, as showcased by MIRAGE[94] and Pathway Tools[15]. These algorithms employ a different approach, emphasizing the probabilistic aspect of gap-filling. They consider a range of criteria, including genomic evidence and context, to determine the necessary reactions for gap-filling. Notably, this approach is applicable to both prokaryotic and eukaryotic organisms, underlining its versatility. Distinguished from these categories is Meneco, a graph-based gap-filling method. Meneco takes a unique approach by examining the topological structure of the metabolic model network. Meneco identify reactions for gap-filling when all the required reactants are already present within the network. This method is particularly valuable for comprehending the spatial relationships and connections within the metabolic network[95]. The classification of the tools is also determined by their specialization for either prokaryotic or eukaryotic organisms. Understanding the distinction between prokaryotes and eukaryotes is paramount in the realm of gap-filling software tools due to the profound biological differences between these two organism types. Prokaryotes, encompassing bacteria and archaea, and eukaryotes, characterized by organisms with a true nucleus, exhibit variations in metabolic pathways and functionalities. Consequently, gap-filling tools must account for these biological distinctions to ensure the accurate prediction and completion of missing reactions within the context of the specific organism being studied. Moreover, the importance of considering the prokaryote-eukaryote dichotomy extends to the optimization and specialization of gap-filling tools. Each organism type may require unique algorithms and databases for effective gap-filling, considering their diverse metabolic networks. Tailoring tools to the specific needs of prokaryotic or eukaryotic organisms enhances their efficiency and relevance, contributing to the accuracy of completed metabolic models. Since gap-filling is a major challenge in metabolic modelling, a range of tools have been developed to enhance the accuracy and completeness of metabolic network models. The summary of the current gap-filling algorithms is presented in Table 2.3. Organism specificity, user-unfriendliness, and the inability to

Table 2.3: Summary of the current gap-filling algorithms

Tool	**Universal reference database**	**Language**	**Target Organism**	**Gap-filling approach**	**Reference**
Reconstructor	Model SEED	Python	Prokaryotes	pFBA based	[96]
OptFill	any database	Python	Only small models	Optimization-based	[93]
AuReMe	KEGG, BiGG Models, MetaCyc	Docker-image	Prokaryotes & eukaryotes	Topology-based	[77]
CarveMe	BiGG	Python	Prokaryotes	Likelihood-based	[79]
BoostGAPFILL	BiGG	MATLAB	Prokaryotes	Optimization-based	[92]
Meneco	MetaCyc	Python	Prokaryotes & eukaryotes	Topology-based	[95]
Pathway Tools	MetaCyc	Stand alone	Prokaryotes & eukaryotes	Likelihood-based	[15]
GLOBALFIT	BiGG	R	Prokaryotes	Optimization-based	[29]
FASTGAPFILL	KEGG	MATLAB	Prokaryotes & eukaryotes	Optimization-based	[9]
COBRApy	any database	Python	Prokaryotes & eukaryotes	Optimization-based	[75]
MIRAGE	KEGG	MATLAB	Prokaryotes	Likelihood-based	[94]
Model SEED	Its own database	Python/web-based	Prokaryotes & plants	Optimization-based	[76]

find gaps for models beyond their provided examples are some of the main limitations of the current tools.

Chapter 3

Genome-Scale Metabolic Model Reconstruction and Validation for Thraustochytrid Strain T18

3.1 Introduction

A genome-scale metabolic model of thraustochytrid strain T18 cells was developed to predict cellular response to variation in carbon input (glucose/xylose) in the culture media as well as investigate the formation of docosahexaenoic acid (DHA), seen as one of the major T18 cellular products. Genomic and proteomic data provided by Mara, combined with information from previously published models, enabled the construction of a stoichiometry flux model with 2252 reactions (including transport and exchange reactions) and 1952 metabolites. Flux balance analysis (FBA) identified the fundamental difference between growth on glucose and xylose to be the utilization of cofactors such as NADPH and NADH. Following extensive gap-filling, growth on xylose resulted in an increased demand for both NADH and NADPH, which caused pronounced changes to the overall flux distribution. Simulated knock-out experiments identified 148 reactions essential for growth on xylose that were not required for growth on glucose. If the gap-filling process is correct, our results suggest that mass balance constraints identified through modelling may make it challenging to address increasing NADH/NADPH demands in one or two targeted genetic modification. However, it must be noted that major gaps in the genome annotation data made reconstruction far more challenging than initially expected. Despite our best efforts to identify and fill model gaps through iterative rounds of model refinement and manual curation, we found significant evidence of multi-enzyme gaps, where true pathway structure could only be guessed. Approximately 40 percent of metabolites included in the model lacked consumption or production reactions, making them effective “dead-ends”. Furthermore, no amount of gap-filling was sufficient to simulate the production of fatty acids such as DHA, throwing into question the completeness of the underlying data.

3.2 Draft metabolic network

Model reconstruction was initiated by extracting 970 unique enzymes (referred to by their enzyme commission, EC, numbers) from both old and new genomic and proteomic data. Extracted EC numbers were converted into metabolic reactions using the BiGG modelling database. A number of databases were considered, but the use of BiGG facilitated comparison to previously published genome-scale metabolic networks using a single resource with standard nomenclature[97]. Combining all putative reactions linked to the identified EC numbers (and considering that a singe enzyme is capable of catalyzing multiple reactions) resulted in a draft network consisted of 1812 intracellular reactions and 1557 metabolites, along with 253 transport and exchange reaction necessary to model metabolite flux into and out of the cell. A biomass reaction (approximating the overall requirements for amino acids, nucleotides, carbohydrates, lipids, etc.) was formulated based on previous modelling work on related organisms of *Schizochytrium limacinum (SR21)*[98], *Aurantiochytrium T66*[99] and *Oblongichytrium sp.RT2316-13*[100, 70]. Designing a series of experiments, such as those based on 13C-glucose, and subsequently utilizing 13C NMR spectroscopy can be advantageous for gaining insights into various metabolic pathways of T18.

3.3 Gap-filling of the metabolic model

3.3.1 Growth on glucose

Primary gap-filling was based on the assumption that the metabolic model should be capable of simulating the production of all biomass precursors from the components found in glucose-containing media that is capable of supporting cell growth. In short, this was done by setting each of biomass components as the objective function of model simulation in-turn. Setting each of the biomass model metabolites as the objective of the model simulation, mathematically, involves performing simulations with the goal of maximizing the flux through each individual biomass metabolite. This served to identify biomass precursors such as tryptophan, lysine, L-galactose, and cholesterol as so-called “dead-end” metabolites (nodes in the overall metabolic model with either no consumption or production pathways leading to them). Tracing upstream reactions and comparing related models suggested the inclusion of histidinol-phosphatase (EC: 3.1.3.15), GDP mannose-phosphorylase (EC: 2.7.7.22), GDP-L-galactose phosphorylase (EC: 2.7.7.69), diaminopimelate de-hydrogenase (EC: 1.4.1.16) and phosphatidylcholine synthase (EC: 2.7.8.24). As illustrated in Figure 3.1, each of the gaps above were relatively straightforward to identify as only a single enzyme was needed to restore function to an otherwise complete pathway. However, this process was not sufficient for simulating growth. A survey of all published flux models on BiGG suggested a number of highly conserved enzymes that were missing from our draft model, such as, sphingomyelin synthase (EC: 2.7.8.27) and sterol O-acyltransferase (EC: 2.3.1.26). In total, 27 additional reactions were needed for the model to be able to produce biomass in the presence of glucose as the carbon source in culture media. Unfortunately, the accuracy of multi-enzyme gap-filling cannot be estimated at this time.

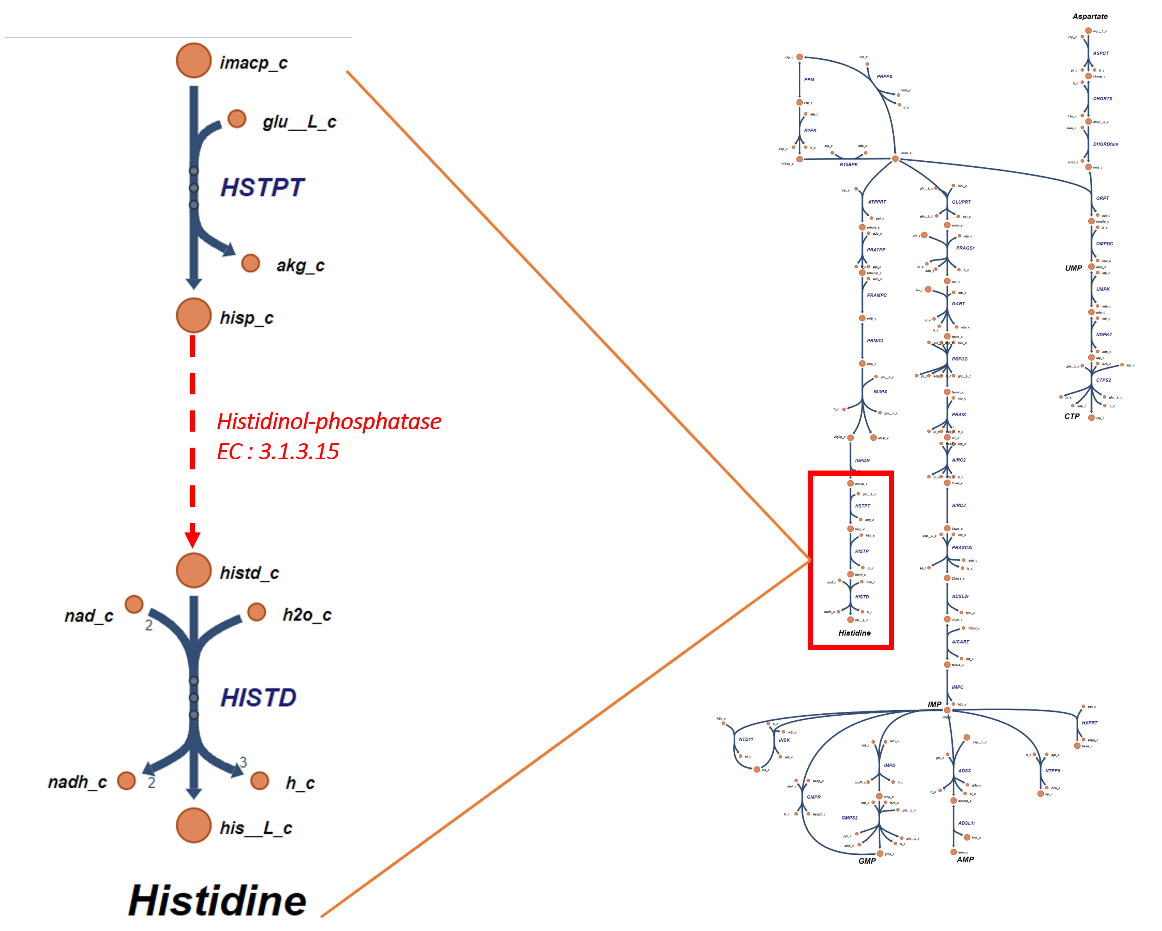

Figure 3.1: A gap in the histidine metabolism pathway of thraustochytrid strain T18. The gap between L-Histidinol phosphate (hisp-c) and L-Histidinol (histd-c) is caused by a lack of supporting information from annotated data (EC: 3.1.3.15), which breaks the network's connectivity and inhibits histidine synthesis.

3.3.2 Growth on xylose

The same process was repeated for growth on xylose as the primary carbon source. As with glucose, xylose initially found to be insufficient to support growth. The only firm evidence of enzymes for xylose metabolism were xylose isomerase (EC: 5.3.1.5), converting D-xylose to D-xylulose, as well as D-xylose reductase (EC: 1.1.1.307) and xylitol oxidase (EC: 1.1.3.41) in the pentose and glucuronate interconversions pathway. However, no firm downstream reactions could be identified. A review of existing models was conducted to determine common pathways for xylose metabolism (summarized in Figure 3.2). In prokaryotes, D-xylose conversion was found to occur via direct isomerization to D-xylulose catalysed by xylose isomerase, and in eukaryotes, D-xylose is first reduced to xylitol by xylose reductase, and the resulting xylitol is subsequently oxidized to D-xylulose by xylitol dehydrogenase. These findings were further supported by a review of the literature Rodriguez et al.,[101] and Wittmann et al.,[102]. Although no direct genomic evidence was found for the conversion of xylitol to D-xylulose in T18 annotated data, the presence of several incomplete EC numbers in the 1.1.1.- format and enzyme conservation in similar models suggested the addition of D-xylulose reductase (EC: 1.1.1.9) and L-iditol 2-dehydrogenase (EC: 1.1.1.14). A schematic of xylose entry into TCA cycle following the aforementioned additions is presented in Figure 3.3. Despite carbon entry into the TCA cycle, major gaps remained in pathways corresponding to fatty acid biosynthesis (steroid biosynthesis, glycerophospholipid, and biosynthesis of unsaturated fatty acids). Assuming that T18 cells are capable of growing on xylose as a primary carbon source and that their lipid profile remains consistent with growth on glucose, 16 more enzymes were added to the draft model based on conservation to similar models. The engineered T18 was experimentally confirmed to exhibit diauxic growth on glucose/xylose media. However, T18 did not demonstrate growth in media with only xylose as the sole carbon source[25]. The growth response to xylose as the primary carbon source was of interest to Mara and thereby one of our modeling objectives.

3.3.3 DHA production

Given the importance of omega-3 fatty acid production to the T18 cell culture process, a third round of gap-filling was performed for docosahexaenoic acid (DHA) biosynthesis. Despite evidence of enzymes involved in the elongation–desaturation pathway (ECs: 1.1.1.211, 1.1.1.35, 1.3.1.38, 2.3.1.16, 3.1.2.2, 3.1.2.22, 4.2.1.17) as

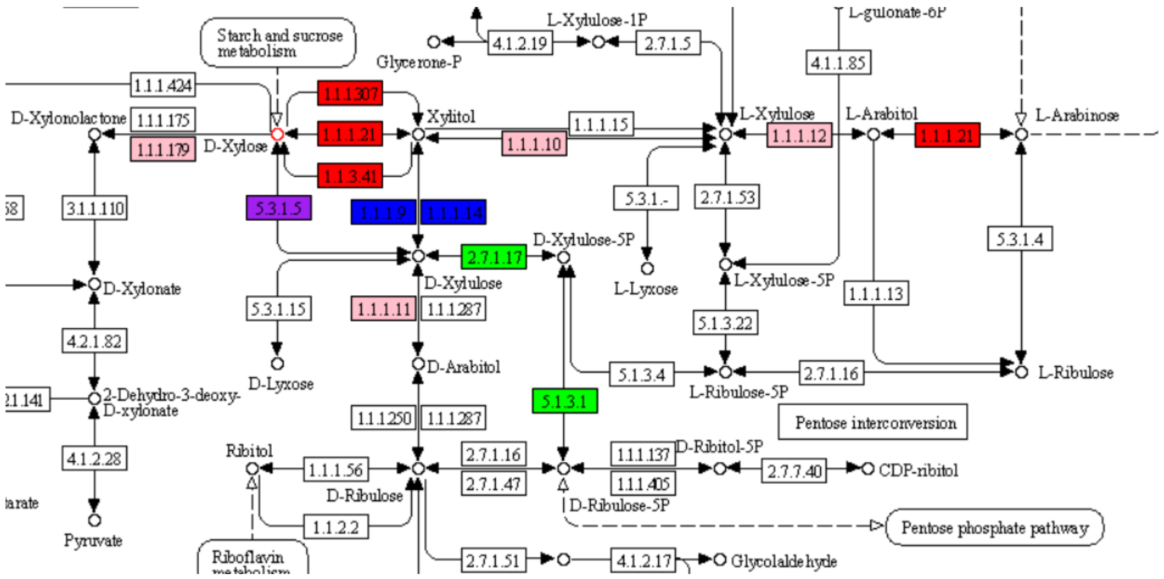

Figure 3.2: EC numbers related to D-xylose in the pentose and glucuronate interconversions pathway for a variety of bacteria, yeast, and mammalian genome-scale metabolic models (BiGG database), as well as similarly previously reported thraustochytrids models of *Schizochytrium limacinum (SR21)* and *Aurantiochytrium T66*. The percentage of existence of each of the EC numbers in the xylose pathway for the models is represented by different colours. All of the studied models contain xylose isomerase (EC: 5.3.1.5) (purple). Green indicates enzymes present in 90 percent of the models (ECs: 2.7.1.17, 5.1.3.1). Red indicates enzymes present in 60 percent of the models (ECs: 1.1.1.307, 1.1.3.41). Blue indicates EC numbers found in 60 percent of the models (as well as similar organisms to T18) (ECs: 1.1.1.9, 1.1.1.14). Pink indicates enzymes precent of the models (ECs: 1.1.1.179, 1.1.1.11, 1.1.1.10, 1.1.1.12).

well as the polyketide synthase (PKS) pathway (ECs: 2.3.1.179, 1.1.1.100, 1.3.1.9, among approximately 10 others), neither route could support flux distribution with DHA production used as the objective function for model simulation. Semi-automated analysis of flux maps and comparison of related models revealed multiple gaps in the elongation-desaturation pathway that required the addition of 14 more ECs revolving around sphingolipid, steroid, and glycerophospholipid biosynthesis (driven in part by partial identification of ECs in the 1.1.1.-, 3.6.1.-, 4.1.1.-, and 4.2.1.- ranges). Our semi-automated analysis primarily involves the utilization of pathway visualization tools, namely Pathview[103], ,Escher[104] and PyGraphviz[105]. To address metabolites that cannot be produced or consumed in the biomass reaction, we integrated

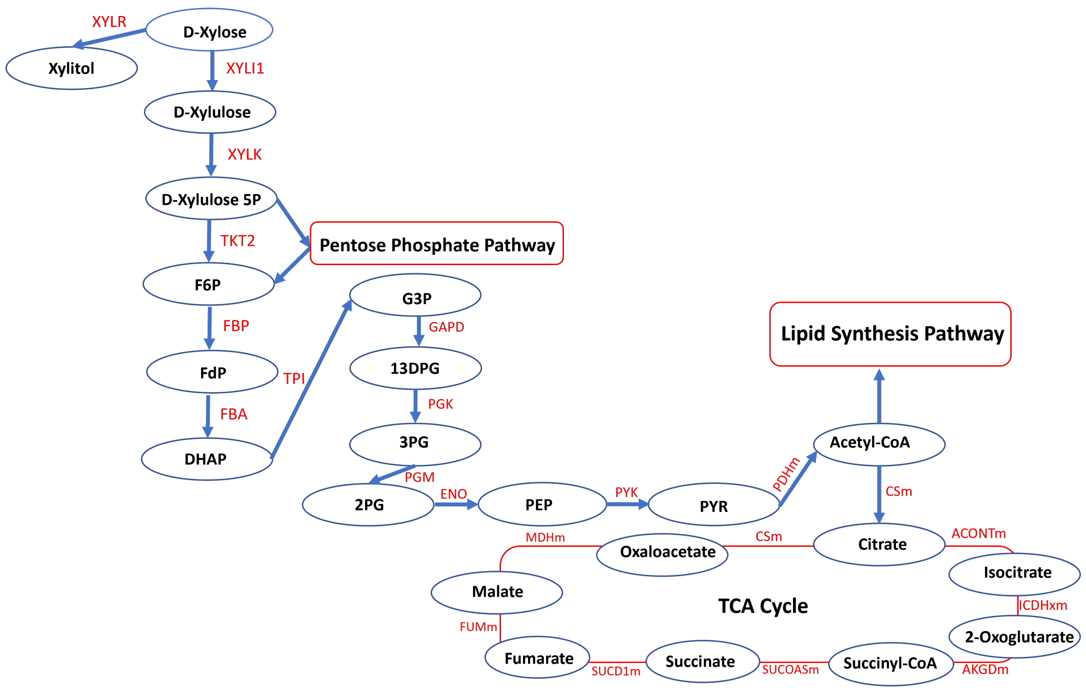

Figure 3.3: Putative xylose metabolism through glycolysis, the pentose phosphate pathway, and the TCA cycle. D-xylose isomerase (XYLI1) converts D-xylose to Dxylulose, and xylitol is produced as a by-product by xylose reductase (XYLR). Xylulokinase (XYLK) catalyses the conversion of D-xylulose to xylulose-5-phosphate. D-xylulose-5-phosphate, a pentose phosphate pathway intermediate, enters the pentose phosphate pathway, glycolysis, and TCA cycle, where it produces Acetyl-CoA.

the capabilities of COBRApy[75] and PyGraphviz. This integration enabled the generation of graphs, allowing us to trace the pathways of each metabolite and understand the relationships between our target metabolites and other reactants and products. Subsequently, we systematically examined each metabolite to identify dead-end metabolites, which are metabolites solely produced or consumed in the model, leading to blocked reactions. Following this identification, we filled the identified breaks and gaps with appropriate reactions sourced from the KEGG/BiGG databases. However, none of these additions were sufficient to enable DHA production. Analysis of the PKS pathway yielded similar results, with hydrolase (EC: 4.2.1.46) and isomerase (EC: 5.5.3.21) identified as the primary pathway gaps to be filled. However, these additions were also insufficient to enable DHA production. Failure to simulate DHA production suggests a generally broad level of uncertainty in the underlying genomic data around the enzymatic pathways used for general fatty acid biosynthesis that could not be resolved in the scope of this project. The annotated data does, however, appear to agree with the general literature that the biosynthesis of polyunsaturated fatty acids in thraustochytrids is primarily controlled by the action of a PKS enzymatic complex[100, 70].

3.4 Topological validation

In light of the gaps identified above, a general survey of model completeness was conducted on all 108 prokaryote and eukaryote models found in the BiGG database. Completeness was primarily judged based on the total number of dead-end metabolites as a fraction of the overall number of reactions. The overall percentage of blocked reactions in published models was found to vary from 0 to 75%, with a significant mode at 40% (full distribution presented Figure 3.4). At 39%, the total fraction of blocked metabolites for the T18 model fell slightly below the mode. These gaps are typically shows up in the less central pathways. The central pathways of metabolism, such as glycolysis or the citric acid cycle, are often well-studied and characterized. Additionally, central metabolic pathways tend to be more evolutionarily conserved across organisms. The conservation of these pathways may lead to more reliable information, whereas less central pathways may vary more between organisms, contributing to gaps in our model. Moreover, less central metabolism may involve more complex or specialized pathways that are harder to study and model accurately. This complexity could contribute to a higher likelihood of gaps. Similarly, the total number of metabolites aligns closely with the distribution mode, as depicted in Figure 3.5, while the total number of reactions slightly deviates below the mode, as illustrated in Figure 3.6. Peaks in both figures signify the distribution patterns of prokaryote and eukaryote GEMs. Despite the presence of significant gaps in pathways such as fatty acid biosynthesis (detailed above), the generated T18 model was found to be quite comparable to published models in terms of general model topology (structure). Nevertheless, the fact that 39% of the metabolites in the model do not have either consumption or production reactions presents a significant limitation to the level of detail that can be extracted from the overall modelling effort, especially when combined with the specific gaps present in the xylose and fatty acid metabolism.

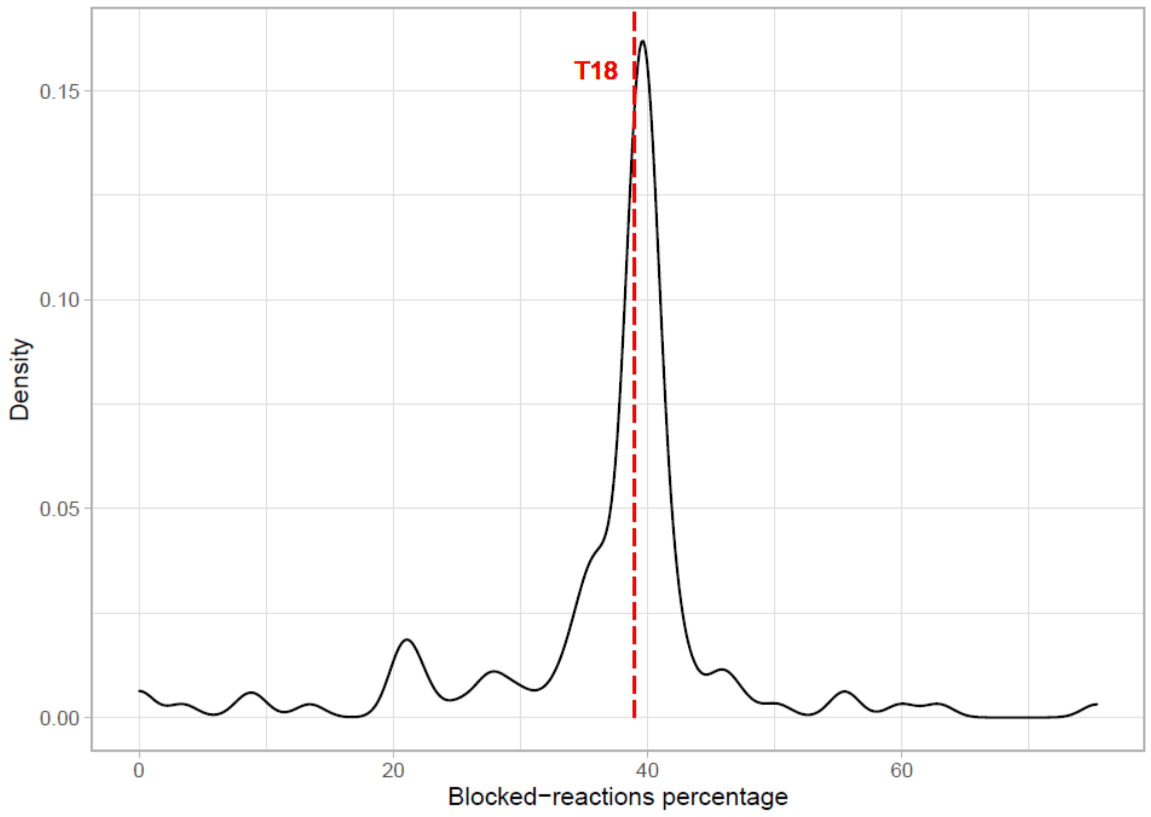

Figure 3.4: Distribution of percentage of blocked reactions for all 108 BiGG models in comparison to the T18 model.

3.5 Comparison of glucose and xylose metabolism

A basic comparison of growth on glucose as the primary carbon source vs. xylose was simulated by considering the total production of NADH, NADPH, ATP, Acetyl-CoA, and CO_2, as well as the consumption of O_2 per uptake flux of carbon. Although the

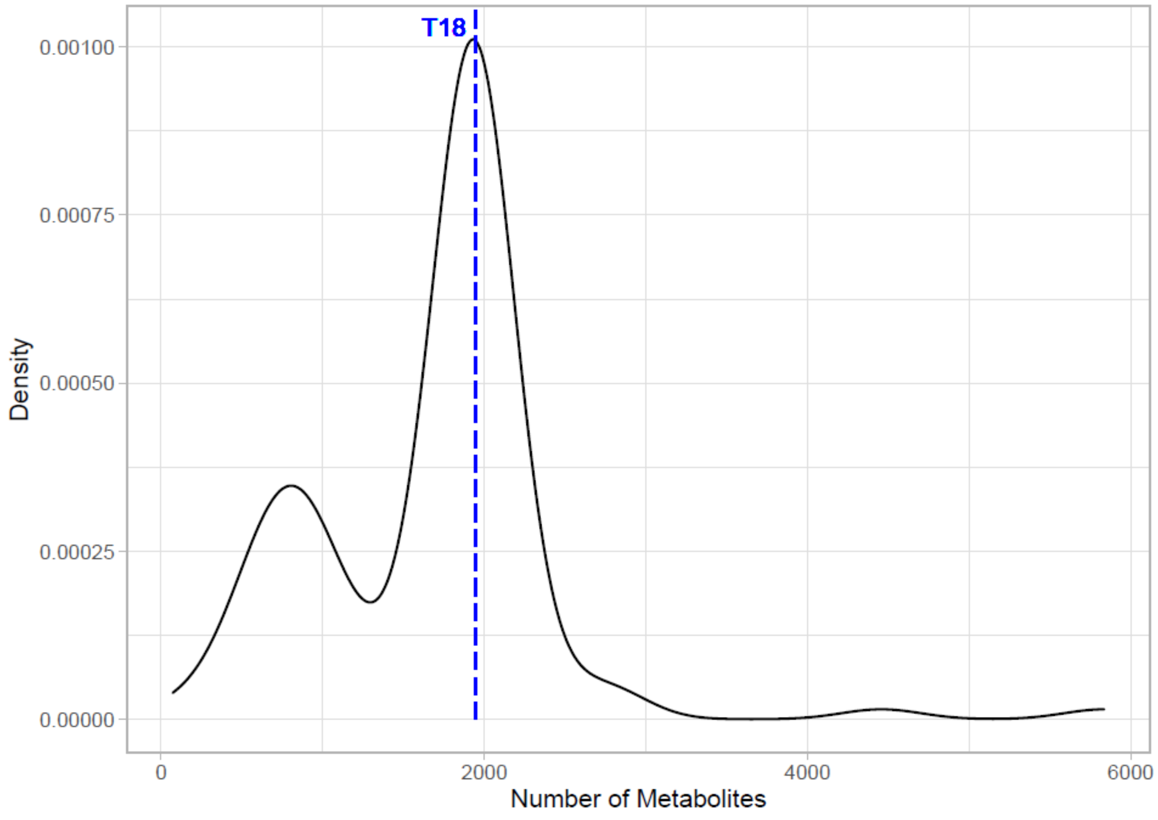

Figure 3.5: Distribution of total metabolite number for all 108 BiGG models in comparison to the T18 model.

overall metabolic model is premised on a balance around energetic species such as NADH, NADPH, and ATP, it is nonetheless possible to tally all production reactions and all consumption reactions to understand the difference that carbon source plays on the overall flux distribution. When these values were evaluated for xylose and glucose in the culture media, it was revealed that this ratio stayed relatively constant for ATP, Acetyl-CoA, O_2, and CO_2. On the contrary, the ratios for total NADH production per uptake flux of the carbon source and total NADPH production per uptake flux of carbon source were 5 percent and 6 percent higher for xylose. Although the increased need for NADH/NADPH is not particularly surprising given the different degrees of reduction of xylose and glucose, a couple of basic observations do stand out. First, both NADH and NADPH production increased rather

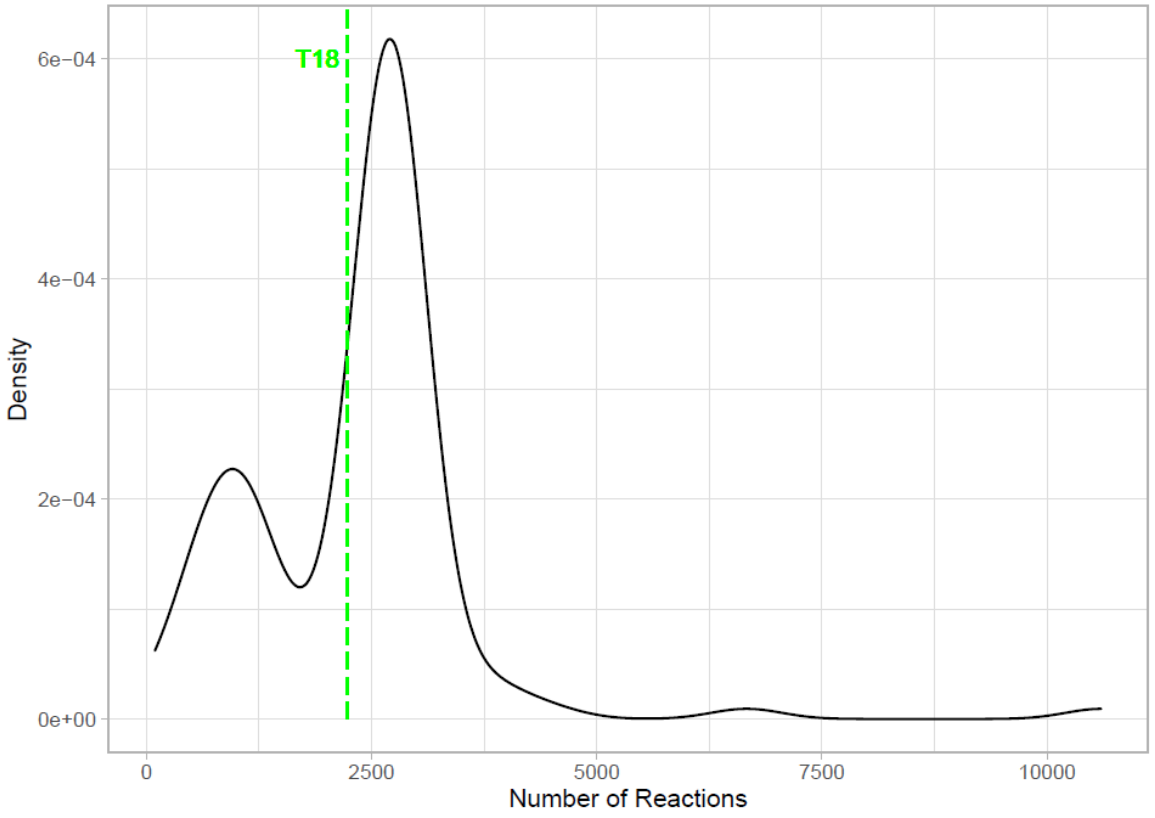

Figure 3.6: Distribution of total reaction number for all 108 BiGG models in comparison to the T18 model.

than only one. This is significant because the overall flux model did not include a conversion reaction between NADH and NADPH, meaning that mass constraints do place a demand on both. Furthermore, even with relatively loose constraints, referring to constraints assigned solely through a comparison of the model with other similar models, both NADH and NADPH fluxes were found to increase across a large number of pathways. Simulated knock-out analysis revealed 501 essential internal reactions (22.2 percent of total reactions) that were necessary for the consumption of either carbon source. Growth on xylose alone required an additional 148 reactions (6.5 percent of all reactions), whereas growth on glucose alone required just 24 additional reactions (1 percent of total reactions). Notably, growth on xylose required an increased uptake of metabolites such as methionine and malate, although it is unclear on whether these were symptoms of the model dealing with the increased demand for NADH/NADPH. Taken together, these results suggest that mass balance constraints may make it challenging to address increasing NADH/NADPH demands in one or two targeted genetic modification; however, more detailed analysis is constrained by overall model uncertainty.

Chapter 4

Systematic Assessment of Current Gap-filling Tools

4.1 Introduction

Genome-scale metabolic model (GEM) reconstruction for an organism typically begins with annotated genomic data for the organism of interest[6, 7]. However, genomic data may not always be complete, and genome annotation tools may not be able to assign proper functional roles to some genes[11, 91, 106]. Even for *E. coli*, one the most popular model organisms, around 30 percent of the protein encoding genes lack a functional role. All the annotation tools rely on public databases, and since some of the sequences in these databases are not experimentally characterized, accurate annotation becomes challenging. Moreover, different tools use different annotation approaches, including as sequence-based, structure-based, transcription-based, and pangenome-based annotations, which can all yield diverse outcomes and may introduce errors. Other errors in annotation algorithms generally and most related to relativity of the alignment threshold, namely low sensitivity, and specificity, which leads to incorrect or inaccurate annotation. Consequently, the transition from genomic data to enzymes and then to reactions often results in the presence of several gaps in GEMs, referring to the absence of reactions that either consume or produce a metabolite, which leads to dead-end metabolites and blocked reactions in the model[107, 15].

As a result, a gap-filling process is necessary to enhance the analysis and interpretation of GEMs. Gap-filling involves identifying and addressing gaps in the metabolic network by incorporating candidate reactions from biochemical databases that can complete the metabolic network. This process can be performed manually, which is both time-consuming and tedious, or can be performed with the aid of automated tools[10, 108, 109]. Several automated gap-filling algorithms have been developed for various organisms over the past two decades. Some such as BoostGAPFILL[92], OptFill[93] , GLOBALFIT[29][109], COBRApy[75], and Model SEED[76] are optimization-based algorithms that solve Mixed Integer Linear Programming (MILP) problem with the goal of determining the minimum number of reactions required to resolve gaps in the model. The tools resolve the gaps by identifying and adding the most essential reactions to the model, which could involve incorporating missing biochemical processes, enzyme-catalyzed reactions, or metabolite conversions. The ultimate aim is to ensure that the model is accurate and functionally complete. In contrast, others such as MIRAGE[94], Pathway Tools[15] rely on likelihood-based algorithms that consider a set of criteria such as genomic evidence to identify the necessary reactions to fill a potential gap for both prokaryotes and eukaryotes. On the other hand, there are topology-based gap-filling approaches. Topological gap-filling approaches in metabolic modeling involve strategies that leverage the network structure or topology of a metabolic model to identify and fill gaps in its reactions. These methods capitalize on the connectivity and relationships between metabolites and reactions within the network. One common approach is to prioritize the addition of reactions that bridge gaps in the metabolic network, ensuring the formation of a connected and functional pathway. Topological analysis, such as assessing the betweenness centrality of metabolites or reactions, plays a crucial role in these approaches, guiding the identification of key missing links. By exploiting the inherent organization of the metabolic network, topological gap-filling approaches aim to enhance the accuracy and biological relevance of completed metabolic models. Meneco is the only available topology-based gap-filling method that addresses gaps in the model by examining the topology of the model network. It identifies and suggests reactions for gap-filling when all required reactants are present within the network. In other words, Its main emphasis lies in the identification of reactions or sequences of reactions that disrupt the continuity of metabolites from the culture media (seeds) to the biomass reaction (targets)[95].

These tools have the capability to execute a range of essential analyses, including flux balance analysis, and demonstrate practicality across a wide array of organisms. However, certain limitations, such as organism-specific applicability, lack of user-friendliness, and the inability to identify gaps in models that extend beyond their pre-established examples, have restricted their broader utility.

A systematic approach to assess the performance of gap-filling has not been developed thus far[10]. Due to the fact that gap-filling is still one of the main challenges in genome-scale metabolic modelling, a practical assessment can provide valuable insights to potential users, helping them determine which tool or combination of tools can effectively be used to address gap-filling process for their GEM. In this study, we developed an evaluation framework to assess the performance of available tools, with the purpose of identifying the optimal gap-filling tool to fill the gaps in our model. Among the various tools that were investigated, we concentrated on the most widely used tools, ultimately selecting COBRApy, CarveMe, and Meneco for a comprehensive evaluation of their effectiveness in gap-filling. It is crucial to emphasize that, in this chapter of our study, the choice of tools for comparison was based solely on their popularity, without taking into account any additional factors. COBRApy is one of the most widely utilized frameworks in the field of metabolic modeling. It is well-known for its ability to carry out fundamental analyses like Flux Balance Analysis (FBA), Flux Variability Analysis (FVA), and Dynamic Flux Balance Analysis (dFBA), with over 1000 citations. In addition, It is a useful option for modeling various organisms[75]. Next, CarveMe, with 387 citations, offers a wide application for microbial species and communities. Both COBRApy and CarveMe employ an optimization-based methodology for gap-filling[79]. In contrast, Meneco was chosen because of its unique approach of gap identification. It is the only topology-based gap-filling method that is currently available, and it fills in model's gaps by looking at the model network's topology[95].

4.2 Materials and methods

Our systematic assessment approach includes both qualitative and quantitative investigations for each of the 3 tools including COBRApy, CarveMe, and Meneco.

4.2.1 Qualitative analysis

For our qualitative analysis, we first established a set of criteria to compare and evaluate the performance of each tool, as shown in Table 1. We continued our qualitative evaluation by building a small model to fully comprehend the stated concept behind each tool's approach, test each tool more quickly, and debug and troubleshoot more easily. Evaluating the performance of tools with larger and complex models can be challenging and may make it more difficult to identify the source of errors or problems. Therefore,we generated a small model containing only 5 reactions, 2 exchange reactions and 3 internal reactions as shown in Figure 4.1 to test the COBRApy, CarveMe, and Meneco tools performance. This process involves removing the reaction that converts metabolite B to C (a trivial gap), adding that single reaction to the universe model, and investigating whether each tool could successfully identify the deleted reaction to fill this gap.

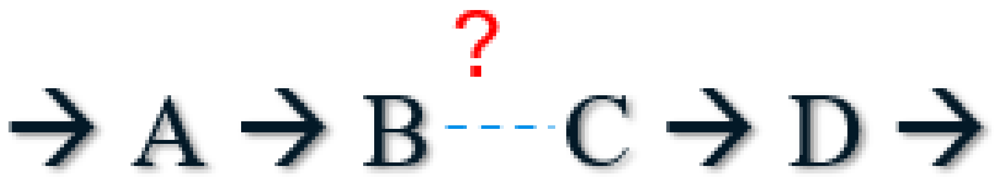

Figure 4.1: An example of a small toy model comprises 5 reactions, including 2 exchange reactions and 3 internal reactions, featuring a straightforward gap (involving the conversion of metabolite B to C). This small model is utilized to assess the performance of COBRApy, Meneco, and CarveMe.

4.2.2 Quantitative analysis

The more the number of gaps in a model, the more challenging the gap-filling process can become, and there is a higher likelihood of errors when using automated gap-filling tools. In order to evaluate the performance of each of the tools, we considered a straightforward analysis in a consistent condition to minimize the possibility of any errors or problems. Our approach generally involved creation of artificial gaps/inconsistencies in a model and evaluate each tool's ability in identifying these generated gaps. To have same GEMs as input for all the three tools as well as a diverse set of GEMs, we selected BiGG database as our case study. BiGG is a comprehensive source containing 108 manually curated genome-scale metabolic models, including both prokaryotic and eukaryotic models[87]. We began by identifying the essential reactions for growth in each model. Essential reactions are those reactions that their removal results in zero growth in FBA. The objective function for identifying essential reactions in all of the models was the maximization of the biomass reaction (organism growth). Next, we introduced artificial gaps by randomly removing one essential reaction from each model. The random removal of reactions was done without repetition. Subsequently, each gap model along with a trivial universe, were used as the main inputs for each of the three tools and their performance was assessed. The trivial universe model, the required reference database for gap-filling, was generated from the removed reaction (the artificial gap reaction) in each of the tested BiGG models and only consists of that single reaction. This approach ensures the exclusion of any potential errors coming from other reactions in a huge universe database, which allow us to focus only on evaluating each tool's performance. For COBRApy we only needed to provide the gap model and the trivial universe whereas for Meneco other inputs including a seed model (e.g., organism culture media), a target model (e.g., biomass reaction), rather than the gap model and the trivial universe model were required. Moreover, CarveMe only required the gap model as it utilizes its own universal model for gap-filling. To be able to compare the performance of each of the tools in identifying gap reactions in each model, we defined variables such as model size (including the total number of reactions and metabolites), the number of model compartments, and model phylogenetic domain, which encompass eukaryotes and prokaryotes, as our variables for comparison of results.

4.2.3 Effect of multiple artificial gaps in the models on tools performance

Given that GEMs typically contain numerous missing reactions/gaps, our next step of the analysis involved generating multiple artificial gaps in each model to evaluate the performance of tools in a more realistic scenario. To accomplish this goal, we created random sets of removal of 1, 10, 50, and 100 percent from the essential reactions in each model. We then evaluated and compared the performance of each tool by calculating the average recovery percentage. The average recovery percentage was determined by dividing the number of reactions identified by each tool to fill the gaps in each model by the total number of reactions for each set in each model.

Table 4.1: Results for qualitative assessment, including each tool's specific features and our custom set of criteria

Category	**Criteria/features**	**COBRApy**	**Meneco**	**CarveMe**
Custom Test Criteria	Gap fill arbitrary models	Yes	Yes	Yes
Custom Test Criteria	Gap fill a trivial model	Yes	Yes	No
Custom Test Criteria	Compile its own example	Yes	Yes	Yes
Custom Test Criteria	Gap fill a trivial gap in an existing model	Yes	Yes	Yes
Tool Specification Features	Take a universe (T) or provide its own(P)	T	T	P
Tool Specification Features	Gap fill eukaryotes	Yes	Yes	No
Tool Specification Features	Gap fill in a specific culture media	No	Yes	Yes
Tool Specification Features	Independent of a specific database	Yes	Yes	No
Tool Specification Features	Trimming reactions of an universal database	No	No	No
Tool Specification Features	Output format	List of reactions	List of reactions	Gap-filled SBML model
Tool Specification Features	Compartment dependency	Yes	No	Yes

4.3 Results and discussion

4.3.1 Characterization of tools

We conducted a comprehensive assessment of three current gap-filling tools for genome-scale metabolic models, employing both qualitative and quantitative analyses. To achieve this objective, we initially defined a set of criteria for evaluating the performance of each tool. Detailed results for these criteria, including each tool's specific features and our custom set of criteria, are presented in Table 4.1. The specific features of the tools encompass characteristics that each tool claims to possess, such as the capability to fill gaps in eukaryotic models, the requirement for specific culture media, or compartment dependency, refers to the influence of the number of compartments. In contrast, custom test criteria denote uniquely defined and tailored evaluation metrics established in this section of our analysis to gauge the effectiveness and practicality of a tool. For instance, these criteria include the tool's capacity to fill gaps in any model irrespective of size or phylogenetic domain, or its proficiency in compiling and running its own examples. All the three tools passed the custom test criteria almost successfully. However, CarveMe stood out mainly for being practical only for models using metabolite and reaction IDs from the BiGG database. CarveMe's gap fill function requires the use of CarveMe's own universe model, unlike COBRApy and Meneco, which can function with a broader range of universe models. Furthermore, COBRApy and Meneco can be practical for both prokaryotes and eukaryotes, while CarveMe is only suitable for prokaryotes. In addition, while COBRApy doesn't require specific culture media as input for gap-filling, Meneco and CarveMe rely on user-provided media information. One significant similarity and weakness among all three tools is that none of them employ any trimming criteria when selecting reactions from the universe model. Consequently, every reaction from a large database is considered equally important during the identification of the most suitable candidate reactions for gap-filling by each tool which increase the likelihood of errors and the inclusion of incorrect reaction to the GEM.

4.3.2 Quantitative gap-filling performance of COBRApy, Meneco, and CarveMe

For our systematic assessment, we chose the BiGG database as our case study, encompassing both prokaryotic and eukaryotic models. Our quantitative assessment started by examining of all models(containing these two categories) across the three selected tools. Additionally, we investigated the impact of model compartments and the phylogenetic domains of models on the performance of the tools. The number of prokaryotic and eukaryotic models that successfully worked, indicating that the gap-filling produced results without any computational error, is presented for each of the tools in Table 4.2. From the initially selected 108 BiGG models, 59 models were common among COBRApy, Meneco, and CarveMe, all of which belonged to the prokaryotic category. Nonetheless, there are differences among the tools as we can see in Table 4.2. Specifically, 21 out of 108 models, 39 out of 108 models, and 3 out of 108 models were applicable to COBRApy, Meneco, and CarveMe, respectively. It's important to highlight that the decrease in the total number of models for each tool stemmed from inaccuracies found in specific BiGG models. Some models lacked clarity in their objective functions or had an unclear count of essential reactions for growth, resulting in errors across all three tools when these inaccurate models were used as input. Consequently, the definition of tools working for the models is that the tool produced the output without encountering any errors.

Table 4.2: Number of models shared and distinct among COBRApy, Meneco, and CarveMe across all 108 BiGG models

Models similarity	**COBRApy**	**Meneco**	**CarveMe**
Same	59	59	59
Different	21	39	3

4.3.3 The relationship between models' compartments and phylogenetic domain and gap-filling performance of COBRApy, Meneco, and CarveMe

Compartments represent different sub-cellular locations within a cell, such as the cytoplasm, mitochondria, or the extracellular space. With the increase in the number of compartments in the GEM, the complexity of the model increases. More compartments mean more spatial and functional details, more reactions, and associated constraints which can impact the computational demands of simulation and overall performance of gap-filling tools. Locating the precise reaction to address a potential gap within any of the compartments can be a challenging. Therefore, our other analysis was the exploration of the relationship between the number of compartments in each model and the tools' performance. The details of the model compartments and their impact on gap-filling performance are presented in Table 4.3. The numbers in the table represent the count of models that were applicable in COBRApy, Meneco, and CarveMe, along with their corresponding compartments. As observed, the models were categorized into three main groups based on the number of compartments: 2, 3, and more than 4.

The majority of models with three compartments were the ones that worked in

Table 4.3: The relationship between the number of compartments and COBRApy, Meneco, and CarveMe gap-filling performance across all 108 tested models

Number of Compartments	**COBRApy**	**Meneco**	**CarveMe**
2	4	15	0
3	67	69	62
4+	10	18	0

all three tested tools. This implies that the tools' performance was better in identifying the removed essential reactions of models with three compartments, more so than models with a higher number of compartments. For COBRApy, Meneco, and CarveMe, the results across all 108 BiGG models were 67 out of 108, 69 out of 108, and 69 out of 108 models with three compartments, respectively. When considering more complex models with a greater number of compartments, COBRApy and Meneco successfully identified the removed essential reactions in 10 out of 108 and 18 out of 108 eukaryotic models, respectively. Moreover, zeros in CarveMe indicated an empty set of models for that category. It is important to note that the number and domain of tested models differed among the three tools. COBRApy was applicable for total of 81 out of 108 models, Meneco to total of 102 out of 108 models, encompassing both prokaryotes and eukaryotes, while CarveMe was applicable for total of 62 out of 108 models, exclusively including prokaryotic models. Consequently, the number of compartments may not be the sole determining factor influencing the tools' performance. Furthermore, models with four or more compartment represented eukaryotic organisms in our analysis. Table 4.4 provides the results for the influence of the models' phylogenetic domains on gap-filling performance. The majority of models fall into the prokaryotic category, with CarveMe identified as exclusively applicable to prokaryotes, encompassing 62 out of the total 108 models. Conversely, COBRApy and Meneco, with counts of 11 out of 108 and 17 out of 108 for eukaryotic models, respectively, demonstrate applicability for both eukaryotic and prokaryotic models. The numbers in the table signify the count of models that were applicable in

COBRApy, Meneco, and CarveMe, along with their respective phylogenetic domains.

Table 4.4: The relationship between organisms' phylogenetic domains of life and COBRApy, Meneco, and CarveMe gap-filling performance across all 108 tested models

Organisms Domain	**COBRApy**	**Meneco**	**CarveMe**
Prokaryote	69	81	62
Eukaryote	11	17	0

4.3.4 A novel systematic framework for evaluating the performance of gap-filling tools

The systematic approach comprises several steps. Firstly, we selected each model from our sample case study, consisting of 108 BiGG models. Secondly, we introduced a single gap (by removing one essential reaction) in each model, creating a reference database of size one with the omitted essential reaction. Lastly, we tested whether the tool could successfully identify this gap. This approach was repeated across all 108 BiGG models. A significant finding from our investigation, which consists of an artificial gap creation in all BiGG models and subsequently evaluation of COBRApy, Meneco, and CarveMe performance, is presented in Figure 4.2. Meneco was able to provide results for 102 out of 108 models, while COBRApy was able to fill 81 of the initial 108 models with artificial gaps. On the other hand, of the 108 models that we originally selected for our investigation, CarveMe was only able to provide results for 62 of them, and those models were solely prokaryotes. It should be noted that the reduction in the number of models for each of the tools resulted from inaccuracies in certain BiGG models. Instances of unclear objective functions or an indistinct number of essential reactions for growth were identified in some models, leading to errors in all three tools when such inaccurate models were selected as input. Therefore, in this context, it can be defined that COBRApy successfully produced output without encountering any errors for 81 out of the total 108 models. The comparison of results for COBRApy, Meneco, and CarveMe reveals that there is a mode at approximately

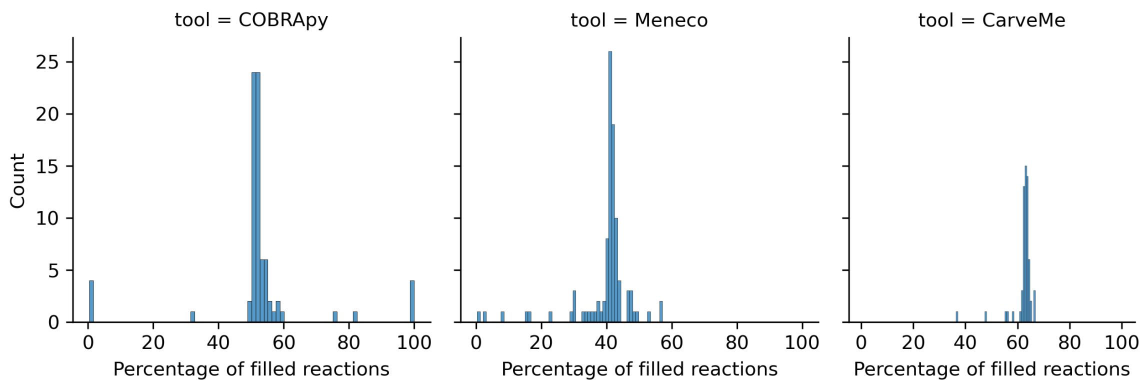

Figure 4.2: The distribution of the percentage of filled reactions for each of the 108 BiGG models that was tested by using COBRApy, Meneco, and CarveMe. First, the set of essential reactions in each model was identified. Second, one essential reaction was randomly removed from each of the models. Third, the gap model and trivial universe were utilized as inputs in each of the tools, calculating the percentage of reactions that each tool could identify to fill the gap in each model.

40-60 percent, indicating the percent of filled essential reactions in all of the tested models. It should be noted that CarveMe utilizes its own universe model, a trimmed version of the BiGG database specifically for prokaryotes. Most of the models in the BiGG database are prokaryotes, which explains the clustering of similarities among these prokaryotic models.

Although both COBRApy and Meneco are suitable for prokaryotic and eukaryotic models, COBRApy performs better in identifying the gaps for more eukaryotic models with being able to identify all the 100 percent of essential reactions removal for four models across all the tested models. In fact, for four models, COBRApy achieved a 100 percent success rate, implying that it successfully identified all tested removed essential reactions in these models, these models were eukaryotes. On the other hand, Meneco visualizes the model as a graph and evaluates the connectivity of metabolites. If an artificial gap does not break the network connectivity, Meneco may not be able to identify it. Furthermore, gap-filled models by Meneco are not guaranteed to be functional for advanced analyses such as Flux Balance Analysis (FBA) due to the fact that Meneco's gap-filling approach is not based on any mass balance or stoichiometry-based constraints.

In conclusion, our investigation into the performance of each tool reveals that, across all tested models, only approximately 50 percent of the removed essential reactions could be identified by the tools. Afterwards, we continued the evaluation of each tool's performance concerning model size, specifically the total number of reactions in a model, with the aim of determining whether it is a key factor influencing the tools' performance.

4.3.5 Effect of models' size on performance of tools

A big model with more cellular processes compartmentalization, pathways, number of reactions and metabolites can affect the performance of gap-filling tools and finding all the gaps in all involved metabolic pathways could be more challenging. Moreover, gap-filling of larger models can be more computationally expensive, meaning that the number of possible combinations of reactions that need to be considered increases exponentially with the size of the model. This combinatorial explosion can lead to a substantial increase in computational complexity. Furthermore, the search space for potential reactions to fill gaps becomes larger in larger models, making it computationally more challenging to explore and evaluate all possibilities. Thus, we investigated the effect of model size on the performance of the three tools. Although the results showed that the models' size does not have a huge effect on performance of the tools, there is significant pattern shared between the three tools represented in Figure 4.3. There is a cluster of models characterized by approximately 2700 total number of reactions. However, there is a distribution of points indicating that the percentage of filled reactions by the tools varies across models of different sizes. Consequently, the size of the models is not the sole factor influencing the performance of the tools.

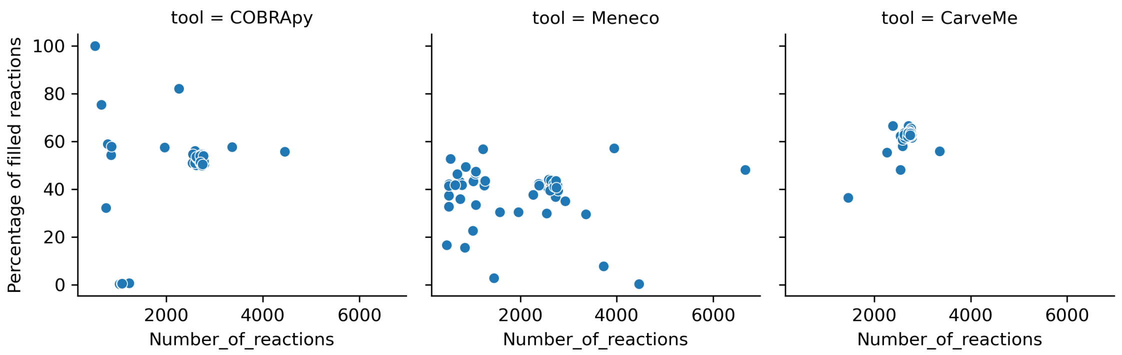

Figure 4.3: The effect of the model's size on the gap-filling performance of COBRApy, Meneco, and CarveMe.

4.4 Analysis of multiple gaps in each of the models

The idea of removing one reaction at a time to evaluate the performance of the tools was the simplest possible approach to commence our analysis. However, as we have observed, this approach could highlight a weakness in these tools, as it proves impractical for identifying almost half of the essential reactions removed across all the 108 tested models. In reality, most of GEMs contain several gaps and usually resolving all the gaps can be more challenging. Moreover, the performance of gap-filling tools can vary with the increasing number of gaps within a GEM. Therefore, we continued our analysis of each tool's performance with multiple artificial gaps creation to gain deeper insights into the advantages and limitations of the three tools gap-filling. Figure 4.4 provides an overview of how the tools' performance changes as the number of gaps in each model ranges from 1 to 100 percent. Commencing from 1 percent, which corresponds to one percent of the total essential reactions, either one reaction or multiple reactions (distinct from the previous section, which involved the removal of a single reaction), we progressed incrementally to 100 percent, signifying the removal of all essential reactions. In the case of COBRApy, the average recovery percentage remains relatively consistent over all applicable models, indicating that its performance is not significantly affected by the number of gaps in the models. In

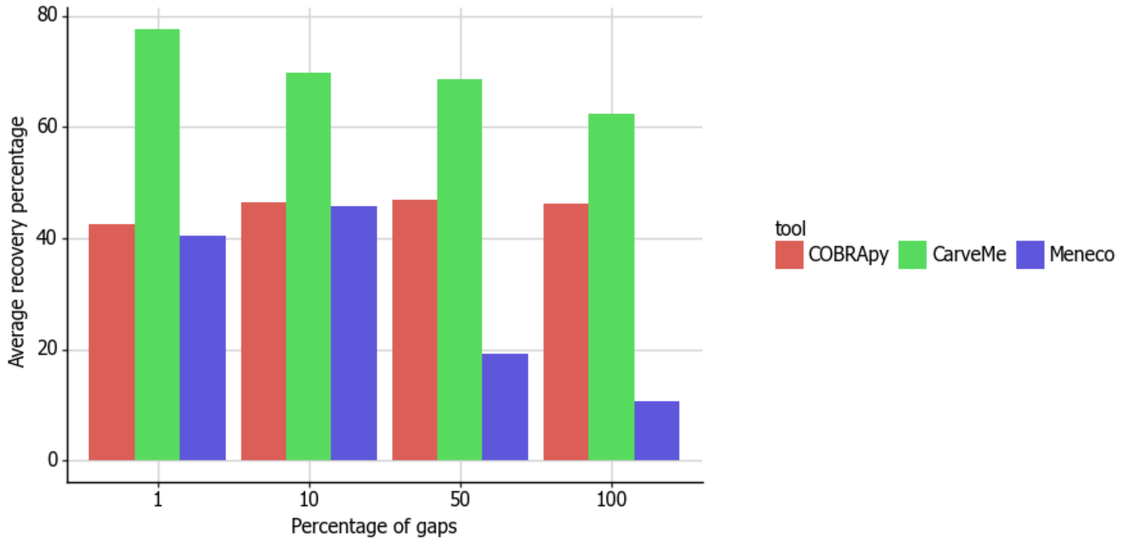

Figure 4.4: The comparison of tools performance with the increase in the number of gaps in each of the applicable models in the tools including 81 models for COBRApy, 102 models for Meneco, and 62 models for CarveMe.

contrast, for CarveMe there is a decreasing trend in the average recovery percentage with the increase in the number of gaps in the models. However, it should be noted that CarveMe performs better in comparison with the other two tools when dealing with prokaryotic models. It's important to emphasize that the assessment of CarveMe was limited to 62 prokaryotic models from the BiGG database, while COBRApy and Meneco were evaluated across a broader spectrum, with 81 and 102 models, respectively. For Meneco, there is noticeable decrease in performance as the number of gaps increases. Meneco may not necessarily suggest all the reactions that are essential for the growth of the organism if they are missing from the model. Its primary focus is on identifying reactions or series of reactions that break the connectivity of metabolites from the culture media (seeds) to the biomass reaction (targets).

4.5 Discussion

Gap-filling by COBRApy has not been systematically assessed to date. In the original COBRApy paper, all the analysis including gap-filling in COBRApy is limited to

GEMs of *Salmonella enterica Typhimurium LT2* and *Escherichia coli K-12 MG1655* as case studies. The gap-filling was only validated by testing one typical example. In this instance, the validation involved the removal of all reactions associated with D-Fructose-6-phosphate as an essential metabolite. The tool successfully suggested the missed reactions in response to this test scenario[75]. However, in the protocol provided by Norsigian et al.,[97] for the creation of multi-strain prokaryotes, they documented gap-filling failure by COBRApy. The report indicated that the tool did not yield reasonable or reliable results. Additionally, the GitHub issue section of the COBRApy package discusses optimization failure, a situation where the algorithm cannot find a solution, in the context of gap-filling for a plant genome-scale metabolic model using COBRApy. Similarly, CarveMe[79] has only been validated for specific case studies, including bacterial models such as *M. genitalium* and *R. solanacearum.* Overall, in four out of five models validated with the goal of growth on minimal media, gap-filling was successful. It should be noted that CarveMe considers a manually curated universal model generated from the BiGG database, which may limit the tool's validity for a range of organisms. In a recent study by Mendoza et al.,[110], CarveMe underwent assessment for 29 genome-scale metabolic models (GEMs) of bacterial organisms, encompassing species like *Lactobacillus plantarum*, *Bordetella pertussis*, and *Pseudomonas putida.* The evaluation involved comparing the gap-filling outcomes with those of other tools, focusing on the number of reactions proposed to fill the gaps for organism growth. The findings highlighted that CarveMe's gap-filling performance is significantly influenced by media composition. Additionally, the model included numerous reactions that require manual verification. Furthermore, Meneco[95], has been validated for specific case studies including *E. coli* models(iJR904, iAF1260, and iJO1366) and algal model of *Ectocarpus siliculosus*, along with 10,800 degraded networks derived from them. Gap-filling was evaluated based on the tool's ability to restore reaction connectivity from seed metabolites (in the growth medium) to target metabolites. The results for these models revealed that with a 10 percent degradation rate, more than 95 percent of the reactions could be identified in 82 percent of the tested networks. However, the authors emphasize that Meneco cannot replace manual curation and should be used alongside stoichiometry-based tools.

It should be noted that there has been no systematic assessment available specifically for current gap-filling tools. However, a systematic assessment of automated GEM reconstruction tools, which also discussed the gap-filling performance of the tools, was conducted in a recent study by Mendoza et al.,[110]. In this study, 29 GEMs of bacterial organisms were used for assessment. In contrast, our systematic framework utilizes a larger database, 108 models of BiGG database, including a range of both prokaryote and eukaryote models.

In conclusion, we have developed a straightforward systematic assessment approach that can be applied and expanded for the evaluation of various gap-filling tools. This framework gives user a comprehensive view of the performance of the tools in practical scenarios. Gap-filling is still a major challenge in metabolic modeling and none of these tested tools consistently outperforms the others across all defined features, the three of the selected tools showed some strengths and weaknesses across all 108 tested models. However, utilizing each of the 3 tools can be valuable for automated gap-filling, saving a substantial amount of time. Meneco serves as a useful tool for initially visualizing the model as a graph and assessing metabolite connectivity. However, it is important to note that the gap filled models by Meneco are not guaranteed to be functional for advanced analyses such as Flux Balance Analysis (FBA). COBRApy emerges as an indispensable tool, offering a comprehensive suite of analyses on the gap-filled model. Its versatility extends across diverse organisms, can be a reliable choice for working with various organisms. For scenarios involving prokaryotic organisms that utilize BiGG identifiers, CarveMe stands out as a specialized option for gap-filling. It can be a suitable tool when focusing on bacteria or archaea and employing BiGG nomenclature. By recognizing the specific strengths of each tool, these recommendations aim to guide users in selecting the most suitable gap-filling tool based on their organism type and identifier preferences. Given that gap-filling remains a significant challenge in metabolic modeling, leveraging the capabilities of these tools in relevant scenarios can prove beneficial for addressing gaps in a model.

Chapter 5

A Novel Framework for Trimming and Ranking Reactions for Filling Gaps in Genome-Scale Metabolic Models

5.1 Introduction

Genome-scale metabolic models (GEMs) provide a comprehensive and quantitative understanding of the metabolic pathways and reactions within organisms[18, 111]. GEMs have been widely generated and utilized to study a range of organism including archaea, bacteria, and eukaryotes. Gu et al., [17] reported that GEMs for 6239 organisms with 5897 for bacteria, 127 for archaea, and 215 for eukaryotes have been constructed both manually as well as by using automated GEM reconstruction tools. For example, models of *Methanosarcina acetivorans*[112] to explore methanogen research, *Escherichia coli*, *Bacillus subtilis*[111], *Saccharomyces cerevisiae*[35, 113] to optimize the production of valuable compounds such as biofuels, amino acids, and organic acids, *Mus musculus*, *Rattus norvegicus*, *Danio rerio*, *Drosophila melanogaster* to study human diseases and metabolic malfunctions[114], and microbial communities to investigate human gut microbiome, have been constructed[77, 97].

However, direct translation of genomic data to GEMs may not always result in a complete set of reactions due to the potential errors in the annotation algorithms or inherent incompleteness of the genomic data [108, 109, 15]. Correct prediction of the enzymes that are allocated to genes in a sequenced genome is vital in the performance of model [115, 85]. Methods for genome annotation are unreliable and frequently fail to designate to many genes their intended functions or do so incorrectly [86, 85]. The errors in annotation algorithms come from errors in source database, relativity of the alignment threshold, namely low sensitivity and specificity, which leads to incorrect or inaccurate annotation[86, 91, 85, 106]. Consequently, GEMs typically contain gaps, referring to the absence of reactions that consume or produce a metabolite, which lead to dead-end metabolites and blocked reactions in the model, and even existing GEMs of various organisms (e.g., *Saccharomyces cerevisiae*) have been consistently updated multiple times since their initial reconstruction [35, 113].

Thus, a gap-filling process is necessary to enhance the analysis and interpretation of GEMs. The gap-filling process begins with the identification of gaps in the metabolic network and then filling the gaps by candidate reactions from biochemical databases that could complete the metabolic network[109]. This process is usually an iterative process, with multiple rounds of candidate reaction proposals and evaluation, to ensure that the metabolic network is complete and accurate. The ideal outcome of gap-filling process is to identify the minimum number of reactions that are biologically relevant to add to the draft model in order to make it functional, restoring biomass production or the production of specific products. This process typically involves combination of manual editing as well as the use of different semi-automated tools[115, 107, 109].

Numerous tools and algorithms with diverse approaches, including topology-based, optimization-based, and likelihood-based have been developed to accomplish the task of gap-filling in metabolic networks. For instance, Meneco considers a topology-based approach, focusing on the network connectivity and structure to identify missing reactions[95]. On the other hand, several other tools like COBRApy[75], CarveMe[79], OptFill[93], FastGapfill[9], GlobalFit[109], and ModelSEED[76] utilize an optimization-based approach to complete metabolic networks, considering various constraints such as thermodynamics, minimal culture medium and objectives like biomass production or the production of specific products. The underlying goal of these tools is to identify the minimal set of new reactions that need to be added to the initial draft model[6, 15]. In addition, tools such as Pathway Tools[15] and MIRAGE[94] use a likelihood-based approach by incorporation of genomic data and calculation of probability-based score for reactions to be add to the model. Orth and Palsson[108] reviewed the classic approaches for gap-filling and Faria et al.[76] and Karp et al.[10] reviewed most of the recent automated tools and their properties. In all of these mentioned tools the suitable set of reactions for filling gaps are selected from a universal database.

Despite the importance of this universal database, its effect has been generally unappreciated. A universal database serves as a comprehensive pool of biochemical reactions, compounds, and associated information that can be utilized in gap-filling process. It has two significant impacts on the gap-filling process: speed and model validity. Searching for the suitable set of reactions to add to the model can be computationally expensive, especially in optimization-based approaches that use Mixed-Integer Linear Programming (MILP). MILP problems are significantly more challenging to solve which yields to a NP complexity problem. NP complexity, or non-deterministic polynomial time complexity, is a classification of computational problems that characterizes the efficiency of algorithms in solving them. Problems categorized as NP (Non-deterministic Polynomial) are those for which a proposed solution can be verified quickly, but finding a solution itself may require an exponential amount of time. In other words, once a potential solution is provided, its correctness can be checked in polynomial time, yet there is no known algorithm to efficiently find a solution. NP-completeness is a subset of NP problems that are considered among the most challenging; if a polynomial-time algorithm exists for any NP-complete problem, it implies polynomial-time solutions for all problems in NP. Furthermore, the validity of the model can be compromised by the inclusion of incorrect reactions. A larger database can introduce the possibility of including reactions that may not have sufficient evidence for their presence. To illustrate this, several reaction IDs in BiGG database, a repository of well-structured genome-scale metabolic models, do not correspond to real reactions or lack clear reactants/products. Also, multiple reaction IDs represent the same reaction. To eliminate these inconsistencies and enhance the probability of selecting a reaction with certain level of biological relevance over an alternative reaction with insufficient evidence, a trimming approach is needed.

Due to the mentioned issues the typical validation/reconstruction pipeline does not include an explicit universal model selection step. We propose a novel ranking-based approach which curate and reduce the size of the universal database with goal of trimming out biologically implausible and irrelevant reactions. The reduction in universal database size not only improves the efficiency of the gap-filling process but also reduces simulation time. In this approach we build our ranking technique based on three key metrics including similarity, betweenness, and proximity as filters to the initial universal pool of reactions. We calculated the 3 metrics for each reaction and through the combination of metrics we assigned a score to each reaction in the universal pool. For validation of our approach, we applied it to 108 metabolic models available on BiGG database. As a further evaluation of our approach, we tested it on a non-classical newly annotated organism, thraustochytrid, marine unicellular protist, strain T18 which have proven to be a valuable source for production of fatty acids, such as omega-3 long-chain fatty acids as well as biofuels[67, 116]. Although pure genomic data can provide limited information and insight into the cellular processes and confirm the existence of pathways and reactions in the model, looking at the organism as a whole connected network, rather than just a set of genes, improves the characterization of enzymatic functions at the systematic level[117]. Thus, we generated a genome scale metabolic model for T18 to explore its capabilities. However, due to the errors in its annotated data, its genome scale metabolic model contains a significant number of gaps. We applied our methodology to estimate the necessary size of the universal pool for gap filling and incorporated our ranking approach into the SBPRank package developed in-house using Python. Given the pivotal role of a universal database in gap-filling tools, our novel SBPRank package, a specific contribution of this thesis, functions as a crucial pre-processing and trimming tool. It aids in obtaining an appropriate universal database, setting the stage for its effective utilization in the subsequent gap-filling process.

5.2 Theory

Selecting the appropriate set of reactions from a pool of thousands of possible reactions to fill the gaps in a model is a challenging procedure that may lead to some model inaccuracies. Consequently, it is necessary to establish a set of criteria to narrow down the universal pool size to a smaller and more specific set of reactions. By the calculation of a similarity metric, we incorporated the phylogenetic information as a criterion when choosing reactions from a reference database. Evolutionary close organisms are more likely to have similar reactions[118, 119]. Hence, employing a similarity metric allows us to eliminate reactions of organisms that are evolutionarily distant (positioned further away in the phylogenetic tree) from the model with the gap. Instead, our focus shifts to harnessing reactions from closely related organisms.

Furthermore, we calculated the betweenness metric as our second criterion with the goal of prioritizing reactions that are essential for maintaining connectivity of the model, those nodes (e.g., metabolites/reactions) that their removal results in a higher fragmentation of the model. Considering the betweenness metric also aligns with the principle of Occam's razor. The underlying idea is that if we can identify a single reaction that can fill multiple gaps, it may not guarantee accuracy, but it adheres to the general principle of simplicity[120]. By focusing on highly connected nodes, we aim to keep the additions to the model minimal, thereby keeping things concise and reducing complexity. Finally, we defined the proximity metric because missing reactions are typically close to dead-end metabolites in the model. By incorporating the proximity metric, we prioritize reactions that are closer to a dead-end metabolite rather than those reactions that are too far away from it.

5.2.1 Reaction metrics

Similarity

Similarity metric was defined as the phylogenetic values ranging from 0 to 1, showing the similarity between target model (a model with potential gaps) and every single model in the universal database. It is the only metric calculated at the model level in our approach, rather than reaction level. For calculating the similarity metric, we compare metabolites in a model containing gaps with the metabolites of every single model in the reference database. Subsequently, we calculate a ratio representing the similarity between the models, the description of the algorithm we considered for similarity calculation is detailed in [118]. Finally, we assign the calculated similarity value to all the reactions in a model and then normalize the values. This allows for a comparison of our metrics at the reaction level. Although there are different techniques that use a variety of biological bases, including 16S rRNA or 18S rRNA, a selection of conserved proteins, and the entire genome to obtain similarity between organisms, for our study, we have selected a model-based method to limit our input solely to the model itself[118, 121]. By doing so, our analysis remains focused on the available model data, without requiring additional external information inclusion, such as specific details about the 16S rRNA which might not be available for all organisms. Thus, we calculated our similarity metric based on the comparison of the presence or absence of metabolites between the two models. The methodology is described in detail in[118].

Betweenness

Betweenness is defined as the fraction of shortest paths that pass-through a given node (e.g., metabolites) in the network. The shortest path refers to the path of reactions that connects two metabolites in the network with the least number of steps. We chose betweenness in our study to identify and prioritize potential missing key reactions that act as bottlenecks in the model[105].

$$B(i) = \sum_{j \neq i \neq k} \frac{\sigma_{j_k}(i)}{\sigma_{j_k}} \tag{5.1}$$

σ_{j_k} is the total number of shortest paths from node j to node k, and $\sigma_{j_k}(i)$ is the total number of those paths that pass-through node i. A node with a higher betweenness indicates that a greater number of reactions pass through that node. During the calculation of betweenness we excluded currency metabolites (e.g., NADH, NADPH, ATP, etc.) as well as organic(e.g., adenine, guanine, cytosine, etc.), and inorganic(e.g., Na, K^+, $Ca_2{}^+$, etc.) metabolites. Currency metabolites, due to their involvement in numerous cellular reactions, can exhibit high degrees of connectivity, potentially overshadowing the roles of other metabolites. This exclusion allows for a more targeted analysis, emphasizing the contributions of non-currency metabolites to specific pathways and cellular processes. Additionally, it reduces computational complexity, facilitating more manageable analyses. Betweenness was calculated for all metabolites of each reaction, and a single betweenness value was assigned to each reaction in each model by averaging all the metabolites betweenness values. Betweenness values ranging from 0 to 1 with 0 being the low and 1 being the high value. We initially generated a connectivity matrix from all the metabolites within a model. Subsequently, this matrix was transformed into a graph,with metabolites as nodes and reactions as edges, and using the NetworkX package in Python, we computed the betweenness. NetworkX is a Python package designed for the creation, manipulation, and analysis of complex networks or graphs[105]. betweenness-centrality function for computing betweenness. It is important to note that the betweenness measure was applied to the models within the reference database.

Proximity

We calculated the proximity metric, which is the measurement of the distance between a node (e.g., metabolites) and all the other nodes in the model. We initially determined the maximum possible number of steps (distance) from each dead-end metabolite to all other metabolites in each model and then calculated the distance ratio for each reaction (number of steps divided by max number of steps in each model). It's important to emphasize that proximity was calculated for the network of the model with dead-end metabolites, where nodes represent metabolites, and edges/steps represent reactions. Subsequently, we converted the distance ratio values to the proximity metric ranging from 0 to 1. Reactions with values close to 0 indicate worse proximity, meaning they are far away from a dead-end metabolite, while reactions with values close to 1 indicate better proximity, suggesting that those reactions have higher likelihood of being perfect matches to fill the gap hence yield a higher score. Similar to the betweenness calculation, currency metabolites were not considered during the calculation of proximity[105]. It should be noted that proximity is the only metric highly dependent on dead-end metabolites in the model containing gaps.

5.2.2 Metrics aggregation

The metrics calculation process results in a series of values for every single reaction, with one value per unique competitor model in the universal database. A single putative reaction can have several different values because it may appear in multiple models. Therefore, after the calculation of our three metrics for every single reaction in each model, we normalized the values to a range from zero to one. We then selected the maximum value for each reaction across all models. In order to rank the reactions, we need to combine the three calculated metrics into a single score for each reaction. Thus, we utilized Principal Component Analysis (PCA) and heuristic weights to find the best possible combination of the three metric and determine the weight of each metric for the final optimized combination.

Principal Component Analysis (PCA)

We employed PCA as an unbiased straightforward method, which enables us to understand the underlying structure of data and identify the most discriminative and high-impact features. PCA is the most popular approach in analysing variance. Our three metrics of similarity, betweenness, proximity can be correlated to varrying degree. A high PCA score for a specific metric suggests that it plays a crucial role in differentiating the data points and contributes substantially to the total variance captured by the principal components. Therefore, by calculating the maximum PCA score for each of the metrics of similarity, betweenness, and proximity, we ascertained their relative importance with respect to the variance explained in the data and identified the linear combination among them. Thus, we calculated the maximum PCA scores for each of the proximity, similarity, and betweenness metrics, and compared the values for all reactions to identify which metric had the greatest impact (relative importance among the three of them). Following that, we multiplied the optimal combination of PCA scores to our data set for the three metrics and sorted the reactions.

Heuristic weights

PCA provides a convenient way to combine the three metrics, especially in situations where there is no information available about the number of gaps in the model. In addition, to further enhance the results, we investigated a series of combination of weights for the three metrics. We assigned weights (w1, w2, w3) to each metric using the Cartesian product of these weights and identified the optimized weights. The Cartesian product of the weights generates all possible combinations of weights, and by testing different combinations, it is possible to identify the set of weights that best accounts for the interactions between the three metrics. Subsequently, we used these optimized weights as a means to derive the heuristic weights. Furthermore, we employed the heuristic weights as our strategy to rank the universal pool of reactions. By doing so, we were able to assess the influence of every single combinations of the proximity, similarity, and betweenness when analyzing a range of essential reactions as gaps in a model. To assess the impact of each ranking methodology on the identification of essential reactions within the universal pool, as well as the proportion of the universal pool size required to identify each of the essential reactions, we generated samples of essential reactions ranging from 10 to 100 percent. We began by removing 10 percent of essential reactions from a model and utilized our SBPRank package to determine the destination of these removed reactions within the universal pool. This process was then iteratively extended to higher percentages of gap creation, reaching up to 100 percent.

5.3 Methods

Our ranking approach requires a model with potential gaps, which we will refer to as the target model, and a reference database which we will refer to as universal database. Unlike other techniques, our universal database consists of a set of discrete GEMs instead of a single model, which can be downloaded from BiGG database or other repositories. The overall process involves comparing our target model to each of the comparator models in the universal database one-by-one and generating a ranked universal pool of reactions in which each reaction has a score. However, prior to the score calculation we performed a pre-screening step in order to exclude repeated reactions with multiple names and converted BiGG IDs to chemical reactions.

5.3.1 Implementation

SBPRank is our novel Python-based package aimed to narrow down the universal pool size to a smaller and specific set of reactions and thus decrease the simulation time and increase accuracy of the gap-filling process."SBP" in the name of our package is derived from the terms similarity, betweenness, and proximity, while the term "Rank" originates from the nature of our framework as a ranking/scoring approach. SBPRank takes a target model, universal reference models, and blocked metabolites as inputs and generates a set of ranked reactions with their computed scores as output. The core of SBPRank involves three key steps:1) Calculation of similarity, betweenness and proximity metrics, 2) Aggregation of the three metrics; and 3) Ranking the reactions using PCA and heuristic weights. SBPRank is compatible with Python 3.7. It requires the installation of COBRApy, NetworkX, NumPy, Pandas, Matplotlib, and SciPy libraries. The core function of the SBPRank package is available at appendix A.

5.3.2 SBPRank performance analysis

The primary objective of SBPRank as a ranking approach is to assign higher priority to the reactions that belong to a particular model. Furthermore, metabolically essential reactions are expected to receive higher rankings compared to less important reactions within the model. Essential reactions refer to those reactions that their removal from the model leads to zero growth in flux balance analysis (FBA). As a result, we would anticipate that the essential reactions within a model appear relatively high, at the top of the list or as close to the top as possible, during the ranking process. Therefore, our fundamental assessment involves identifying essential reactions in a model in order to investigate the relationship between ranking and reaction essentiality, as well as the relationship between ranking and simulated gaps in a model. We used all the unique models available in the BiGG database (48 models including both prokaryotic and eukaryotic models out of 108 excluding the multistrain models) and identified the essential reactions from each model. Among the 48 selected models, each is unique to a specific organism. The term 'multi-strain models' implies the existence of multiple GEMs for a single organism, each corresponding to different strains of that organism. For instance, *Escherichia coli (E. coli)* is associated with several models, such as iECD1391, iB211397, iAPECO11312, and iECW1372, among others. Furthermore, we measured the performance of our ranking algorithm by universal pool fraction and the percentage of universal pool reduction. For instance, if the universal pool fraction is 10 percent, it signifies that all the missing reactions can be located within the top 10 percent of the pool, enabling us to eliminate the remaining 90 percent of the pool.

5.3.3 The role of ranking in identifying essential reactions

We tested each of the models individually as target models, while using the remaining models as reference models in the rank reaction function in SBPRank to calculate the proximity, similarity, and betweenness metrics. Moreover, we explored two strategies for ranking, namely PCA, and heuristic weights (details in theory section 5.2.2).

5.3.4 The role of ranking in identifying artificially induced gaps

Gaps typically result from the absence of several essential reactions in a model rather than the complete absence of all essential reactions. Thus, it is more practical to focus on a fraction of reactions that are missing, which refer to as gaps. To investigate this further, we created random samples of essential reactions for each model, ranging from 10 to 100 percent. Afterwards, we generated artificial gaps by deleting random sets of essential reactions from each model. Similar to the first case study, we tested these models individually as target models, with using the remaining models as reference models in the rank reaction function in SBPRank as well as PCA and heuristic weights to rank the reactions. The objective was to determine the required size of the ranked universal pool to recover the artificial gaps.

5.3.5 The role of ranking for real gaps (T18 as our case study)

Genome-scale metabolic model generation for T18

We reconstructed a genome-scale metabolic model for T18 based on the available genomic annotated data. The process of reconstruction began by identifying the EC numbers of the reactions from genomic data. A total of 895 unique EC numbers were extracted from T18 genomic data and the EC numbers were subsequently used to infer the presence of metabolic reactions in the microorganism. This is achieved by mapping the EC numbers to the corresponding reactions in a database of known metabolic reactions. We used BiGG as a cross-referencing database to facilitate comparisons with published models[89]. The reaction information was then used to build a metabolic network which included 1812 metabolic reactions and 1557 metabolites present in the model. Moreover, we formulated and added a biomass reaction (approximating the overall requirements for amino acids, nucleotides, carbohydrates, lipids, etc.) based on previous modelling work on similar organisms of *schizochytrium limacinum (SR21)*[98] and *Aurantiochytrium T66*[70]to the model. We performed primary gap-filling based on the assumption that the metabolic model can simulate the production of all biomass metabolites from the components found in glucose-containing media. However, this process was not sufficient for simulating growth. We basically assessed whether the model consistently predicts growth in the presence of glucose and xylose in the culture media.

T18 model validation

We used the ranked reaction function in SBPRank to calculate the betweenness, similarity, and proximity scores for T18 as the target model, with BiGG models as universal database. After that, in order to investigate the model connectivity among various automated tools, we chose Meneco. To determine the minimum universal pool size required to construct all 42 target metabolites (biomass metabolites from T18 model) from the seeds metabolites (culture media metabolites), we divided the total ranked universal pool size into percentages ranging from 1 to 100 and applied the Meneco tool. Subsequently, we calculated both the simulation time and percentage of the required universal pool.

5.4 Results and discussion

We successfully implemented our ranking approach in the SBPRank package. It takes the input of target models, universal reference and blocked metabolites and generates a set of ranked reactions with their computed scores as output. A significant finding from analysis of various percentages of essential reactions by SBPRank as demonstrated in Figure 5.1, is that a range of 10-40 percent of essential reactions can be identified by only searching about 10 percent of the ranked universal pool. Moreover, we observed an interesting trend in the proximity metric. When the number of essential reactions increases by over 30 percent, the impact of proximity metric becomes less significant which is due to the fact that with the increase in the number of gaps in the model, the overall network structure becomes more densely connected around the gaps, bringing surrounding reactions closer to them. Although this situation provides helpful context, it also diminishes the ability to differentiate and prioritize reactions solely based on their proximity to the gaps. The results obtained by ranking reactions through PCA as an unbiased method offer valuable insights into the relationship among the three metrics (similarity, betweenness, and proximity). This prompts further exploration into heuristic weights to identify optimal combinations of metrics that enhance the overall performance of our framework.

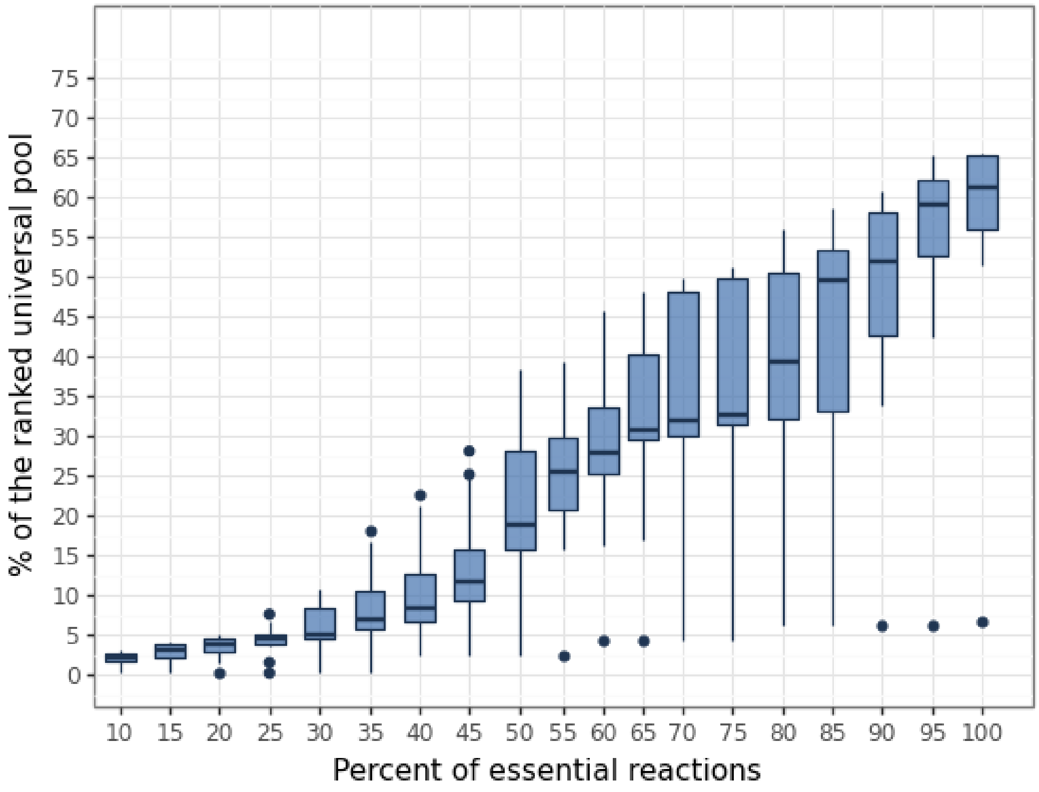

Figure 5.1: Variations in the percentage of ranked universal pool (ranked by PCA) required to capture different percentages of essential reactions from 10 to 100.

5.4.1 Ranking by heuristic weights

The results obtained from ranking by PCA were already promising. We gained information about the underlying relationship between the three metrics and successfully identified almost half of the essential reactions within just searching the top 10 percent of the universal pool, resulting in a substantial 90 percent reduction in universal size. However, to further enhance the performance of our framework, we investigated heuristic weights as an alternative ranking strategy. As outlined in the methods, we examined the effects of different combination of weights and assessed their impact on the minimum required percentage of universal pool. The best outcome from the combination of optimized weights is compared with PCA in Figure 5.2 and the variations in the combination of weights for proximity, similarity, and betweenness across different percentages of essential reaction are presented in Table 5.1. As we can see

Table 5.1: Variations in the optimized weights for proximity, similarity, and betweenness from 10-100% of essential reaction arbitrarily chosen for removal

Percent of Reactions	**Optimized Combination of Weights**		
	Proximity	**Similarity**	**Betweenness**
10	0.85	0.05	0.05
15	0.85	0.05	0.05
20	0.75	0.05	0.20
25	0.75	0.05	0.20
30	0.05	0.90	0.05
35	0.10	0.80	0.10
40	0.10	0.85	0.05
45	0.10	0.85	0.05
50	0.10	0.85	0.05
55	0.20	0.55	0.25
60	0.05	0.60	0.35
65	0.05	0.60	0.35
70	0.10	0.60	0.30
75	0.10	0.60	0.30
80	0.15	0.55	0.30
85	0.15	0.55	0.30
90	0.25	0.20	0.55
95	0.25	0.20	0.55
100	0.25	0.20	0.55

in Table 5.1, with the increase in percent of essential reactions removed, the combination of optimized weights of the three metrics that results in best performance of our ranking framework changes. There has been an improvement in the overall trend compared to Figure 5.1. Within the range of 10-85 percent, essential reactions can be identified by searching only the top 20 percent of the ranked universal pool. The results also show an agreement with the PCA results, mostly at 10-25 percent. However, beyond 30 percent the two trends diverge significantly. Below 25 percent, proximity emerges as the most effective metric, while beyond that point, there is a

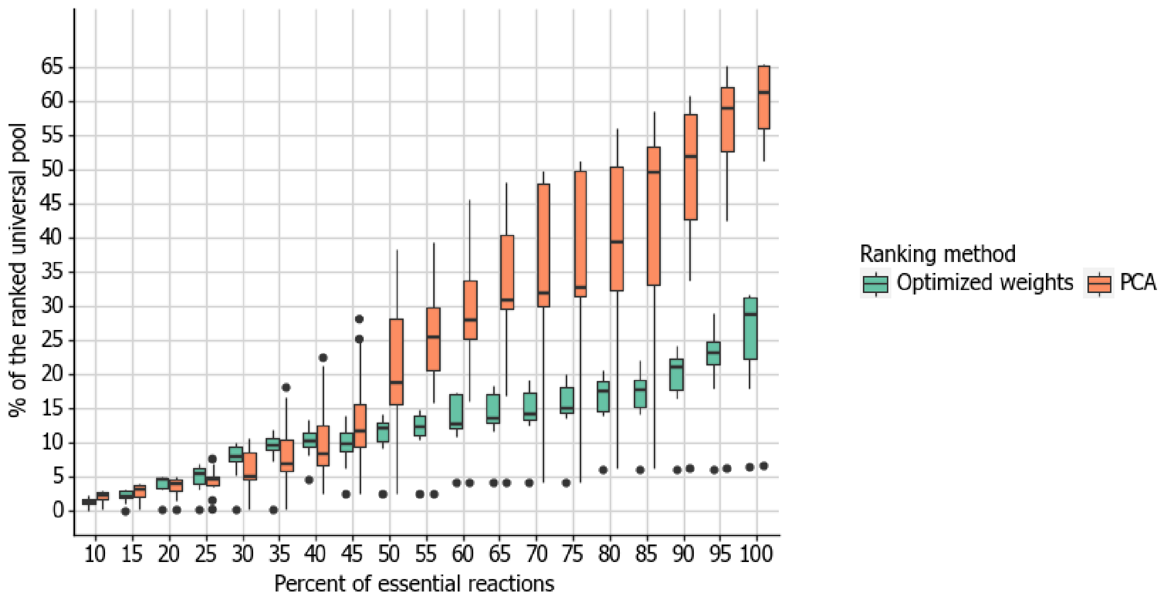

Figure 5.2: Comparison of optimized weights and PCA to identify the smallest ranked universal pool required to identify various percentages of essential reactions from 10 to 100.

drastic shift in the importance of metrics, with similarity becoming the most effective metric. Eventually, at around 80 percent, betweenness replaces similarity in importance, as shown in Table 5.1. PCA serve as a useful approach, especially when there is no initial information available regarding the number of gaps. However, if there is evidence that the model contains many gaps, it would be much more reasonable to implement the heuristic approach to gain better result. The optimal combination of heuristic weights was categorized into three groups based on low, medium, and high percentages of gaps in the model. These categories offer users flexibility in choosing the best combination of weights depending on their preference for addressing a low to high number of gaps in their model during the gap-filling process. In addition, these results are summarized in Table 5.2 and Figure 5.3. For instance, when classifying gaps into low, medium, and high percentages, with 10 to 30 percent considered low, practical weight values could be 0.80 for proximity, 0.10 for similarity, and 0.10 for betweenness. In the 30 to 50 percent gap range, practical weights might be 0.10 for proximity, 0.60 for similarity, and 0.30 for betweenness. For gaps ranging from 55 to 100 percent, we tested with values of 0.10, 0.30, and 0.60 for proximity, similarity, and betweenness, respectively. It's important to note that users have the flexibility to assign these values based on the specific nature of the models they are working with.

Table 5.2: Variations in the heuristic weights for proximity, similarity, and betweenness from 10-100% of essential reaction arbitrarily chosen for removal

Category of % Reactions	**Heuristic Weights**		
	Proximity	**Similarity**	**Betweenness**
Low	0.80	0.10	0.10
Medium	0.10	0.60	0.30
High	0.10	0.30	0.60

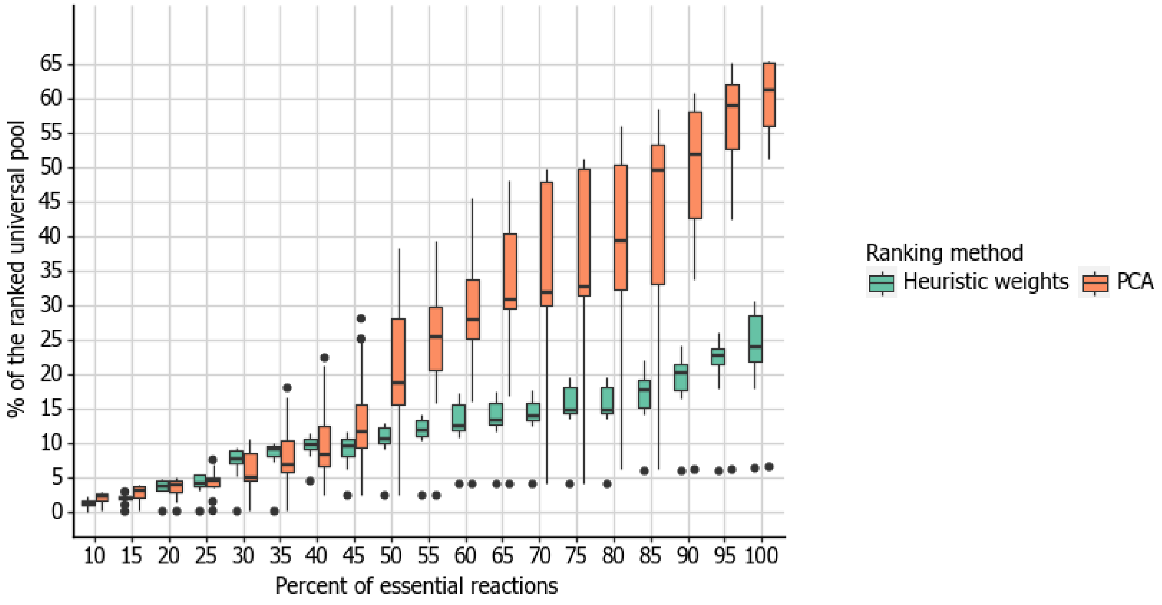

Figure 5.3: Comparison of heuristic weights and PCA to identify the smallest ranked universal pool required to identify various percentages of essential reactions from 10 to 100.

5.4.2 Variations in universal pool size with random artificial gaps

Since gaps often result from the absence of only a subset of essential reactions in the model, addressing these sorts of gaps is more practical. Therefore, we calculated the required universal pool for random sets of artificial gaps created from all essential reactions in each model, using both PCA and heuristic weights as our ranking methods. Figure 5.4 presents the percentage of the ranked universal pool for different retrieval percentages (10-100%) of random essential reactions using PCA and heuristic weights. The result from this analysis is valuable since dealing with the complete absence of all essential reactions can be much more challenging due our limited knowledge about the precise number of gaps in models and may require substantial efforts. However, focusing on a subset of missing reactions provides a more manageable scope for improvements. The results indicate similar trend to previous analysis. While random

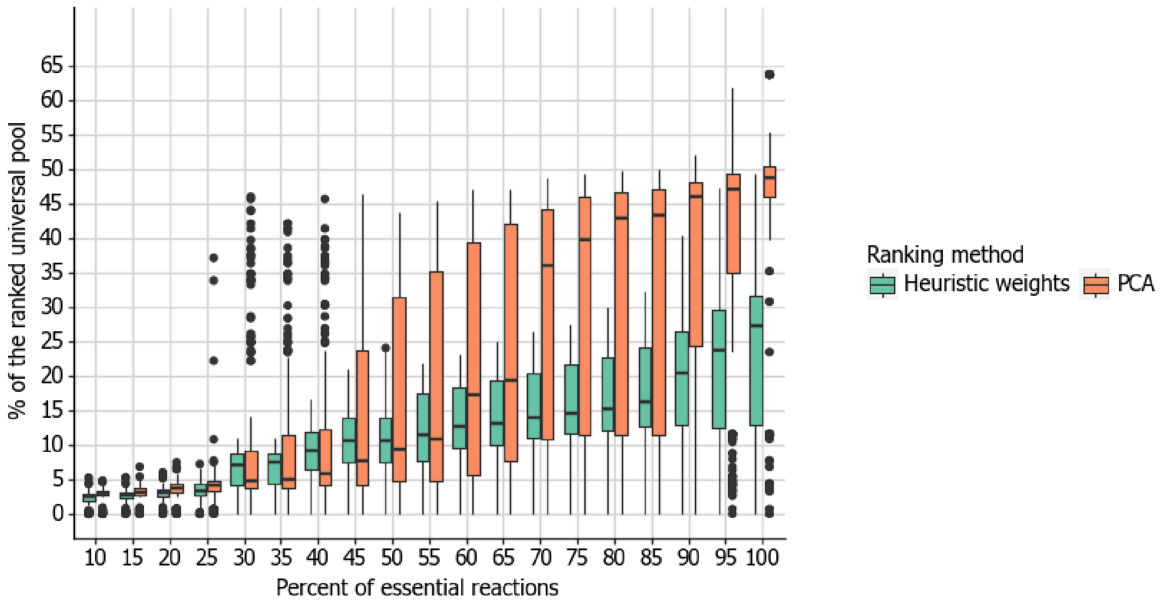

Figure 5.4: Comparison of the heuristic weights and PCA to identify the required ranked universal pool required to identify various percentages of random essential reactions from 10 to 100.

reaction selection can be a challenging case, it is still possible to find 10-85 percent of reactions by exploring only 20 percent of the ranked universal pool. In conclusion, in cases where only 10-30 percent of reactions are missing from a model, realistically occurring in gap-filling processes, we might only need to search 5 percent of the universal pool to find reactions. Furthermore, it should be highlighted that lack of 60 to 100 percent of essential reactions from a model indicates serious issues with the model construction rather than merely several missing reactions that need to be filled to model for completion.

5.4.3 Ranking results for T18 model

We were interested to identify the required universal pool size for filling the gaps in T18 model. Hence, in the analysis of T18, we utilized both PCA and heuristic weights for ranking. The required pool size was the same with either ranking method, and the gaps were identified at around top 20 percent of the ranked universal pool as it is shown in Figure 5.5. Then, we continued by the PCA ranking method, which is the default ranking approach, in Figure 5.6. Meneco was used as the gap-filling tool in this case. Meneco stands as the only graph-based gap-filling approach currently available. It was selected to determine the required pool size for the T18 model based on our extensive exploration of available gap-filling tools to address the gaps in T18 model, as discussed in chapter 3 and 4. Meneco showed to be the only gap-filling tool that successfully worked for the gap-filling of the T18 model, as the newly annotated microorganism. Notably, Meneco is the only tool specifically designed for identifying gaps in newly-explored organisms with limited available information. Furthermore, Meneco distinguishes itself as the fastest tool among all available gap-filling tools, making it a valuable approach for analyzing larger models in terms of their overall structure. A significant improvement was observed in 5 percent of the ranked universal pool, with 26 reconstructable targets out of a total of 42 target metabolites. Ultimately, at 20 percent of the ranked universal pool, 41 targets were successfully reconstructed. Furthermore, the simulation time decreased from 64.5 minutes when considering 100 percent of the ranked universal pool to 1.5 minutes when focusing on 20 percent of the ranked universal pool. The simulation time is exclusively presented in this section to offer insights when the precise number of gaps is unclear. T18 serves as a realistic case study wherein the exact number of gaps was not known, unlike the previous sections where we were aware of the gap count from the outset (artificial gaps). Furthermore, it is important to note that the relationship between simulation time and the percentage of the ranked universal pool is nonlinear. While Meneco provides faster processing compared to other approaches and proves to be a practical tool when addressing numerous gaps that impact the overall connectivity of the model, it is important to note that Meneco is not a constraint-based tool and does not involve optimizations such as FBA. In contrast, other approaches that incorporate optimization techniques may result in significantly longer run times, potentially extending to several days when considering the complete universal pool. Therefore, a universal pool size reduction of 80 percent is necessary in order to practically employ gap-filling techniques. This reduction in size allows for more manageable simulation times while still achieving effective gap-filling results. In this study, simulations and analyses were performed by AMD Ryzen 9 5900x processor using Linux operating system.

5.5 Alternative trimming methods

Trimming of reference databases for filling gaps in genome-scale metabolic models is a crucial step in ensuring the accuracy and reliability of the gap-filling process[122]. SBPRank stands out as the only fully automated approach capable of effectively trimming the universal database for any organism type, considering three metrics. In some studies, manual curation is employed for database refinement. For instance, CarveMe manually curated the BiGG database to generate a universal database for bacterial model gap-filling[79] and Reconstructor manually curate ModelSEED database by removing all reactions that are unbalanced for bacterial model gap-filling[96]. Another study manually curated the MetaCyc database for gap-filling in *Bifidobacterium longum subsp. longum JCM 1217*, revealing inaccuracies in reactions, such as the

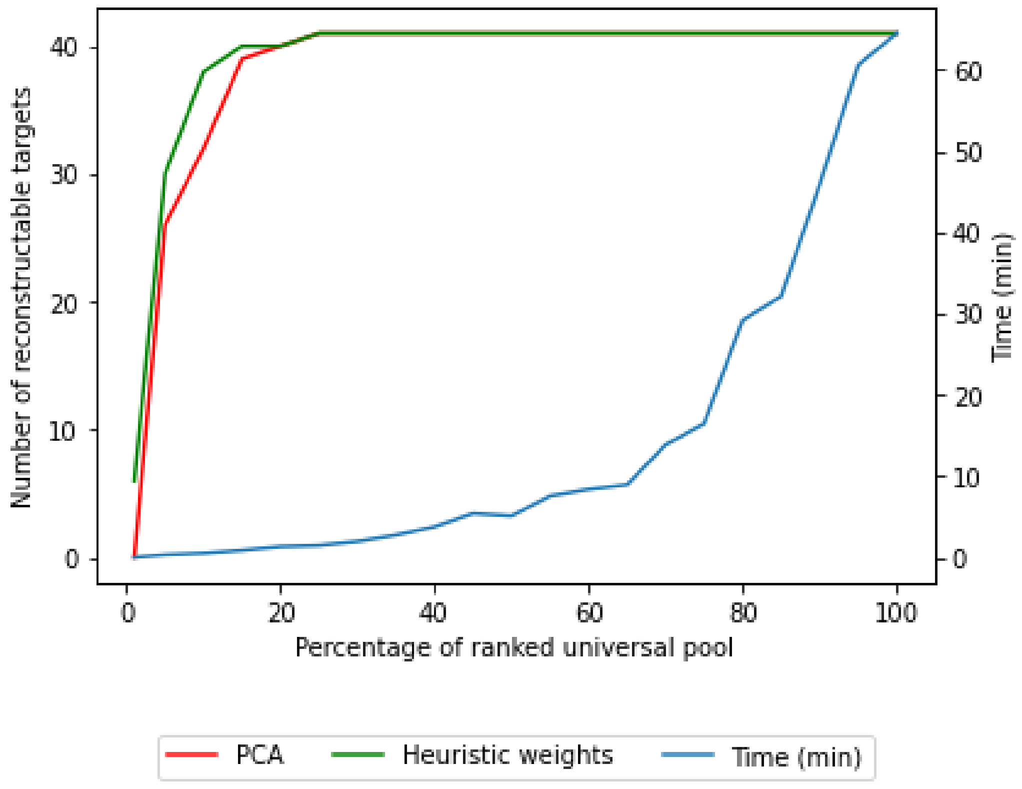

Figure 5.5: Comparison of changes in constructability of target metabolites and Meneco tool implementation time with different percentages of ranked universal pool size for genomic T18 model when ranking by both PCA and heuristic weights.

absence of a suitable cytosolic-only electron transfer from NADPH to ferredoxin[10]. Similarly, a study by Latendresse et al.,[123] focused on the EcoCyc database, degrading it to EcoCyc-20.0-GEM to explore growth on glucose in a single case study. Some approaches incorporate extra data, like genomic information, for database trimming, a feature that may not be universally available. For example, Mirage is a genomic-based algorithm that considers sequence-similarity searches and enzyme phylogenetic profiles (based on comparing the presence and absence of enzymes) to trim the universal database across cyanobacteria species[94], while ModelSEED employs genomics data for gap filling by integrating genomic information to identify candidate genes for the reactions that need to be filled in a genome-scale metabolic model. This process involves mapping between genes and reactions[76]. In conclusion, the typical

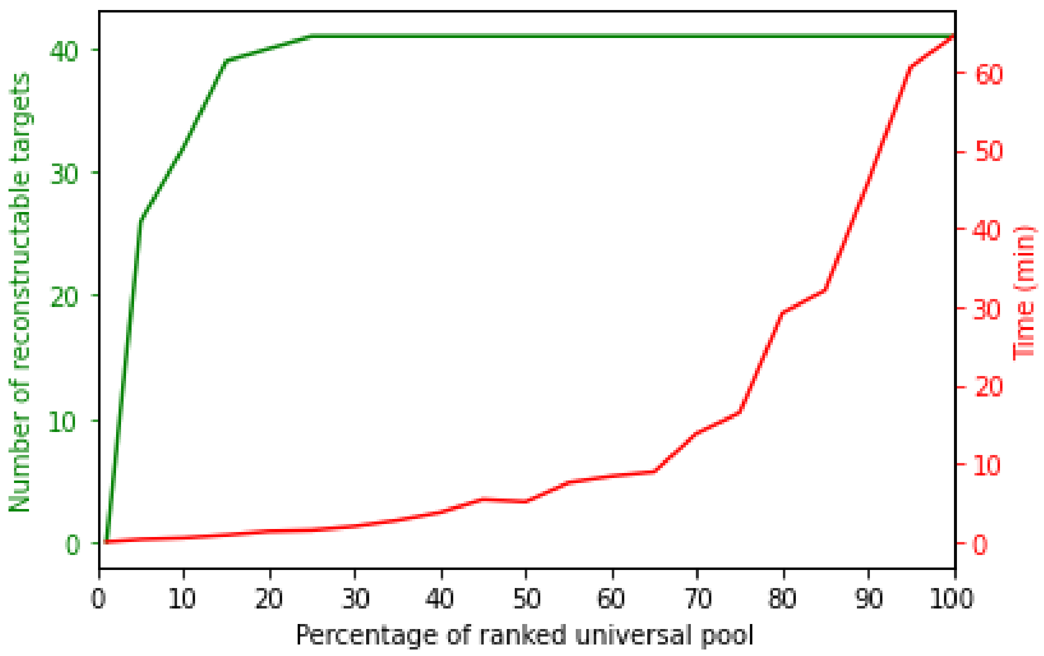

Figure 5.6: Variations in constructability of target metabolites and Meneco tool implementation time with different percentages of ranked universal pool size for genomic T18 model.

approach for trimming a universal database currently relies on manual curation, often tailored to specific models of interest, or utilizes a single metric such as phylogenetic or genetic information. SBPRank advances beyond these approaches by incorporating three reasonable metrics.

Chapter 6

Conclusion

The primary objective of this thesis was to explore the application of the reconstruction of genome-scale metabolic model for a recently sequenced microorganism, the thraustochytrid strain T18. Our goals included investigating its metabolic capabilities for utilizing xylose as a cost-effective carbon source, comparing its behavior with glucose consumption, and examining its fatty acid synthesis pathways. Although we generated a GEM for strain T18 with 2252 reactions and 1952 metabolites, the overall modeling effort was not entirely sufficient due to significant uncertainties and incompleteness in the T18 genomic data. Incomplete annotations often resulted in multiple gaps in the metabolic model. In the case of T18, despite our best efforts to identify and fill model gaps through iterative rounds of model refinement and manual curation, major gaps remained unsolved, and the true pathway structure could only be guessed. Approximately 40 percent of metabolites included in the model lacked consumption or production reactions. We have proposed potential solutions, including a plausible xylose metabolism pathway, specific EC numbers, and associated reactions, which could contribute to enhancing the data quality for T18. These suggestions, however, necessitate additional experimental validation for confirmation. Furthermore, refining other aspects, such as precisely defining the composition of the biomass reaction for T18 and obtaining a detailed profile of fatty acids production, holds the potential to further elevate the model's accuracy. To mitigate uncertainty in genomic data, it is imperative to employ high-quality DNA extraction methods that guarantee the purity and integrity of genetic material. Furthermore, the selection of sequencing technologies should be meticulous, taking into account factors like accuracy, coverage, and read length. Additionally, a comprehensive validation of genomic findings using complementary experimental techniques, such as polymerase chain reaction (PCR) or functional assays, is crucial to affirm the accuracy of the genomic data. As our second objective, we conducted a comprehensive systematic assessment of the performance of current gap-filling tools. We have introduced a novel and versatile framework designed for the comprehensive assessment of gap-filling tools, with a specific focus on evaluating popular tools including COBRApy, Meneco, and CarveMe. Through our methodology, we scrutinized the effectiveness of these tools, revealing that gap-filling remains a substantial challenge in metabolic modeling. Notably, across the extensive BiGG database, COBRApy, CarveMe, and Meneco collectively addressed only half of the tested gaps(essential reactions removal). Importantly, our framework possesses a broad applicability, extending its use beyond the evaluated tools to assess other biochemical databases like MetaCyc and ModelSEED. Furthermore, while our assessment primarily centered on biological aspects of the models, there is potential for incorporating additional variables that focus on mathematical aspects such as shadow prices and condition numbers for a more comprehensive evaluation. For our final goal in this thesis, we developed a novel ranking approach and successfully implemented that in a Python-based SBPRank package. This framework takes target models, a universal reference and blocked metabolites as inputs and generates a set of ranked reactions with their computed scores as output. Our results, utilizing this approach in identifying the missing reactions in a model indicate that searching only top 5 percent of the universal pool is sufficient to find reactions when only 10-30 percent of reactions are missing from a model, realistically occurring in gap-filling processes. Moreover, when 10-85 percent of reactions are missing from a model, exploring only top 20 percent of the ranked universal pool is adequate for the gap-filling process.

Moreover, the utilization of SBPRank for trimming the universal database markedly improves both simulation times and gap-filling results. The culmination of this thesis provides a robust genome-scale metabolic modeling investigation with the focus on non-conventional microorganisms with high industrial potentials, the thraustochytrid strain T18. Additionally, it introduces a novel and practical strategy for assessing gap-filling algorithms in addressing gaps and uncertainties. Furthermore, we propose a novel framework, SBPRank, that emphasizes the importance of the universal database size in the gap-filling process, contributing to the advancement of metabolic modeling techniques. This research represents a valuable contribution to the fields of metabolic modelling and systems biology.

Bibliography

[1] Axel von Kamp, Sven Thiele, Oliver Hädicke, and Steffen Klamt. Use of cellnet-analyzer in biotechnology and metabolic engineering. *Journal of Biotechnology*, 261:221–228, 11 2017.

[2] Adam M. Feist and Bernhard O. Palsson. The biomass objective function. *Current Opinion in Microbiology*, 13:344–349, 6 2010.

[3] Leonid Chindelevitch, Jason Trigg, Aviv Regev, and Bonnie Berger. An exact arithmetic toolbox for a consistent and reproducible structural analysis of metabolic network models. *Nature Communications*, 5, 2014.

[4] Athanasios Antonakoudis, Rodrigo Barbosa, Pavlos Kotidis, and Cleo Kontoravdi. The era of big data : Genome-scale modelling meets machine learning. *Computational and Structural Biotechnology Journal*, 18:3287–3300, 2020.

[5] Erwin P. Gianchandani, Arvind K. Chavali, and Jason A. Papin. The application of flux balance analysis in systems biology. *Wiley Interdisciplinary Reviews: Systems Biology and Medicine*, 2:372–382, 5 2010.

[6] Daniel A. Cuevas, Janaka Edirisinghe, Chris S. Henry, Ross Overbeek, Taylor G. O'Connell, and Robert A. Edwards. From dna to fba: How to build your own genome-scale metabolic model. *Frontiers in Microbiology*, 7:907, 6 2016.

[7] Karthik Raman and Nagasuma Chandra. Flux balance analysis of biological systems: applications and challenges. *Briefings in Bioinformatics*, 10(4):435–449, 03 2009.

[8] Laurent Heirendt, Ines Thiele, and Ronan M T Fleming. DistributedFBA.jl: high-level, high-performance flux balance analysis in Julia. *Bioinformatics*, 33(9):1421–1423, 01 2017.

[9] Ines Thiele, Nikos Vlassis, and Ronan M.T. Fleming. No title. *Bioinformatics*, 30:2529–2531, 9 2014.

[10] Peter D. Karp, Daniel Weaver, and Mario Latendresse. How accurate is automated gap filling of metabolic models? *BMC Systems Biology*, 12:73, 12 2018.

[11] Matthew N. Benedict, Michael B. Mundy, Christopher S. Henry, Nicholas Chia, and Nathan D. Price. Likelihood-based gene annotations for gap filling and quality assessment in genome-scale metabolic models. *PLoS Computational Biology*, 10, 2014.

[12] Jorge Fernandez de Cossio-Diaz, Kalet Leon, and Roberto Mulet. Characterizing steady states of genome-scale metabolic networks in continuous cell cultures. *PLoS Computational Biology*, 13, 11 2017.

[13] Georgios Marinos, Christoph Kaleta, and Silvio Waschina. Defining the nutritional input for genome-scale metabolic models: A roadmap. *PLoS ONE*, 15, 8 2020.

[14] Beom Gi Park, Minsuk Kim, Joonwon Kim, Heewang Yoo, and Byung-Gee Kim. Systems biology for understanding and engineering of heterotrophic oleaginous microorganisms. *Biotechnology Journal*, 12(1):1600104, 2017.

[15] Ines Thiele and Bernhard Ø Palsson. A protocol for generating a high-quality genome-scale metabolic reconstruction. *Nature Protocols 2010 5:1*, 5:93–121, 1 2010.

[16] Scott A. Becker, Adam M. Feist, Monica L. Mo, Gregory Hannum, Bernhard Ø Palsson, and Markus J. Herrgard. Quantitative prediction of cellular metabolism with constraint-based models: The cobra toolbox. *Nature Protocols*, 2:727–738, 3 2007.

[17] Changdai Gu, Gi Bae Kim, Won Jun Kim, Hyun Uk Kim, and Sang Yup Lee. Current status and applications of genome-scale metabolic models. *Genome Biology*, 20:1–18, 2019.

[18] Kate Campbell, Jianye Xia, and Jens Nielsen. The impact of systems biology on bioprocessing. *Trends in Biotechnology*, 35:1156–1168, 12 2017.

[19] Darryl Joy, Kohei Yoneda, and Iwane Suzuki. Genetic modi fi cation of the thraustochytrid aurantiochytrium sp . 18w-13a for cellobiose utilization by secretory expression of β-glucosidase from aspergillus aculeatus. *Algal Research*, 40:101503, 2019.

[20] Xiao man Sun, Ying shuang Xu, and He Huang. Trends in biotechnology forum lipid compounds. *Trends in Biotechnology*, 39:648–650, 2021.

[21] Ann Kathrin Löbs, Cory Schwartz, and Ian Wheeldon. Genome and metabolic engineering in non-conventional yeasts: Current advances and applications. *Synthetic and Systems Biotechnology*, 2:198–207, 2017.

[22] Thomas W. Jeffries. Engineering yeasts for xylose metabolism. *Current Opinion in Biotechnology*, 17:320–326, 6 2006.

[23] Hang Zhou, Jing sheng Cheng, Benjamin L. Wang, Gerald R. Fink, and Gregory Stephanopoulos. Xylose isomerase overexpression along with engineering of the pentose phosphate pathway and evolutionary engineering enable rapid xylose utilization and ethanol production by saccharomyces cerevisiae. *Metabolic Engineering*, 14:611–622, 11 2012.

[24] Shalley Sharma and Anju Arora. Tracking strategic developments for conferring xylose utilization/fermentation by saccharomyces cerevisiae. *Annals of Microbiology 2020 70:1*, 70:1–17, 8 2020.

[25] Alexandra Merkx-Jacques, Holly Rasmussen, Denise M Muise, Jeremy JR Benjamin, Haila Kottwitz, Kaitlyn Tanner, Michael T Milway, Laura M Purdue, Mark A Scaife, Roberto E Armenta, et al. Engineering xylose metabolism in thraustochytrid t18. *Biotechnology for biofuels*, 11(1):1–18, 2018.

[26] Gregory L. Medlock and Jason A. Papin. Guiding the refinement of biochemical knowledgebases with ensembles of metabolic networks and machine learning. *Cell Systems*, 10:109–119.e3, 1 2020.

[27] Barbara Petschacher and Bernd Nidetzky. Altering the coenzyme preference of xylose reductase to favor utilization of nadh enhances ethanol yield from xylose in a metabolically engineered strain of saccharomyces cerevisiae. *Microbial Cell Factories 2008 7:1*, 7:1–12, 3 2008.

[28] I Thiele, N Jamshidi, Rmt M T Fleming, and B Ø Palsson. Genome-scale reconstruction of escherichia coli's transcriptional and translational machinery: A knowledge base, its mathematical formulation, and its functional characterization. *PLoS Comput Biol*, 5:1000312, 2009.

[29] Daniel Hartleb, Florian Jarre, and Martin J Lercher. Improved metabolic models for e. coli and mycoplasma genitalium from globalfit, an algorithm that simultaneously matches growth and non-growth data sets. *PLoS computational biology*, 12(8):e1005036, 2016.

[30] Soo Yun Moon, Soon Ho Hong, Tae Yong Kim, and Sang Yup Lee. Metabolic engineering of escherichia coli for the production of malic acid. *Biochemical Engineering Journal*, 40(2):312–320, 2008.

[31] Ana Rita Brochado, Claudia Matos, Birger L Møller, Jørgen Hansen, Uffe H Mortensen, and Kiran Raosaheb Patil. Improved vanillin production in baker's yeast through in silico design. *Microbial cell factories*, 9(1):1–15, 2010.

[32] Stephen S. Fong, Anthony P. Burgard, Christopher D. Herring, Eric M. Knight, Frederick R. Blattner, Costas D. Maranas, and Bernhard O. Palsson. In silico design and adaptive evolution of escherichia coli for production of lactic acid. *Biotechnology and Bioengineering*, 91:643–648, 9 2005.

[33] Intawat Nookaew, Michael C Jewett, Asawin Meechai, Chinae Thammarongtham, Kobkul Laoteng, Supapon Cheevadhanarak, Jens Nielsen, and Sakarindr Bhumiratana. The genome-scale metabolic model iin800 of saccharomyces cerevisiae and its validation: a scaffold to query lipid metabolism. *BMC Systems Biology 2008 2:1*, 2:1–15, 8 2008.

[34] Kaisa Karhumaa, Bärbel Hahn-Hägerdal, and Marie-F Gorwa-Grauslund. Investigation of limiting metabolic steps in the utilization of xylose by recombinant saccharomyces cerevisiae using metabolic engineering. *Yeast*, 22(5):359–368, 2005.

[35] Stefan Krahulec, Barbara Petschacher, Michael Wallner, Karin Longus, Mario Klimacek, and Bernd Nidetzky. Fermentation of mixed glucose-xylose substrates by engineered strains of saccharomyces cerevisiae: role of the coenzyme specificity of xylose reductase, and effect of glucose on xylose utilization. *Microbial cell factories*, 9(1):1–14, 2010.

[36] Feng-Yan Bai, Da-Yong Han, Shou-Fu Duan, and Qi-Ming Wang. The ecology and evolution of the baker's yeast saccharomyces cerevisiae. *Genes*, 13(2):230, 2022.

[37] Andersen MR, Vongsangnak W, Panagiotou G, Salazar MP, Lehmann L, and Nielsen J. A trispecies aspergillus microarray: comparative transcriptomics of three aspergillus species. *Proceedings of the National Academy of Sciences of the United States of America*, 105:4387–4392, 3 2008.

[38] Fangyu Cheng, Huimin Yu, and Gregory Stephanopoulos. Engineering corynebacterium glutamicum for high-titer biosynthesis of hyaluronic acid. *Metabolic Engineering*, 55:276–289, 2019.

[39] Hal Alper, Yong Su Jin, J. F. Moxley, and G. Stephanopoulos. Identifying gene targets for the metabolic engineering of lycopene biosynthesis in escherichia coli. *Metabolic Engineering*, 7:155–164, 5 2005.

[40] Yu Kyung Jung, Tae Yong Kim, Si Jae Park, and Sang Yup Lee. Metabolic engineering of escherichia coli for the production of polylactic acid and its copolymers. *Biotechnology and bioengineering*, 105(1):161–171, 2010.

[41] Hyung Seok Choi, Sang Yup Lee, Tae Yong Kim, and Han Min Woo. In silico identification of gene amplification targets for improvement of lycopene production. *Applied and Environmental Microbiology*, 76:3097–3105, 5 2010.

[42] Soo Yun Moon, Soon Ho Hong, Tae Yong Kim, and Sang Yup Lee. Metabolic engineering of escherichia coli for the production of malic acid. *Biochemical Engineering Journal*, 40:312–320, 6 2008.

[43] Hwan Park Jin, Ho Lee Kwang, Yong Kim Tae, and Yup Lee Sang. Metabolic engineering of escherichia coli for the production of l-valine based on transcriptome analysis and in silico gene knockout simulation. *Proceedings of the National Academy of Sciences of the United States of America*, 104:7797–7802, 5 2007.

[44] Brett A. Boghigian, John Armando, Daniel Salas, and Blaine A. Pfeifer. Computational identification of gene over-expression targets for metabolic engineering of taxadiene production. *Applied Microbiology and Biotechnology*, 93:2063–2073, 3 2012.

[45] Yu Kyung Jung, Tae Yong Kim, Si Jae Park, and Sang Yup Lee. Metabolic engineering of escherichia coli for the production of polylactic acid and its copolymers. *Biotechnology and Bioengineering*, 105:161–171, 1 2010.

[46] Heiko Babel and Jens O Krömer. Evolutionary engineering of e. coli mg1655 for tolerance against isoprenol. *Biotechnology for Biofuels*, 13(1):1–13, 2020.

[47] Jing Wang, Baoyun Zhang, Jie Zhang, Honghui Wang, Minghui Zhao, Nan Wang, Lichun Dong, Xiaohua Zhou, and Dan Wang. Enhanced succinic acid production and magnesium utilization by overexpression of magnesium transporter mgta in escherichia coli mutant. *Bioresource technology*, 170:125–131, 2014.

[48] Kuhn Ip, Neil Donoghue, Min Kyung Kim, and Desmond S Lun. Constraint-based modeling of heterologous pathways: Application and experimental demonstration for overproduction of fatty acids in escherichia coli. *Biotechnology and bioengineering*, 111(10):2056–2066, 2014.

[49] Mihir V. Shah, Hadi Nazem-Bokaee, James Antoney, Suk Woo Kang, Colin J. Jackson, and Colin Scott. Improved production of the non-native cofactor f420 in escherichia coli. *Scientific Reports 2021 11:1*, 11:1–16, 11 2021.

[50] Chao Ye, Qiuling Luo, Liang Guo, Cong Gao, Nan Xu, Li Zhang, Liming Liu, and Xiulai Chen. Improving lysine production through construction of an escherichia coli enzyme-constrained model. *Biotechnology and bioengineering*, 117(11):3533–3544, 2020.

[51] Lisha Qu, Xiang Xiu, Guoyun Sun, Chenyang Zhang, Haiquan Yang, Yanfeng Liu, Jianghua Li, Guocheng Du, Xueqin Lv, and Long Liu. Engineered yeast for efficient de novo synthesis of 7-dehydrocholesterol. *Biotechnology and Bioengineering*, 119:1278–1289, 5 2022.

[52] Kangsan Kim, Donghui Choe, Yoseb Song, Minjeong Kang, Seung Goo Lee, Dae Hee Lee, and Byung Kwan Cho. Engineering bacteroides thetaiotaomicron to produce non-native butyrate based on a genome-scale metabolic model-guided design. *Metabolic Engineering*, 68:174–186, 11 2021.

[53] Ilaria Massaiu, Lorenzo Pasotti, Nikolaus Sonnenschein, Erlinda Rama, Matteo Cavaletti, Paolo Magni, Cinzia Calvio, and Markus J Herrgård. Integration of enzymatic data in bacillus subtilis genome-scale metabolic model improves phenotype predictions and enables in silico design of poly-γ-glutamic acid production strains. *Microbial cell factories*, 18(1):1–20, 2019.

[54] Fangyu Cheng, Huimin Yu, and Gregory Stephanopoulos. Engineering corynebacterium glutamicum for high-titer biosynthesis of hyaluronic acid. *Metabolic Engineering*, 55:276–289, 9 2019.

[55] Zhuangrong Huang, Jianlin Xu, Andrew Yongky, Caitlin S. Morris, Ashli L. Polanco, Michael Reily, Michael C. Borys, Zheng Jian Li, and Seongkyu Yoon. Cho cell productivity improvement by genome-scale modeling and pathway analysis: Application to feed supplements. *Biochemical Engineering Journal*, 160, 8 2020.

[56] Noemi Tejera, Lisa Crossman, Bruce Pearson, Emily Stoakes, Fauzy Nasher, Bilal Djeghout, Mark Poolman, John Wain, and Dipali Singh. Genome-scale metabolic model driven design of a defined medium for campylobacter jejuni m1cam. *Frontiers in Microbiology*, 11, 6 2020.

[57] Iman Shahidi Pour Savizi, Tooba Soudi, and Seyed Abbas Shojaosadati. Systems biology approach in the formulation of chemically defined media for recombinant protein overproduction. *Applied microbiology and biotechnology*, 103:8315–8326, 2019.

[58] Hae Woo Lee, Andrew Christie, Jason A. Starkey, Erik K. Read, and Seongkyu Yoon. Intracellular metabolic flux analysis of cho cells supplemented with wheat hydrolysates for improved mab production and cell-growth. *Journal of Chemical Technology and Biotechnology*, 90:291–302, 2 2015.

[59] Nicolas Loira, Thierry Dulermo, Jean marc Nicaud, and David James Sherman. A genome-scale metabolic model of the lipid-accumulating yeast yarrowia lipolytica. *BMC Systems Biology 2012 6:1*, 6:1–9, 5 2012.

[60] Pornkamol Unrean, Sutamat Khajeeram, and Verawat Champreda. Combining metabolic evolution and systematic fed-batch optimization for efficient single-cell oil production from sugarcane bagasse. *Renewable Energy*, 111:295–306, 2017.

[61] Ahmad Ahmad, Hassan B. Hartman, S. Krishnakumar, David A. Fell, Mark G. Poolman, and Shireesh Srivastava. A genome scale model of geobacillus thermoglucosidasius (c56-ys93) reveals its biotechnological potential on rice straw hydrolysate. *Journal of Biotechnology*, 251:30–37, 6 2017.

[62] Natalie C. Duarte, Scott A. Becker, Neema Jamshidi, Ines Thiele, Monica L. Mo, Thuy D. Vo, Rohith Srivas, and Bernhard Ø. Palsson. Global reconstruction of the human metabolic network based on genomic and bibliomic data. *Proceedings of the National Academy of Sciences*, 104(6):1777–1782, 2007.

[63] Elizabeth Brunk, Swagatika Sahoo, Daniel C Zielinski, Ali Altunkaya, Andreas Dräger, Nathan Mih, Francesco Gatto, Avlant Nilsson, German Andres Preciat Gonzalez, Maike Kathrin Aurich, et al. Recon3d enables a three-dimensional view of gene variation in human metabolism. *Nature biotechnology*, 36(3):272–281, 2018.

[64] Justyna Nocon, Matthias G. Steiger, Martin Pfeffer, Seung Bum Sohn, Tae Yong Kim, Michael Maurer, Hannes Rußmayer, Stefan Pflügl, Magnus Ask, Christina Haberhauer-Troyer, Karin Ortmayr, Stephan Hann, Gunda Koellensperger, Brigitte Gasser, Sang Yup Lee, and Diethard Mattanovich. Model based engineering of pichia pastoris central metabolism enhances recombinant protein production. *Metabolic Engineering*, 24:129–138, 7 2014.

[65] Xingyu Zhu, Siting Shuangfei Li, Liangxu Liu, Siting Shuangfei Li, Yanqing Luo, Chuhan Lv, Boyu Wang, Christopher H.K. Cheng, Huapu Chen, Xuewei Yang, Biosynthesis Pathway, Xingyu Zhu, Siting Shuangfei Li, Liangxu Liu, Siting Shuangfei Li, Yanqing Luo, Chuhan Lv, Boyu Wang, Christopher H.K. Cheng, Huapu Chen, and Xuewei Yang. Genome sequencing and analysis of thraustochytriidae sp. szu445 provides novel insights into the polyunsaturated fatty acid biosynthesis pathway. *Marine Drugs*, 18, 2020.

[66] Lu jing Ren Æ He, Huang Æ Ai hua Xiao, Æ Min Lian, Li jing Jin Æ Xiao-jun Ji, Lu-Jing Jing Ren, He Huang, Ai-Hua Hua Xiao, Min Lian, Li-Jing Jing Jin, Xiao-Jun Jun Ji, Lu jing Ren Æ He, Huang Æ Ai hua Xiao, Æ Min Lian, Li jing Jin Æ Xiao-jun Ji, Lu-Jing Jing Ren, He Huang, Ai-Hua Hua Xiao, Min Lian, Li-Jing Jing Jin, and Xiao-Jun Jun Ji. Enhanced docosahexaenoic acid production by reinforcing acetyl-coa and nadph supply in schizochytrium sp . hx-308. *Bioprocess and Biosystems Engineering 2009 32:6*, 32:837–843, 3 2009.

[67] Alok Patel, Stephan Liefeldt, Ulrika Rova, Paul Christakopoulos, and Leonidas Matsakas. Co-production of dha and squalene by thraustochytrid from forest biomass. *Scientific Reports*, 10(1):1992, 2020.

[68] Ikumi Endo, Takashi Watanabe, Tomofumi Miyamoto, Hatsumi Monjusho-Goda, Junichiro Ohara, Masahiro Hayashi, Yoichiro Hama, Yohei Ishibashi, Nozomu Okino, and Makoto Ito. C4-monomethylsterol β-glucoside and its synthase in aurantiochytrium limacinum mh0186. *Glycobiology*, 7 2021.

[69] Kandasamy Ulaganathan, Sravanthi Goud, Madhavi Reddy, and Ulaganathan Kayalvili. Genome engineering for breaking barriers in lignocellulosic bioethanol production. *Renewable and Sustainable Energy Reviews*, 74:1080–1107, 2017.

[70] Vetle Simensen, André Voigt, and Eivind Almaas. High-quality genome-scale metabolic model of aurantiochytrium sp. t66. *Biotechnology and Bioengineering*, 118:2105–2117, 5 2021.

[71] Sean M. Tibbetts, Mark A. Scaife, and Roberto E. Armenta. Apparent digestibility of proximate nutrients, energy and fatty acids in nutritionally-balanced diets with partial or complete replacement of dietary fish oil with microbial oil from a novel schizochytrium sp. (t18) by juvenile atlantic salmon (salmo salar l.). *Aquaculture*, 520:735003, 4 2020.

[72] Minmin Wei, Christopher C. Parrish, Nigel I. Guerra, Roberto E. Armenta, and Stefanie M. Colombo. Extracted microbial oil from a novel schizochytrium sp. (t18) as a sustainable high dha source for atlantic salmon feed: Impacts on growth and tissue lipids. *Aquaculture*, 534:736249, 3 2021.

[73] Kangsan Kim, Donghui Choe, Yoseb Song, Minjeong Kang, Seung-Goo Lee, Dae-Hee Lee, and Byung-Kwan Cho. Engineering bacteroides thetaiotaomicron to produce non-native butyrate based on a genome-scale metabolic model-guided design. *Metabolic Engineering*, 68:174–186, 2021.

[74] Xin Fang, Colton J Lloyd, and Bernhard O Palsson. Reconstructing organisms in silico: genome-scale models and their emerging applications. *Nature Reviews Microbiology*, 18(12):731–743, 2020.

[75] Ali Ebrahim, Joshua A. Lerman, Bernhard O. Palsson, and Daniel R. Hyduke. Cobrapy: Constraints-based reconstruction and analysis for python. *BMC Systems Biology*, 7:1–6, 8 2013.

[76] José P. Faria, Miguel Rocha, Isabel Rocha, and Christopher S. Henry. Methods for automated genome-scale metabolic model reconstruction. *Biochemical Society Transactions*, 46:931–936, 2018.

[77] Méziane Aite, Marie Chevallier, Clémence Frioux, Camille Trottier, Jeanne Got, María Paz Cortés, Sebastián N. Mendoza, Grégory Carrier, Olivier Dameron, Nicolas Guillaudeux, Mauricio Latorre, Nicolás Loira, Gabriel V. Markov, Alejandro Maass, and Anne Siegel. Traceability, reproducibility and wiki-exploration for "à-la-carte" reconstructions of genome-scale metabolic models. *PLoS Computational Biology*, 14:1–25, 5 2018.

[78] Wai Kit Ong, Peter E Midford, and Peter D Karp. Taxonomic weighting improves the accuracy of a gap-filling algorithm for metabolic models. *Bioinformatics*, 36:1823–1830, 3 2020.

[79] Daniel Machado, Sergej Andrejev, Melanie Tramontano, and Kiran Raosaheb Patil. Fast automated reconstruction of genome-scale metabolic models for microbial species and communities. *Nucleic Acids Research*, 46:7542–7553, 9 2018.

[80] Hao Wang, Simonas Marcišauskas, Benjamín J Sánchez, Iván Domenzain, Daniel Hermansson, Rasmus Agren, Jens Nielsen, and Eduard J Kerkhoven. Raven 2.0: A versatile toolbox for metabolic network reconstruction and a case study on streptomyces coelicolor. *PLoS computational biology*, 14(10):e1006541, 2018.

[81] Emil Karlsen, Christian Schulz, and Eivind Almaas. Automated generation of genome-scale metabolic draft reconstructions based on kegg. *BMC bioinformatics*, 19(1):1–11, 2018.

[82] Michael Mundy, Helena Mendes-Soares, and Nicholas Chia. Mackinac: a bridge between modelseed and cobrapy to generate and analyze genome-scale metabolic models. *Bioinformatics*, 33(15):2416–2418, 2017.

[83] Joost Boele, Brett G Olivier, and Bas Teusink. Fame, the flux analysis and modeling environment. *BMC systems biology*, 6:1–5, 2012.

[84] Wheaton L. Schroeder and Rajib Saha. Optfill: A tool for infeasible cycle-free gapfilling of stoichiometric metabolic models. *iScience*, 23:100783, 1 2020.

[85] Burkhard Rost. Enzyme function less conserved than anticipated. *Journal of Molecular Biology*, 318:595–608, 2002.

[86] Girum Fitihamlak Ejigu and Jaehee Jung. Review on the computational genome annotation of sequences obtained by next-generation sequencing, 9 2020.

[87] Norsigian CJ, Pusarla N, McConn JL, Yurkovich JT, Dräger A, Palsson BO, and King Z. Bigg models 2020: multi-strain genome-scale models and expansion across the phylogenetic tree. *Nucleic acids research*, 48:D402–D406, 1 2020.

[88] Eun-Youn Kim, Daniel Ashlock, and Sung Ho Yoon. Identification of critical connectors in the directed reaction-centric graphs of microbial metabolic networks. *BMC Bioinformatics*, 20:328, 12 2019.

[89] Zachary A. King, Justin Lu, Andreas Dräger, Philip Miller, Stephen Federowicz, Joshua A. Lerman, Ali Ebrahim, Bernhard O. Palsson, and Nathan E. Lewis. Bigg models: A platform for integrating, standardizing and sharing genome-scale models. *Nucleic Acids Research*, 44:D515–D522, 2016.

[90] Eddy J. Bautista, Joseph Zinski, Steven M. Szczepanek, Erik L. Johnson, Edan R. Tulman, Wei Mei Ching, Steven J. Geary, and Ranjan Srivastava. Semi-automated curation of metabolic models via flux balance analysis: A case study with mycoplasma gallisepticum. *PLoS Computational Biology*, 9:1003208, 9 2013.

[91] M. L. Green and P. D. Karp. Genome annotation errors in pathway databases due to semantic ambiguity in partial ec numbers. *Nucleic Acids Research*, 33:4035, 2005.

[92] Tolutola Oyetunde, Muhan Zhang, Yixin Chen, Yinjie Tang, and Cynthia Lo. Boostgapfill: improving the fidelity of metabolic network reconstructions through integrated constraint and pattern-based methods. *Bioinformatics*, 33(4):608–611, 2017.

[93] Wheaton L. Schroeder and Rajib Saha. Optfill: A tool for infeasible cycle-free gapfilling of stoichiometric metabolic models. *iScience*, 23(1):100783, 2020.

[94] Edward Vitkin and Tomer Shlomi. Mirage: a functional genomics-based approach for metabolic network model reconstruction and its application to cyanobacteria networks. *Genome biology*, 13:R111, 2012.

[95] Sylvain Prigent, Clémence Frioux, Simon M. Dittami, Sven Thiele, Abdelhalim Larhlimi, Guillaume Collet, Fabien Gutknecht, Jeanne Got, Damien Eveillard, Jérémie Bourdon, Frédéric Plewniak, Thierry Tonon, and Anne Siegel. Meneco, a topology-based gap-filling tool applicable to degraded genome-wide metabolic networks. *PLOS Computational Biology*, 13:e1005276, 1 2017.

[96] Matthew L Jenior, Emma M Glass, and Jason A Papin. Reconstructor: a cobrapy compatible tool for automated genome-scale metabolic network reconstruction with parsimonious flux-based gap-filling. *Bioinformatics*, 39(6):btad367, 2023.

[97] Charles J. Norsigian, Xin Fang, Yara Seif, Jonathan M. Monk, and Bernhard O. Palsson. A workflow for generating multi-strain genome-scale metabolic models of prokaryotes. *Nature Protocols*, 15:1–14, 2020.

[98] Chao Ye, Weihua Qiao, Xiaobin Yu, Xiaojun Ji, He Huang, Jackie L. Collier, and Liming Liu. Reconstruction and analysis of the genome-scale metabolic model of schizochytrium limacinum sr21 for docosahexaenoic acid production. *BMC Genomics*, 16:1–11, 12 2015.

[99] Valcenir Júnior Mendes Furlan, Irineu Batista, Narcisa Bandarra, Rogério Mendes, and Carlos Cardoso. Conditions for the production of carotenoids by thraustochytrium sp. atcc 26185 and aurantiochytrium sp. atcc pra-276. *Journal of Aquatic Food Product Technology*, 28:465–477, 5 2019.

[100] Dauenpen Meesapyodsuk and Xiao Qiu. Biosynthetic mechanism of very long chain polyunsaturated fatty acids in thraustochytrium sp . *Journal Lipid Research*, 57:1854–1864, 2016.

[101] Gabriel M. Rodriguez, Murtaza Shabbir Hussain, Lauren Gambill, Difeng Gao, Allison Yaguchi, and Mark Blenner. Biotechnology for biofuels engineering xylose utilization in yarrowia lipolytica by understanding its cryptic xylose pathway. *Biotechnology for Biofuels*, 9:1–15, 7 2016.

[102] Nele Buschke, Judith Becker, Rudolf Schäfer, Patrick Kiefer, Rebekka Biedendieck, and Christoph Wittmann. Systems metabolic engineering of xylose-utilizing corynebacterium glutamicum for production of 1, 5-diaminopentane. *Biotechnology journal*, 8(5):557–570, 2013.

[103] Weijun Luo and Cory Brouwer. Pathview: an r/bioconductor package for pathway-based data integration and visualization. *Bioinformatics*, 29(14):1830–1831, 2013.

[104] Zachary A King, Andreas Dräger, Ali Ebrahim, Nikolaus Sonnenschein, Nathan E Lewis, and Bernhard O Palsson. Escher: a web application for building, sharing, and embedding data-rich visualizations of biological pathways. *PLoS computational biology*, 11(8):e1004321, 2015.

[105] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using networkx. In Gaël Varoquaux, Travis Vaught, and Jarrod Millman, editors, *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA, 2008.

[106] Alexandra M. Schnoes, Shoshana D. Brown, Igor Dodevski, and Patricia C. Babbitt. Annotation error in public databases: Misannotation of molecular function in enzyme superfamilies. *PLoS Computational Biology*, 5, 2009.

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

[108] Jeffrey D. Orth and Bernhard Palsson. Systematizing the generation of missing metabolic knowledge. *Biotechnology and Bioengineering*, 107:403–412, 10 2010.

[109] Shu Pan and Jennifer L. Reed. Advances in gap-filling genome-scale metabolic models and model-driven experiments lead to novel metabolic discoveries. *Current Opinion in Biotechnology*, 51:103–108, 6 2018.

[110] Sebastián N. Mendoza, Brett G. Olivier, Douwe Molenaar, and Bas Teusink. A systematic assessment of current genome-scale metabolic reconstruction tools. *Genome Biology*, 20:1–20, 8 2019.

[111] Hugh M. Purdy and Jennifer L. Reed. Evaluating the capabilities of microbial chemical production using genome-scale metabolic models. *Current Opinion in Systems Biology*, 2:91–97, 2017.

[112] James G. Ferry. Methanosarcina acetivorans: A model for mechanistic understanding of aceticlastic and reverse methanogenesis. *Frontiers in Microbiology*, 11:1–12, 2020.

[113] Jiazhang Lian, Shekhar Mishra, and Huimin Zhao. Recent advances in metabolic engineering of saccharomyces cerevisiae: New tools and their applications. *Metabolic Engineering*, 50:85–108, 2018.

[114] Hao Wang, Jonathan L. Robinson, Pinar Kocabas, Johan Gustafsson, Mihail Anton, Pierre Etienne Cholley, Shan Huang, Johan Gobom, Thomas Svensson, Mattias Uhlen, Henrik Zetterberg, and Jens Nielsen. Genome-scale metabolic network reconstruction of model animals as a platform for translational research. *Proceedings of the National Academy of Sciences of the United States of America*, 118:e2102344118, 7 2021.

[115] Sjoerd Opdam, Anne Richelle, Benjamin Kellman, Shanzhong Li, Daniel C. Zielinski, and Nathan E. Lewis. A systematic evaluation of methods for tailoring genome-scale metabolic models. *Cell Systems*, 4:318–329.e6, 2017.

[116] Kenshi Watanabe, Charose Marie, Ting Perez, Tomoki Kitahori, Kosuke Hata, Masato Aoi, Hirokazu Takahashi, Tetsushi Sakuma, Yoshiko Okamura, Yutaka Nakashimada, Takashi Yamamoto, Keisuke Matsuyama, Shinzo Mayuzumi, and Tsunehiro Aki. Improvement of fatty acid productivity of thraustochytrid , aurantiochytrium sp . by genome editing. *Journal of Bioscience and Bioengineering*, 131:373–380, 2021.

[117] Rosemary Braun. Systems analysis of high–throughput data. *Advances in experimental medicine and biology*, 844:153, 2014.

[118] Daniel Gamermann, Arnaud Montagud, J. Alberto Conejero, Javier F. Urchueguía, and Pedro Fernández de Córdoba. New approach for phylogenetic tree recovery based on genome-scale metabolic networks. *Journal of Computational Biology*, 21:508–519, 7 2014.

[119] Daniel Gamermann, Arnau Montagud, J. Alberto Conejero, Pedro Fernández de Córdoba, and Javier F. Urchueguía. Large scale evaluation of differences between network-based and pairwise sequence-alignment-based methods of dendrogram reconstruction. *PLoS ONE*, 14:1–13, 2019.

[120] William H Jeeerys, James O Berger, and William H Jefferys. Sharpening ockham's razor on a bayesian strop key terms: Bayes' theorem; ockham's razor, 1991.

[121] Ziheng Yang. User guide paml : Phylogenetic analysis by maximum likelihood. *Molecular Biology and Evolution*, 4:1–70, 2007.

[122] David B. Bernstein, Snorre Sulheim, Eivind Almaas, and Daniel Segrè. Addressing uncertainty in genome-scale metabolic model reconstruction and analysis, 12 2021.

[123] Mario Latendresse and Peter D Karp. Evaluation of reaction gap-filling accuracy by randomization. *BMC bioinformatics*, 19:1–13, 2018.

Appendix A

Supplementary Information for Chapter 5

```
# -*- coding: utf-8 -*-
"""
@author: Leila Rezaei
"""
from cobra.io import read_sbml_model, write_sbml_model
from biggmodeldata import metabolites as m
import cobra
import glob
import json
from cobra import Model, Reaction, Metabolite
import pickle
from collections import defaultdict
import csv
import random
import pathlib
from pathlib import Path
import os
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import pathlib
import matplotlib.pyplot as plt
import math
import re
import pandas as pd
import numpy as np
from .sbprank import rank_reactions as _rank_reactions

""" Unpack COBRA model into stoichiometric tuples. """
def _unpack(model):

    coefficients = []

    for r in model.reactions:
        for m, c in r.metabolites.items():
            coefficients.append((r.id, m.id, c))

    return coefficients

def rank_reactions(model, target, blocked):

    # Converting models
    model = _unpack(model)
    target = _unpack(target)

    scores = _rank_reactions(target, model, blocked)

    return pd.DataFrame.from_dict(scores, orient = "index")
```

```
"""The main function in sbprank package that score reactions"""
def reactions_score(target, reference_universal, blocked = None,
                    gap_percentage = None, threshold = 100):

    exclude_list = m.helpers()
    a = read_sbml_model(target)

    ref_models_name = []
    for filename in os.listdir(reference_universal):
        ref_models_name.append(filename)
    if blocked == None:
        blocked_reactions = cobra.flux_analysis.find_blocked_reactions(a)
        d = set()
        for r in blocked_reactions:
            mets = a.reactions.get_by_id(r).metabolites
            for met in mets:
                d.add(met.id)
    else:
        d = []
        filepath = os.path.join(blocked,'sample_blocked.csv')
        with open(filepath,'r',newline = '') as csvfile:
            for row in csv.reader(csvfile):
                d.append(row[0])

    print(d)
    dead_end = []
```

```
for dd in d:
    if dd not in exclude_list:
        dead_end.append(dd)
for model in ref_models_name:
    filepath = os.path.join(reference_universal,model)
    r_to_df = {}
    b = read_sbml_model(filepath)
    target_metabolites = set([mc.id for mc in b.metabolites])
    blocked = [ma for ma in dead_end if ma in target_metabolites]
    df = rank_reactions(b, a, blocked)
    r_to_df[model] = df
    nnew_path = os.getcwd()
    new_dir_name = os.path.join(nnew_path,'metrics_calulation_result')
    new_dir = pathlib.Path(new_dir_name)
    new_dir.mkdir(parents = True, exist_ok = True)
    with open(new_dir/f'{model}.pkl', 'wb') as f:
        pickle.dump(r_to_df, f)

def filter_reactions(new_df):
    reactionlist = list(new_df.index)
    # empty = []
    # for rec in reactionlist:
    #     if not r.reactants(rec) and not r.products(rec):
    #         empty.append(rec)
    # for item in empty:
    #     reactionlist.remove(item)
    pattern = "^EX_"
```

```
remove_list = []

for item in reactionlist:

    if re.match(pattern,item):

        remove_list.append(item)


pattern = "^DM_"


remove_list2 = []

for item in reactionlist:

    if re.match(pattern,item):

        remove_list2.append(item)


pattern = "^SK_"


remove_list3 = []

for item in reactionlist:

    if re.match(pattern,item):

        remove_list3.append(item)


pattern = "^BIOMASS_"


remove_list4 = []

for item in reactionlist:

    if re.match(pattern,item):

        remove_list4.append(item)
    remove_list5 = [x for x in reactionlist if x.endswith('_ST')]
    remove_list6 = [x for x in reactionlist if x.endswith('_copy1')]
    remove_list7 = [x for x in reactionlist if x.endswith('_copy2')]
    remove_lst = remove_list + remove_list2 + remove_list3
    +remove_list4 + remove_list5 + remove_list6 + remove_list7
    for indexx in new_df.index:
        if indexx in remove_lst:
            new_df = new_df.drop(indexx)
    return(new_df)


df_lst = []
files = Path(new_dir).glob('*')
for file in files:
    df_lst.append(file)
#print(df_lst)
dfs = []
for p in df_lst:
    #print(p)
    with open(p, 'rb') as f:
        df = pickle.load(f)
        dfs.append(df)
    #df_final = df.dropna()

values = []
for i in range(len(dfs)):
    for k,v in dfs[i].items():
        values.append(v)

v_final = []
for i in range(len(values)):
    for k,v in values[i].items():
        v_final.append(v)

resultt = pd.concat([f for f in v_final], axis=1)
result = filter_reactions(resultt)
n=float('NaN')

#betweenness calculation
b2 = result['betweenness']
b2[b2 > 100] = n
makh = (max(b2.max())-min(b2.min()))
sor = b2-min(b2.min())
nrmb = sor/makh
qub= nrmb.quantile([.9, .95, 0.99, 1], axis=1)
bett = np.transpose(qub)
bet = bett.dropna()
betsort = bet[1].sort_values(ascending = False)
#betsort.index.get_loc('FBA')

#similarity calculation
ss = result['similarity']
ss[ss >100] = n
makhss = (max(ss.max())-min(ss.min()))
```

```
sorss = ss-min(ss.min())
nrmss = sorss/makhss
quss= nrmss.quantile([.9, .95, 0.99, 1], axis=1)
simm = np.transpose(quss)
sim = simm.dropna()
simsort = sim[1].sort_values(ascending = False)
#simsort.index.get_loc('FBA')

#distance calculation
s2 = result['distance']
s2[s2 >100] = n
#maxmdis = (s2/s2.max())
makhc = (max(s2.max())-min(s2.min()))
#d = ts2-ts2.min()
sord = (max(s2.max())-s2)
f = sord/makhc
g= f.quantile([.9, .95, 0.99, 1], axis=1)
diss = np.transpose(g)

dis = diss.dropna()
#dissort.index.get_loc('FBA')
dissort = dis[1].sort_values(ascending = False)
#return(dissort,simsort,betsort)

if gap_percentage == None:

    final_df= pd.concat([dissort,simsort,betsort], axis=1)
```

```
cols = []
count = 1
for column in final_df.columns:
    if column == 1:
        cols.append(f'1_{count}')
        count+=1
        continue
    cols.append(column)
final_df.columns = cols

#pca analysis
dfss_final= final_df.dropna()
df_st =  StandardScaler().fit_transform(dfss_final)
pca_out = PCA().fit(df_st)
pca_out.explained_variance_ratio_
pca_scores = PCA().fit_transform(df_st)
#maximum_scores = np.max(pca_scores, axis=0)
newdf = pd.DataFrame(pca_scores, columns=dfss_final.columns,
                     index=dfss_final.index)
maxvals = []
minvals = []
for col in newdf.columns:
    clo1 = newdf[col]
    numlist = []
    for k,v in clo1.items():
        numlist.append(clo1[k])
        positive = [i for i in numlist if i >0]
        negative = [i for i in numlist if i <0]
        if max(positive) > abs(min(negative)):
            maxvals.append(max(positive))
        else:
            maxvals.append(abs(min(negative)))
            minvals.append(min(negative))
    #print(minvals)
    new_df= pd.concat([dfss_final['1_1']*maxvals[0],dfss_final['1_2']
                       *maxvals[1],dfss_final['1_3']*maxvals[2]], axis=1)
elif gap_percentage <= 25:
    weights = [0.8, 0.1, 0.1]
elif 30 <= gap_percentage <= 50:
    weights = [0.1, 0.8, 0.1]
elif 55 <= gap_percentage <= 85:
    weights = [0.1, 0.6, 0.3]
else:
   weights = [0.1, 0.3, 0.6]
#def final(dissort,simsort,betsort,weights):
final_df= pd.concat([dissort,simsort,betsort], axis=1)

cols = []
count = 1
for column in final_df.columns:
    if column == 1:
        cols.append(f'1_{count}')
        count+=1
        continue
    cols.append(column)
final_df.columns = cols
w_distance = weights[0]
w_similarity = weights[1]
w_betweenness = weights[2]
dfss_final= final_df.dropna()
new_df= sum([dfss_final['1_1']*w_distance,dfss_final['1_2']*w_similarity,
             dfss_final['1_3']*w_betweenness])
new_df_sort = new_df.sort_values(ascending = False)
new_df = pd.DataFrame(new_df_sort)
new_df_size = len(new_df)
top_percent = int(threshold * new_df_size)
top_list = new_df.head(top_percent).index.tolist()
from biggmodeldata import reactions as r
#from biggmodeldata import metabolites as m
model = Model("ranked_universal")

for rid in top_list:
    print(rid)
    reaction= Reaction(rid)
    reaction.lower_bound= -1000.0
    eqs=(r.reactants(rid)+r.products(rid))
    #print(eqs)
    eqs1= dict((y,x) for x,y in eqs)
    #print(eqs1.keys())
    #print(eqs1)
```

```
        metabolites=[Metabolite(key,compartment= 'c') for key in eqs1]
        model.add_metabolites(metabolites)
        metabolites={Metabolite(key):eqs1[key] for key in eqs1}
        print(metabolites)
        reaction.add_metabolites(metabolites)
        model.add_reactions([reaction])

    unused = cobra.manipulation.prune_unused_reactions(model)
    ids = [rr.id for rr in unused[1]]

    for i in ids:
        ii = model.reactions.get_by_id(i)
        model.remove_reactions(ii)
    write_sbml_model(model, 'ranked_universal.xml')
        #return(new_df)
    return(model)
# -*- coding: utf-8 -*-
"""

@author: Leila Rezaei
"""
import cobra
from cobra.io import read_sbml_model, write_sbml_model
import pandas as pd
import numpy as np
import pickle
import glob
import random
import matplotlib.pyplot as plt
import plotnine
from plotnine import *

#Example code for plots in chapter 5
df_lst = []
for filename in glob.iglob('*.pkl',recursive=True):
    df_lst.append(filename)

dfs = []
for p in df_lst:
    print(p)
    with open(p, 'rb') as f:
        df = pickle.load(f)
        dfs.append(df)
r_to_n = {}
for d in dfs:
    r_to_n.update(d)
df_lst = []
for filename in glob.iglob('*.pkl',recursive=True):
    df_lst.append(filename)



dfs = []
for p in df_lst:
    print(p)
    with open(p, 'rb') as f:
        df = pickle.load(f)
        dfs.append(df)
r_to_n2 = {}
for d in dfs:
    r_to_n2.update(d)

comb = {}
for key in r_to_n.keys():
    comb[key] = (r_to_n[key],r_to_n2[key])



# Prepare the data for the boxplot
box_data = []
labels = []

for key, (list1, list2) in comb.items():
    box_data.extend([list1, list2])
    labels.extend([f'{key}_pca', f'{key}_w'])

# Create the boxplot
plt.figure(figsize=(12, 6))
sns.boxplot(data=box_data)
plt.xticks(range(len(labels)), labels)
plt.xlabel('Percent of reactions')
plt.ylabel('Percentage of universal pool')
plt.title('Boxplot for comparison of PCA & weighted average')
plt.show()



#Plotting by ggplot2
lst1 = r_to_n[10]
x = np.array(["PCA"])
name= np.repeat(x, [183], axis=0)
df1 = pd.DataFrame({'Ranking method':name,
                    '% of the ranked universal pool':lst1})

lst2 = r_to_n2[10]
x = np.array(["Heuristic weights"])
name= np.repeat(x, [75], axis=0)
df2 = pd.DataFrame({'Ranking method':name,
                    '% of the ranked universal pool':lst2})

result = pd.concat([df1, df2], axis=0)
x = np.array(['10'])
name= np.repeat(x, [258], axis=0)
result['Percent of essential reactions'] = name

lst1 = r_to_n[15]
x = np.array(["PCA"])
name= np.repeat(x, [185], axis=0)
df1 = pd.DataFrame({'Ranking method':name,
                    '% of the ranked universal pool':lst1})
```

```
lst2 = r_to_n2[15]

x = np.array(["Heuristic weights"])

name= np.repeat(x, [75], axis=0)

df2 = pd.DataFrame({'Ranking method':name,

                    '% of the ranked universal pool':lst2})


result2 = pd.concat([df1, df2], axis=0)

x = np.array(['15'])

name= np.repeat(x, [260], axis=0)

result2['Percent of essential reactions'] = name


lst1 = r_to_n[20]

x = np.array(["PCA"])

name= np.repeat(x, [183], axis=0)

df1 = pd.DataFrame({'Ranking method':name,

                    '% of the ranked universal pool':lst1})


lst2 = r_to_n2[20]

x = np.array(["Heuristic weights"])

name= np.repeat(x, [97], axis=0)

df2 = pd.DataFrame({'Ranking method':name,

                    '% of the ranked universal pool':lst2})


result3 = pd.concat([df1, df2], axis=0)

x = np.array(['20'])

name= np.repeat(x, [280], axis=0)

result3['Percent of essential reactions'] = name
```

```
#result13 = pd.concat([result, result2], axis=0)
lst1 = r_to_n[25]
x = np.array(["PCA"])
name= np.repeat(x, [185], axis=0)
df1 = pd.DataFrame({'Ranking method':name,
                    '% of the ranked universal pool':lst1})

lst2 = r_to_n2[25]
x = np.array(["Heuristic weights"])
name= np.repeat(x, [97], axis=0)
df2 = pd.DataFrame({'Ranking method':name,
                    '% of the ranked universal pool':lst2})

result4 = pd.concat([df1, df2], axis=0)
x = np.array(['25'])
name= np.repeat(x, [282], axis=0)
result4['Percent of essential reactions'] = name
lst1 = r_to_n[30]
x = np.array(["PCA"])
name= np.repeat(x, [234], axis=0)
df1 = pd.DataFrame({'Ranking method':name,
                    '% of the ranked universal pool':lst1})

lst2 = r_to_n2[30]
x = np.array(["Heuristic weights"])
name= np.repeat(x, [250], axis=0)
df2 = pd.DataFrame({'Ranking method':name,
                    '% of the ranked universal pool':lst2})


result5 = pd.concat([df1, df2], axis=0)
x = np.array(['30'])
name= np.repeat(x, [484], axis=0)
result5['Percent of essential reactions'] = name
#result4 = pd.concat([result13, result2], axis=0)
lst1 = r_to_n[35]
x = np.array(["PCA"])
name= np.repeat(x, [210], axis=0)
df1 = pd.DataFrame({'Ranking method':name,
                    '% of the ranked universal pool':lst1})


lst2 = r_to_n2[35]
x = np.array(["Heuristic weights"])
name= np.repeat(x, [286], axis=0)
df2 = pd.DataFrame({'Ranking method':name,
                    '% of the ranked universal pool':lst2})


result6 = pd.concat([df1, df2], axis=0)
x = np.array(['35'])
name= np.repeat(x, [496], axis=0)
result6['Percent of essential reactions'] = name
lst1 = r_to_n[40]
x = np.array(["PCA"])
name= np.repeat(x, [183], axis=0)
df1 = pd.DataFrame({'Ranking method':name,
                    '% of the ranked universal pool':lst1})

lst2 = r_to_n2[40]
x = np.array(["Heuristic weights"])
name= np.repeat(x, [250], axis=0)
df2 = pd.DataFrame({'Ranking method':name,
                    '% of the ranked universal pool':lst2})

result7 = pd.concat([df1, df2], axis=0)
x = np.array(['40'])
name= np.repeat(x, [433], axis=0)
result7['Percent of essential reactions'] = name
#result5 = pd.concat([result4, result2], axis=0)
lst1 = r_to_n[45]
x = np.array(["PCA"])
name= np.repeat(x, [185], axis=0)
df1 = pd.DataFrame({'Ranking method':name,
                    '% of the ranked universal pool':lst1})

lst2 = r_to_n2[45]
x = np.array(["Heuristic weights"])
name= np.repeat(x, [266], axis=0)
df2 = pd.DataFrame({'Ranking method':name,
                    '% of the ranked universal pool':lst2})

result8 = pd.concat([df1, df2], axis=0)
x = np.array(['45'])
```

```
name= np.repeat(x, [451], axis=0)
result8['Percent of essential reactions'] = name
lst1 = r_to_n[50]
x = np.array(["PCA"])
name= np.repeat(x, [191], axis=0)
df1 = pd.DataFrame({'Ranking method':name,
                    '% of the ranked universal pool':lst1})

lst2 = r_to_n2[50]
x = np.array(["Heuristic weights"])
name= np.repeat(x, [179], axis=0)
df2 = pd.DataFrame({'Ranking method':name,
                    '% of the ranked universal pool':lst2})

result9 = pd.concat([df1, df2], axis=0)
x = np.array(['50'])
name= np.repeat(x, [370], axis=0)
result9['Percent of essential reactions'] = name
#result6 = pd.concat([result5, result2], axis=0)
lst1 = r_to_n[55]
x = np.array(["PCA"])
name= np.repeat(x, [163], axis=0)
df1 = pd.DataFrame({'Ranking method':name,
                    '% of the ranked universal pool':lst1})

lst2 = r_to_n2[55]
x = np.array(["Heuristic weights"])
```

```
name= np.repeat(x, [191], axis=0)
df2 = pd.DataFrame({'Ranking method':name,
                    '% of the ranked universal pool':lst2})

result10 = pd.concat([df1, df2], axis=0)
x = np.array(['55'])
name= np.repeat(x, [354], axis=0)
result10['Percent of essential reactions'] = name

lst1 = r_to_n[60]
x = np.array(["PCA"])
name= np.repeat(x, [163], axis=0)
df1 = pd.DataFrame({'Ranking method':name,
                    '% of the ranked universal pool':lst1})

lst2 = r_to_n2[60]
x = np.array(["Heuristic weights"])
name= np.repeat(x, [198], axis=0)
df2 = pd.DataFrame({'Ranking method':name,
                    '% of the ranked universal pool':lst2})

result11 = pd.concat([df1, df2], axis=0)
x = np.array(['60'])
name= np.repeat(x, [361], axis=0)
result11['Percent of essential reactions'] = name

lst1 = r_to_n[65]
```

```
x = np.array(["PCA"])
name= np.repeat(x, [167], axis=0)
df1 = pd.DataFrame({'Ranking method':name,
                    '% of the ranked universal pool':lst1})

lst2 = r_to_n2[65]
x = np.array(["Heuristic weights"])
name= np.repeat(x, [206], axis=0)
df2 = pd.DataFrame({'Ranking method':name,
                    '% of the ranked universal pool':lst2})

result12 = pd.concat([df1, df2], axis=0)
x = np.array(['65'])
name= np.repeat(x, [373], axis=0)
result12['Percent of essential reactions'] = name

lst1 = r_to_n[70]
x = np.array(["PCA"])
name= np.repeat(x, [167], axis=0)
df1 = pd.DataFrame({'Ranking method':name,
                    '% of the ranked universal pool':lst1})

lst2 = r_to_n2[70]
x = np.array(["Heuristic weights"])
name= np.repeat(x, [206], axis=0)
df2 = pd.DataFrame({'Ranking method':name,
                    '% of the ranked universal pool':lst2})
```

```
result13 = pd.concat([df1, df2], axis=0)
x = np.array(['70'])
name= np.repeat(x, [373], axis=0)
result13['Percent of essential reactions'] = name

lst1 = r_to_n[75]
x = np.array(["PCA"])
name= np.repeat(x, [167], axis=0)
df1 = pd.DataFrame({'Ranking method':name,
                    '% of the ranked universal pool':lst1})

lst2 = r_to_n2[75]
x = np.array(["Heuristic weights"])
name= np.repeat(x, [206], axis=0)
df2 = pd.DataFrame({'Ranking method':name,
                    '% of the ranked universal pool':lst2})

result14 = pd.concat([df1, df2], axis=0)
x = np.array(['75'])
name= np.repeat(x, [373], axis=0)
result14['Percent of essential reactions'] = name

lst1 = r_to_n[80]
x = np.array(["PCA"])
name= np.repeat(x, [167], axis=0)
df1 = pd.DataFrame({'Ranking method':name,
                    '% of the ranked universal pool':lst1})

lst2 = r_to_n2[80]
x = np.array(["Heuristic weights"])
name= np.repeat(x, [206], axis=0)
df2 = pd.DataFrame({'Ranking method':name,
                    '% of the ranked universal pool':lst2})

result15 = pd.concat([df1, df2], axis=0)
x = np.array(['80'])
name= np.repeat(x, [373], axis=0)
result15['Percent of essential reactions'] = name

lst1 = r_to_n[85]
x = np.array(["PCA"])
name= np.repeat(x, [167], axis=0)
df1 = pd.DataFrame({'Ranking method':name,
                    '% of the ranked universal pool':lst1})

lst2 = r_to_n2[85]
x = np.array(["Heuristic weights"])
name= np.repeat(x, [234], axis=0)
df2 = pd.DataFrame({'Ranking method':name,
                    '% of the ranked universal pool':lst2})

result16 = pd.concat([df1, df2], axis=0)
x = np.array(['85'])
```

```
name= np.repeat(x, [401], axis=0)
result16['Percent of essential reactions'] = name

lst1 = r_to_n[90]
x = np.array(["PCA"])
name= np.repeat(x, [183], axis=0)
df1 = pd.DataFrame({'Ranking method':name,
                    '% of the ranked universal pool':lst1})

lst2 = r_to_n2[90]
x = np.array(["Heuristic weights"])
name= np.repeat(x, [234], axis=0)
df2 = pd.DataFrame({'Ranking method':name,
                    '% of the ranked universal pool':lst2})

result17 = pd.concat([df1, df2], axis=0)
x = np.array(['90'])
name= np.repeat(x, [417], axis=0)
result17['Percent of essential reactions'] = name

lst1 = r_to_n[95]
x = np.array(["PCA"])
name= np.repeat(x, [183], axis=0)
df1 = pd.DataFrame({'Ranking method':name,
                    '% of the ranked universal pool':lst1})

lst2 = r_to_n2[95]
```

```
x = np.array(["Heuristic weights"])
name= np.repeat(x, [189], axis=0)
df2 = pd.DataFrame({'Ranking method':name,
                    '% of the ranked universal pool':lst2})

result18 = pd.concat([df1, df2], axis=0)
x = np.array(['95'])
name= np.repeat(x, [372], axis=0)
result18['Percent of essential reactions'] = name

lst1 = r_to_n[100]
x = np.array(["PCA"])
name= np.repeat(x, [183], axis=0)
df1 = pd.DataFrame({'Ranking method':name,
                    '% of the ranked universal pool':lst1})

lst2 = r_to_n2[100]
x = np.array(["Heuristic weights"])
name= np.repeat(x, [160], axis=0)
df2 = pd.DataFrame({'Ranking method':name,
                    '% of the ranked universal pool':lst2})

result19 = pd.concat([df1, df2], axis=0)
x = np.array(['100'])
name= np.repeat(x, [343], axis=0)
result19['Percent of essential reactions'] = name
```

```
resulttotal = pd.concat([result, result2, result3,
                         result4, result5, result6,
                         result7, result8, result9, result10,
                         result11, result12, result13, result14,
                         result15, result16, result17,
                         result18,result19], axis=0)
x_labels = [str(x) for x in range(10, 101, 5)]
x_labels.remove('100')
x_labels.append('100')


resulttotal['Percent of essential reactions'] = \
    pd.Categorical(resulttotal['Percent of essential reactions'], \
    categories=x_labels, ordered=True)



p10 = (
    ggplot(resulttotal, aes("Percent of essential reactions",
                            "% of the ranked universal pool",
                            fill="Ranking method"))+ geom_boxplot()
    + xlab("Percent of essential reactions")
    + ylab("% of the ranked universal pool")
    + scale_y_continuous(breaks=np.arange(0, 70, 5),
                          limits=[0, 70])


    + ggtitle("Comparison of PCA & Heuristic weights")
    + theme(
        legend_direction="horizontal",
        legend_box_spacing=0.4,
        axis_line=element_line(size=1, colour="black"),
        panel_grid_major=element_line(colour="#d3d3d3"),
        panel_grid_minor=element_blank(),
        panel_border=element_blank(),
        panel_background=element_blank(),
        plot_title=element_text(size=15, family="Tahoma",
                                face="bold"),
        text=element_text(family="Tahoma", size=11),
        axis_text_x=element_text(colour="black", size=11),
        axis_text_y=element_text(colour="black", size=11),
    )
    + scale_fill_brewer(type="qual", palette="Set2")
)
p10
```