UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Máster Universitario en Bioinformática y Biología Computacional

TRABAJO FIN DE MÁSTER

Mejora en la generación automática de modelos metabólicos (GEM) de organismos relevantes en la industria alimentaria mediante *gap filling*

Francisco Manuel Muñoz López

DIRECTORES

Juan Nogales Enrique

David San León Granado

CENTRO NACIONAL DE BIOTECNOLOGÍA

DEPARTAMENTO DE BIOLOGÍA DE SISTEMAS

Junio, 2021

Agradecimientos

Me gustaría agradecer al grupo de Biotecnología de Sistemas del CNB, especialmente a mis directores Juan, que me dio la oportunidad de trabajar en un tema tan interesante como es el modelado metabólico, y David, cuya ayuda ha resultado inestimable durante todo el desarrollo del proyecto.

También quiero darle las gracias a Gonzalo, que además de acceder a ser mi ponente, siempre me ha aconsejado bien en temas administrativos, mostrando mucho interés.

Finalmente, no puedo dejar de mencionar a las personas más importantes para mí, mi pareja, mis padres, mi hermana, y mis amigos más cercanos, porque ellos siempre han sido mi principal apoyo.

Contents

List of Figures

List of Tables

Abstract

Microbes have been used at the production of all kinds of food for thousands of years. Over the recent decades, many technologies have been developed in order to address microbial processes. One of these technologies is metabolic modelling, consisting on the *in-silico* reconstruction of genome-scale metabolic models (GEMs) describing any organism metabolism, which are used for the simulation of metabolic fluxes. A special type of GEM are the multi-strain GEMs, which can be used to study similarities and differences between strains of a species, leading to a variety of applications. However, these models are generated from annotated genomes, which usually contain gaps, resulting in non-functional GEMs. This project aims to design and create a gap filling module, called GGF, for its implementation in Gallant, a workflow for the automated generation of multi-strain GEMs. After the creation and testing of the mentioned module, it will be applied onto a multi-strain GEM made from *Lactococcus lactis*, a microorganism with high importance in dairy industry.

Keywords

Genome-Scale Metabolic Models, Flux Balance Analysis, Gap Filling, COBRA, ModelSEED, BiGG, *Lactococcus lactis*, Food Industry

Introduction

Use of Microorganisms in Food Industry

For thousands of years humankind has used microbes in food matters, so long before the discovery of these organisms (Kapur 2019). Some foods and beverages whose production requires a sort of microorganism are bread, wine, beer, yogurt, blue cheese or pickles.

The main use of microbes in food production is fermentation. This process consists on the partial or full transformation of some substrate present on the untreated food, changing its organoleptic properties. The first kind of fermentation that probably comes to mind is alcoholic fermentation, usually carried out by *Saccharomyces* yeasts, which takes glucose or other sugars to produce ethanol and CO_2. It is used in the production of alcoholic beverages such as wine, beer, mead or sake, but also in the leaven of bread due to the CO_2 generation. Other important types of fermentation are lactic fermentation (yogurt, pickles) and acetic fermentation (vinegar) both carried out by bacteria. Nowadays, there are more than 3500 foods and beverages ((EFFCA)) that go through some kind of fermentation like kombucha tea, sauerkraut or Japanese *natto*, just to name a few.

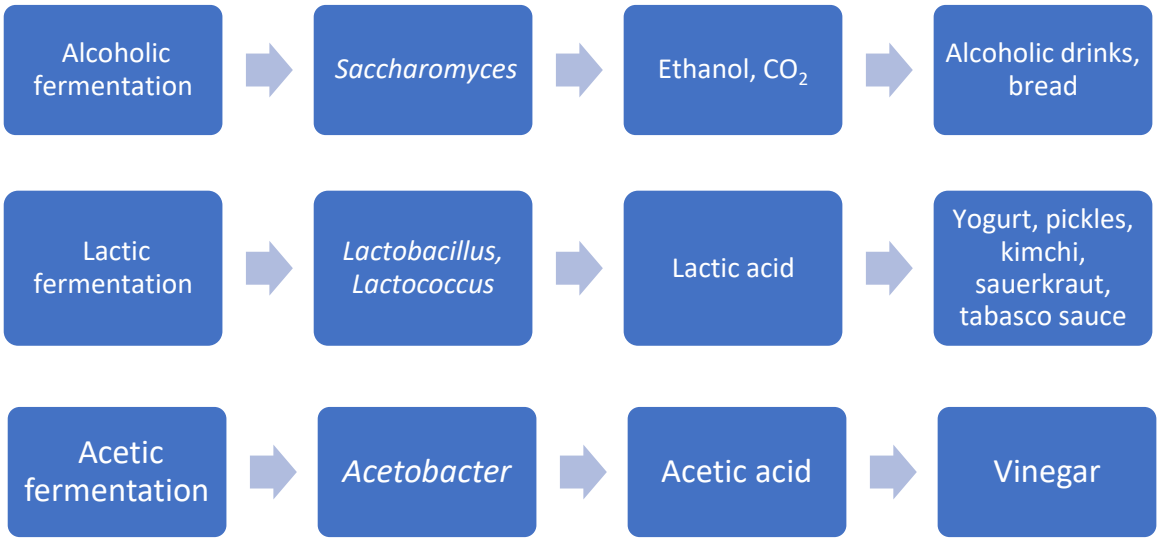

Figure 1. **Most used fermentations**. From left to right it shows fermentation name, involved microorganism, product(s) and resulting food.

Besides fermentation, microbes are useful for changing the flavour profile of the foods as it happens with blue cheese or generating additives like citric acid. Furthermore, some microorganisms can be used as probiotics (e.g. *Lacticaseibacillus casei*) that positively

affects our microbiota (Ercolini and Fogliano 2018) which is known to have a crucial role in our health.

History of Microbes in Food Production

The history of the use of microorganisms in food production can be described by a variety of milestones, highlighting (Kapur 2019):

7000 BC – There are evidences that Babylonians made beer. In that time (and even now in underdeveloped countries) it was safer to drink fermented beverages than water due to the lack of treatment methods.

6000 BC – First reference to food spoilage. Wine appeared in different civilizations like Georgia (6000 BC), Persia (5000 BC) and Sicily (4000 BC). Throughout history, wine has been consumed for its intoxicating effects produced by ethanol.

3000 BC – Egyptians manufactured cheese and butter. As it happened with alcoholic beverages, these fermented foods were safer to eat than their raw counterparts. Around this time people started to use salt as preserver.

1683 – Anton van Leeuwenhoek researched upon bacteria with his own microscopes. He shared his discoveries with the Royal Society (England) for nearly 50 years. As a result, Leeuwenhoek's reports were disseminated and he is considered the person who discovered the microbial world.

1975 – Nicholas Appert developed a method to preserve meat which consisted on placing the food into glass bottles and boiling it. This was the beginning of canning preservation.

1857 – The first person to fully understand the relationship between microorganisms in infusions and the chemical changes that took place in those infusions was Louis Pasteur. Through his experiments, Pasteur convinced the scientific community that all fermentative processes were caused by microorganisms. In 1857, he proved that souring milk was caused by microbes and in 1860, he demonstrated that heat killed those microbes. This heating process is called pasteurization and nowadays is an essential step during food processing. Apart from pasteurization, Pasteur's contributions to microbiology and specifically food microbiology are uncountable. Just to name a few, he definitely refuted spontaneous generation theory, described alcoholic and lactic fermentation, developed germ theory of disease, and so on.

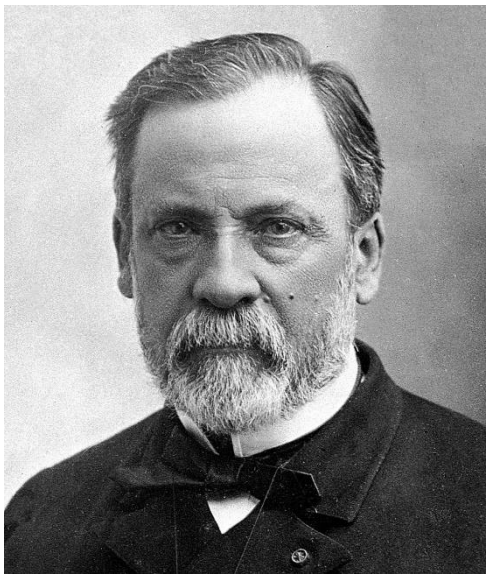

Figure 2 (left). Leeuwenhoek's microscopes replicas (c. 1600).

Figure 3 (right). Louis Pasteur (1880).

Late 1800s – Legislation to protect food quality started to be enacted by several countries.

1920s – In the United States, a big part of food industries remained reluctant to adopt microbiological standards causing several outbreaks of botulism (caused by *Clostridium botulinum*). These incidents led U.S. canning industry to adopt a conservative treatment, known as 12D process, that causes a reduction in the probability of the most heat resistant *C. botulinum* spores to one in a billion.

The use of sterilization standards started by Pasteur and improved over the years is an obligation in every single developed country. Probably the next logical step would be the use of genetically modified microorganisms optimized for fermentation. Currently, the use of GMO is still very controversial, especially in the European Union, but still they are used in a wide variety of applications.

GMO in Food Industry

In some fields like medicine or textile industry, most people seem to accept the use of genetically modified organisms. A good example is the production of insulin for the treatment of diabetes, which is generated by transgenic (microorganisms with human genes) strains of *E. coli* and *S. cerevisiae* (Baeshen, Baeshen *et al.* 2014). Nevertheless, this tendency changes when it comes to GMO food products.

But why are people so afraid of eating genetically modified food products? The majority of scientific community agrees that GMO disavowal by the average citizen is caused by misconception, limited understanding and unfamiliarity with these products (Wunderlich and Gatto 2015). Many consumers report that they receive information about GMO from the media and internet which usually are not as reliable as scientific papers. The ideal solution would be that governments and media lined up and rigorously inform themselves and the population about GMO. However, this is not as easy since there are lots of factors implied and every country (or even region) has its own way of living.

That being said, the number of countries with GMO crops rise almost every year, and in 2020 Kenya and Nigeria joined this group (Cameron English 2020). While most of these countries grow cotton or corn crops, the U.S., Canada, China and Brazil cultivate an important variety of different GMO (corn, cotton, soybean, rice, tobacco, etc). Spain requires a special mention since, in 2017, its GMO corn crops (MON-810) make the 95% of the GMO-cultivated area of Europe (Elcacho 2017).

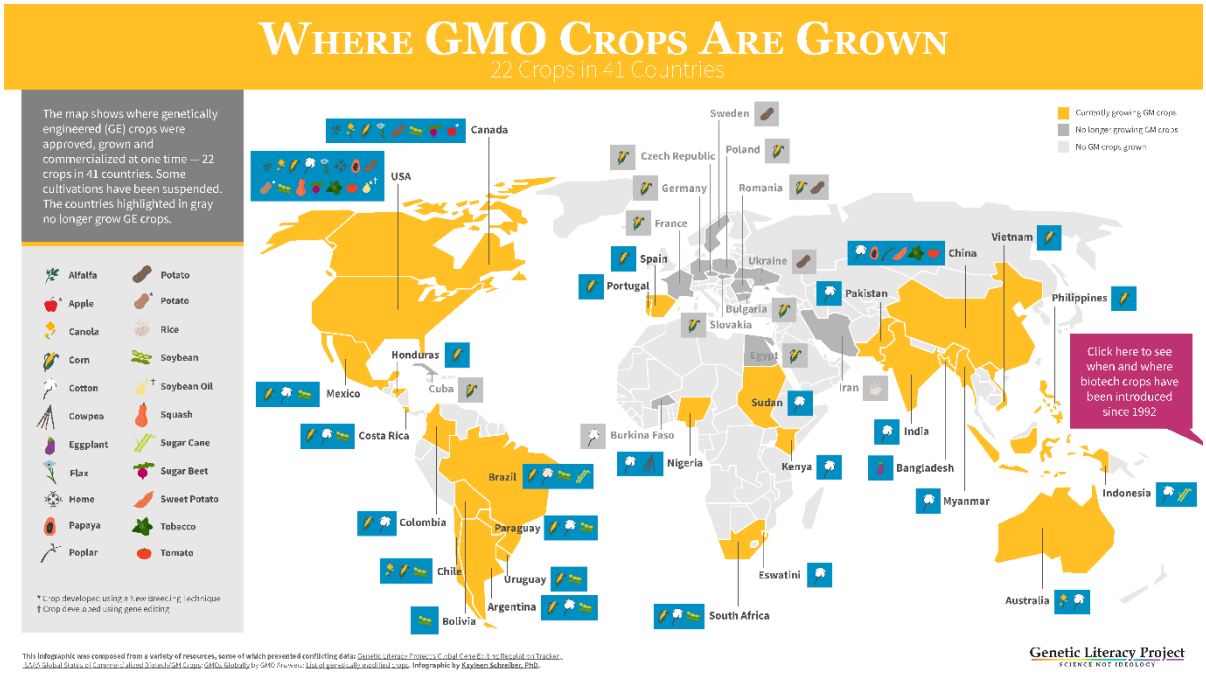

Figure 4. **Countries where GMO crops are (yellow) and were (dark grey) grown**. Source: GLP

GM Microorganisms in Food Industry

A little less controversial use of the GMO is the production of food enzymes which can come from wild-type microorganisms (WT) or genetically modified strains (GM). These enzymes

can also be obtained from plants and animals, but microbial sources are preferred for the following reasons (Deckers, Deforce et al. 2020):

- Many microorganisms naturally excrete these enzymes to extracellular space, simplifying the extraction step.
- Particular strains can be selected in order to obtain well-characterized enzymes with specific properties.
- A shorter production time results in lower costs. Moreover, plants and animals need to be transported to extraction facilities, which increases the costs even more.
- Higher yields can be obtained from microorganisms' enzymes.
- Microbial enzymes usually have a higher activity and stability. Enzymes obtained from thermophilic microorganisms will have a higher temperature tolerance.
- Microorganisms can be easily genetically modified to obtain specific enzymes, higher yields and better characteristics.

All these characteristics together make microorganisms a perfect tool to carry out food fermentations. GM microorganisms are advantageous because it allows the combination of appropriate production sources with the production of desired enzymes. For example, if a pathogenic microorganism produces a useful enzyme, its gene can be inserted in an innocuous strain, making the final product safe to eat or use. Using bioengineering, we can also create recombinant enzymes which are improved versions of their naturally-occurring homologues.

The transformation of a microorganism is a complex process with various steps. First, a host microorganism must be selected (e.g., *E. coli*) which suits the fermentation process to carry out. Then, the gene associated to the desired protein/enzyme must be isolated into a plasmid vector (see *Fig. 5*) which will be used to insert the gene into the host microorganism. Finally, the transformation itself (functional gene insertion) and several selection phases take place until a recombinant strain is isolated and cultivated (Olempska-Beer, Merker et al. 2006).

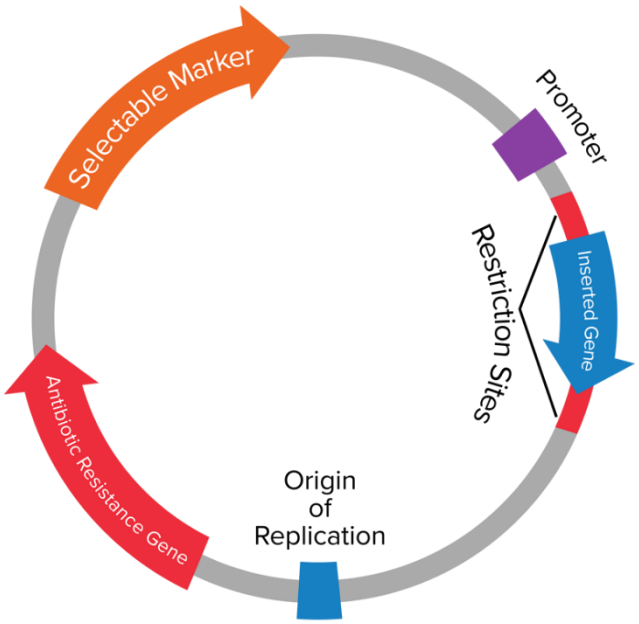

Figure 5. Example of plasmid vector. A plasmid is a molecule of circular DNA which is used to insert genes into organisms. It has a few key sequences: 1) Inserted gene(s) – It is the gene which protein is desired to be produced in the host organism. 2) Origin of replication – It is a sequence needed for the host organism to make copies of the plasmid. 3) Markers – Markers are genes used to know which cells/colonies have integrated the plasmid (e.g., Ampicillin resistance gene; only colonies containing the plasmid will survive in an ampicillin-containing medium). 4) Promoter – Region necessary for the gene expression. 5) Restriction sites – Sequences needed for inserting the query gene in the host genome. Source: Website *blog.addgene.org*; *Plasmids 101: What is a plasmid?* (Margo R. Monroe).

Microbial Consortia

Let's present a hypothetical situation. A chemical reaction takes a reactant R to generate a product P, through an intermediary compound I as follows:

$$A \rightarrow I \rightarrow B$$

Let's imagine that one microorganism is able to perform the reaction A → I, while a different one performs the reaction I → B. An approach could be inserting the gene of the first organism into the second one (or vice versa). Thus, a recombinant organism would carry out the whole process. This solution may appear correct but there are some factors that can impede the optimal production of the final compound. For example, the recombinant organism may lack a coenzyme needed for the integrated reaction, reducing or inhibiting the reaction flux. The concept behind this example is that metabolism is not a set of isolated

biochemical reactions, but a complex network, where a change in the expression of a gene can have an impact on the production of a metabolite apparently not related.

To face this possible situation, as well as avoid a substantial part of the genetic transformation process, there is another way which is the usage of microbial consortia.

Microbial consortia are communities of different species or strains of microorganisms which interact via mutualism (two species gain benefits) or commensalism (one species gain benefits, but the other one is not affected). The clearest example of a microbial consortium is the human gut microbiota where there is not only interaction between microorganisms but with the human host. Gut microbiota is considered a dynamic organ which performs some basic functions in the immunological, metabolic, structural and neurological landscapes of the human body (Adak and Khan 2019).

In the industrial field, microbial consortia are used for the conversion of a substrate into a product, carrying out the different steps of the pathway. These communities present some advantages compared to monocultures, like robustness to perturbation, division of labour and spatial organization (McCarty and Ledesma-Amaro 2019). Moreover, like monocultures, microbial consortia can undergo genetic engineering and synthetic biology, maximizing the possibilities. Some examples of biotechnological processes performed by microbial consortia are presented in *Table 1* (Sgobba and Wendisch 2020).

Biotechnological process	**Microbial consortia**
Production of isobutanol from the lignocellulosic feedstock	Cellulase-secreting fungus *Trichoderma reesei* and an isobutanol producing *E. coli*
Production of L-lysine from sucrose	Sucrose negative and L-lysine auxotrophic *E. coli* and sucrose positive *C. glutamicum* producing L-lysine
Production of ethanol, isobutanol and butanol from cellulose	*Clostridium* sp. modules displaying cellulosomes
Production of fumarate from microcrystalline cellulose and alkaline pre-treated corn stover	*Trichoderma reesei* secreting cellulolytic enzymes and *Rhizopus delemar* producing fumaric acid
Production of bisdemethoxycurcumin from glucose	*E. coli* producing p-coumaric acid and *E. coli* converting p-coumaric acid into bisdemethoxycurcumin
Production of anthocyanins from glucose, glycerol and xylose	Glucose, glycerol and xylose utilizing *E. coli* producing phenylpropanoic acids and *E. coli* converting phenylpropanoic acids and malonate into flavanones and *E.*

	coli converting flavanones into flavan-3-ols and *E. coli* converting flavan-3-ols into anthocyanins
Production of rosmarinic acid from xylose and glucose	Glucose-negative *E. coli* producing caffeic acid and glucose-negative *E. coli* producing salvianic acid A and xylose-negative *E. coli* converting caffeic acid and salvianic acid A into rosmarinic acid

Table 1. Some uses of microbial consortia in biotechnology. The simplicity, knowledge and wide development of *E. coli* transformation make this organism the most used.

A Systems Biology Approach

Whether the approach requires monoculture transformation or microbial consortia, an essential tool is the systems biology. As stated by scientific writer Christopher Wanjek (Wanjek 2011):

Systems biology is an approach in biomedical research to understanding the larger picture – be it at the level of the organism, tissue, or cell – by putting its pieces together. It's in stark contrast to decades of reductionist biology which involves taking the pieces apart.

Wanjek defines it at the level of biomedical research but the reality is that systems biology is applied in a variety of fields, including the different forms of biotechnology. This approach is a mixture of computational modelling and simulations, omics (genomics, transcriptomics, proteomics, metagenomics...), biochemistry and math. Special mention to computation which allows us to work with such massive sets of data.

In this work, systems biology will be applied through metabolic models and flux balance analysis, concepts which will be explained in depth in the following sections.

Metabolic Models

An organism can be represented as a network of all its chemical reactions which take place into (or between) particular cellular compartments. A genome-scale metabolic model (GEM) computationally describes a whole set of stoichiometry-based, mass-balanced metabolic reactions in an organism using gene-protein-reaction (GPR) associations that are formulated on the basis of genome annotation data and experimentally obtained information. They are used to predict metabolic fluxes for various systems-level metabolic studies. Naming some of their applications: strain development for chemical and materials production, drug targeting in pathogens, prediction of enzyme functions, pan-reactome analysis, modelling

interactions among multiple cells or organisms, and understanding human diseases (Gu, Kim et al. 2019). The first created GEM belonged to *Haemophilus influenzae* (Edwards and Palsson 1999), followed by model organisms like *Escherichia coli* and *Saccharomyces cerevisiae*. Nowadays, thousands of GEMs have been reconstructed including plants, animals and even humans.

Reconstruction of GEMs

Many tools have been developed for the reconstruction of GEMs. The **automated methods** are preferred due to their low time necessities and convenience. Two of the first automated methods were **AUTOGRAPH** (Notebaart, van Enckevort et al. 2006) and **GEM System** (Arakawa, Yamada et al. 2006). AUTOGRAPH uses published models as a template to map ortholog genes and their gene-protein reactions. GEM System first assign functions to the genes in the target genome by conducting homology and orthology against SWISS-PROT and TrEMBL databases. Then, it maps appropriate reactions to metabolic genes, based on EC number matches to the KEGG pathway databases. These early methods were an important advance but they had some issues. For example, the resulting models usually were unfunctional, not being able to simulate metabolic fluxes with objectives such as biomass production. In that moment, a substantial amount of manual curation was required for these models to properly work.

Currently used methods reconstruct a draft which is refined and evaluated later as it is shown in *Fig. 6* (Faria, Rocha et al. 2018). One of the most relevant tools of this "new wave" of GEMs builders is **ModelSEED** (Henry, DeJongh et al. 2010), the first platform to combine de generation of draft models with network refinement, curation, automated gap-filling and network evaluation with flux balance analysis and phenotype datasets. ModelSEED has its own standalone website (modelseed.org), but it is also implemented in **PATRIC** (Wattam, Abraham et al. 2014) as a model reconstruction tool.

Other relevant tools include Merlin, RAVEN, Pathway Tools and SuBliMinal Toolbox.

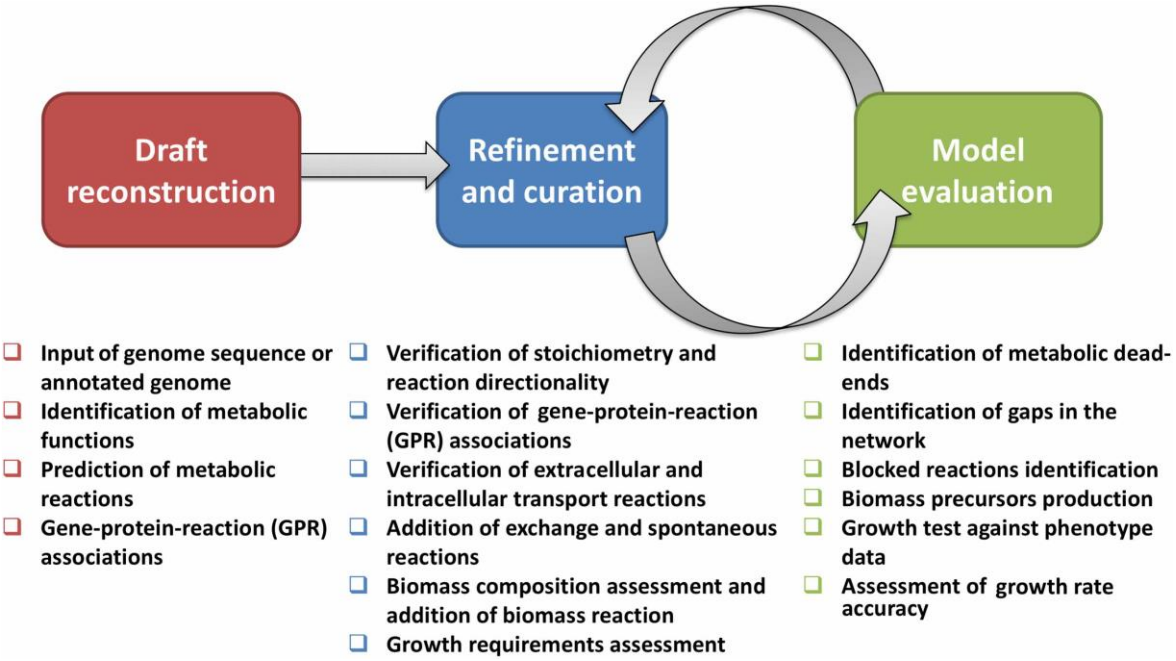

Figure 6. General workflow of GEM reconstruction. First, a draft model is generated from a genome. Then, the model is refined and evaluated via different methods to assure it is as good as possible. **Source:** Faria, Rocha *et al.* "Methods for automated genome-scale metabolic model reconstruction"

GEM files

The standard data output for the most GEM reconstruction/managing tools (including ModelSEED and COBRApy, which will be covered later) is called **SBML** (Systems Biology Markup Language) and consists on lists with one or more of the components below (Hucka, Finney et al. 2003):

- Compartment: A container of finite volume for well-stirred substances where reactions take place (e.g., cytosol).
- Species: A chemical substance or entity that takes part in a reaction.
- Reaction: A statement describing some transformation, transport or binding process that can change one or more species. Reactions have associated rate laws (for example upper and lower bounds) describing the manner in which they take place.
- Parameter: A quantity that has a symbolic name. SBML provides the ability to define parameters that are global to a model, as well as parameters that are local to a single reaction.

- Unit definition: A name for a unit used in the expression of quantities in a model. This is a facility for both setting default units and for allowing combinations of units to be given abbreviated names.
- Rule: A mathematical expression that is added to the model equations constructed from the set of reactions. Rules can be used to set parameter values, establish constraints between quantities, etc.

Although it is not a mandatory field, most of the GEMs contain the genes associated to reactions. SBML is more of a language than a file format, and different sources and tools may generate different files. For example, a SEED model file won't be exactly the same as a BiGG model file. Moreover, the file format can vary between XML, JSON and others, which are convertible.

Multi-Strain GEMs

Recently, a special type of GEM is acquiring relevance. Multi-strain GEMs are extended models containing homologous genes of different strains within the same species (Norsigian, Pusarla et al. 2020). These are generated upon a high-quality reference model, which genome is compared with other strains' and completed with their orthologous genes. The motivation for the generation of these models is the variability across strains of a species, that usually show diverse phenotypes, by growing in different media or carrying out different reactions.

These models have proven to be valuable for several applications (Norsigian, Fang et al. 2020). Metabolic capabilities predicted using multi-strain GEMs have been used to build classification schema capable of organizing strains into nutrient niche, serovars and pathogenicity. As scalability is one of the strengths of these models, some wider analysis can be performed. For example, allele frequencies of genes within a network context can be studied. This can be useful for the discovery of evolutionary hotspots or for tracking and understanding epidemics.

Databases

GEMs rely on different sources of data: genomes, chemical reactions, compounds, genes, etc. Most of these data are stored in a variety of databases which are essential tools for systems biologists. Some of the most relevant databases related to GEMs will be presented next.

One of the databases containing full GEMs is **BiGG Models** (http://bigg.ucsd.edu/) which means *Biochemical Genetic and Genomic knowledge base* (King, Lu et al. 2016). This repository contains more than 100 published genome-scale metabolic networks with a standardized nomenclature. Genes in BiGG models are mapped to NCBI genome annotations, and metabolites and reactions are linked to different sites (KEGG, PubChem...). This database is mainly filled with prokaryotic and single-cell eukaryotic GEMs, but it also contains models for human, mouse and hamster.

ModelSEED (https://modelseed.org/) is a project integrating a variety of resources for the reconstruction, optimization and analysis of GEMs (Henry, DeJongh et al. 2010). As stated before, it can be used as a builder employing user data, but ModelSEED takes on a lot more. It hosts its own reactions and compounds database, which is linked to general databases like KEGG, MetaCyc and BiGG as well as other more specific like BrachyCyc or ChlamyCyc. ModelSEED also relies on **RAST** (Aziz, Bartels et al. 2008) which is used for the annotation of the genomes. For model optimization, ModelSEED makes use of a set of tools, highlighting GapFill and GapGen. Furthermore, another plant-based project has born within ModelSEED which is **PlantSEED** (Seaver, Gerdes et al. 2014). Due to this, a variety of plant GEMs can be found and fetched on ModelSEED main website.

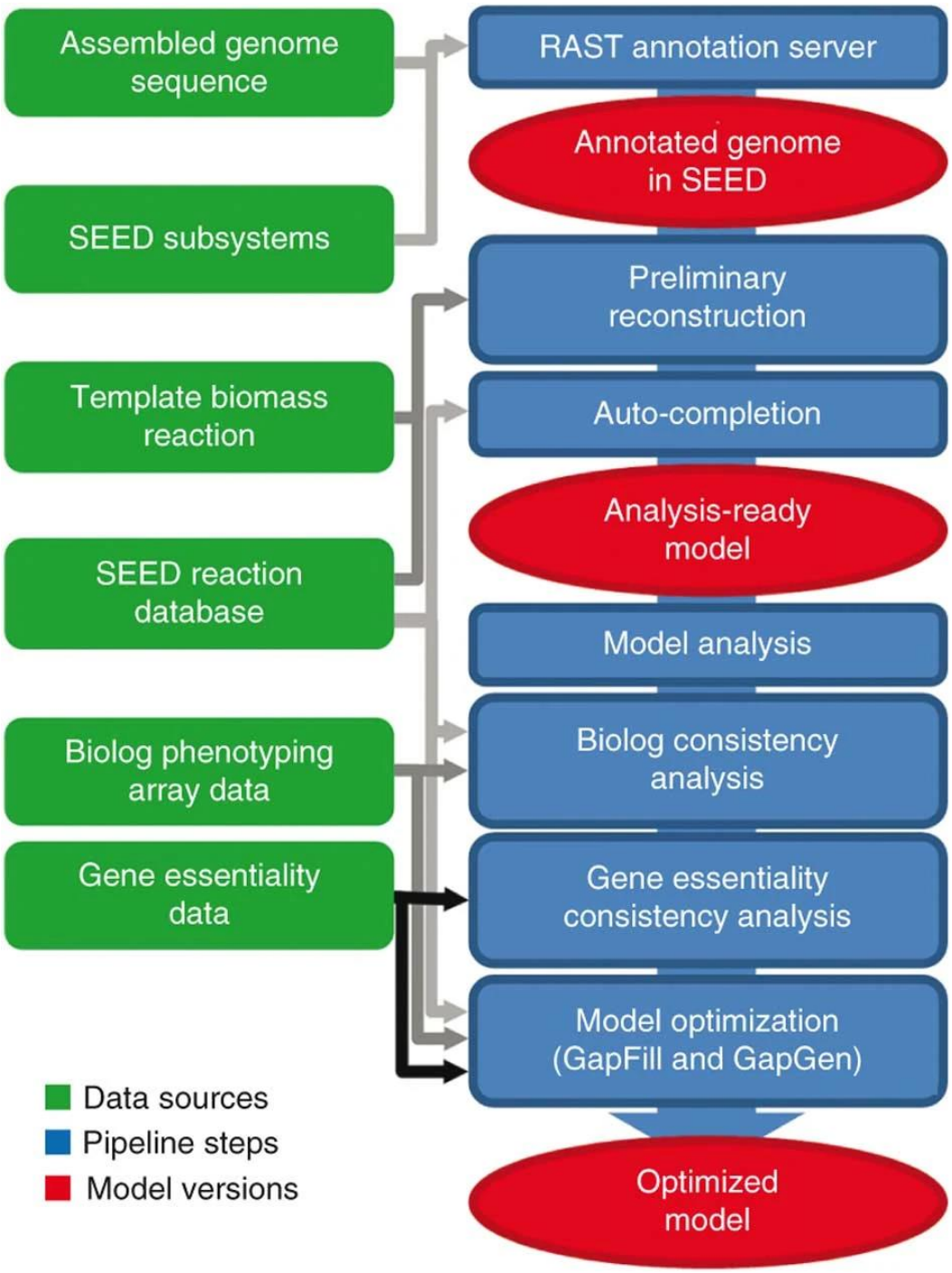

Figure 7. ModelSEED GEM reconstruction pipeline. Source: Henry, DeJongh *et al.* "High-throughput generation, optimization and analysis of genome-scale metabolic models."

KBase (https://www.kbase.us/) (Arkin, Cottingham et al. 2018) is an even more ambitious project held by the United States Department of Energy that integrates ModelSEED, BiGG and many other resources to create an environment for systems biology. It includes pipelines for genome assembly and annotation, RNA-seq and expression analysis, metabolic modelling, community modelling and comparative genomics.

PATRIC (https://www.patricbrc.org/), which stands for *Pathosystems Resource Integration Center* (Gillespie, Wattam et al. 2011), is another resource worth-mentioning. Like ModelSEED, PATRIC is more of a multi-tool platform than a database, but it also contains thousands of microorganism genomes that can be used for the reconstruction of GEMs.

To finish off, some of the reactions/metabolites databases must be cited, as they are useful by themselves. The most relevant databases for the reconstruction and analysis of GEMs are **KEGG** (https://www.genome.jp/kegg/), **MetaCyc** (https://metacyc.org/) and **MetaNetX** (https://www.metanetx.org/). The last one will be really helpful in the course of this work because it contains a set of tables for the conversion of different reaction/metabolite nomenclatures. These tables are essential in order to work with BiGG and SEED models within a single project.

Flux Balance Analysis

The main aim of GEMs is arguably the simulation of metabolic processes in order to develop metabolic engineering. Metabolic engineering consists on modifying the metabolic potential and genetics of an organism to optimize the production (or degradation) of a specific substance, or the organism itself.

Flux balance analysis (FBA) is a constraint-based optimization approach that can be used to simulate ranges of achievable reaction rates, usually referred as metabolic fluxes, in the metabolic network of an organism. These constraints can be mass balance, compartmentalization, or thermodynamic directionality, among other. The available stoichiometric information for a metabolic network is incorporated into a stoichiometric matrix *S*, in which rows represent metabolites and columns represent reactions (Simeonidis and Price 2015). The system is usually assumed to be in a quasi-steady state, where $Sv = 0$ (v is a vector containing reaction fluxes) and lower and upper bounds can be set in order to force a certain directionality or capacity to reactions.

The system usually has many possible solutions that satisfy the constraints. To select a solution an objective function must be optimized. This objective function describes the maximization of biomass production, production of a specific compound or consumption of another one. A general FBA formulation is represented as follows:

Maximize:

$$Z = wv$$

subject to:

$$Sv = 0$$

$$v_{\min} \leq v \leq v_{\max}$$

Where $\boldsymbol{S}$ is a matrix representing stoichiometric coefficients of all compounds in every reaction, $\boldsymbol{v}$ is a vector containing reaction fluxes, $\boldsymbol{v_{min}}$ and $\boldsymbol{v_{max}}$ are constraints imposed by the user, and vector $\boldsymbol{w}$ adds weights that represent relative contribution of each reaction to the objective function.

The solution of an FBA problem is unique for the optimal value of the objective function as well as a non-unique calculation of a flux distribution through every reaction in the system. Thus, production and consumption rates for every metabolite in the network can be determined. *Fig. 8* shows a typical FBA process (Gianchandani, Chavali et al. 2010).

The benefits of FBA are several. First, kinetic information and enzyme concentration are not needed for calculations (nevertheless they can be added as complementary information). Also, the low number of parameters also helps to reduce overfitting. Finally, FBA uses linear programming (LP), which allows the employment of very large metabolic networks.

Modelling Limitations

GEMs are derived from annotated genomes which usually lack some genes, and consequently some enzymes, which have not been identified. These “gaps” can easily result in an unviable model. Thankfully, this issue has been widely addressed with the development of several gap filling algorithms (they will be discussed afterwards). It also can be resolved by integrating additional omics information.

Another limitation is the inability to describe internal regulations (e.g., feedback). This is related to specific expression of isoforms that are transiently present in different compartments (Bazzani 2014).

In some cases, enzymatic isoforms are mainly regulated by single fatty acids (Bogdanov, Mileykovskaya et al. 2008). This wide set of molecules may overcome the descriptive limits of FBA. It can be solved using generalized mass-action kinetics.

To sum up, a GEM will always be an approach of an organism, since they ignore many of the cellular mechanisms aside of enzymatic reactions. This being said, they are still a valuable tool which main limitation is, precisely, inherited from genome annotation deficiencies.

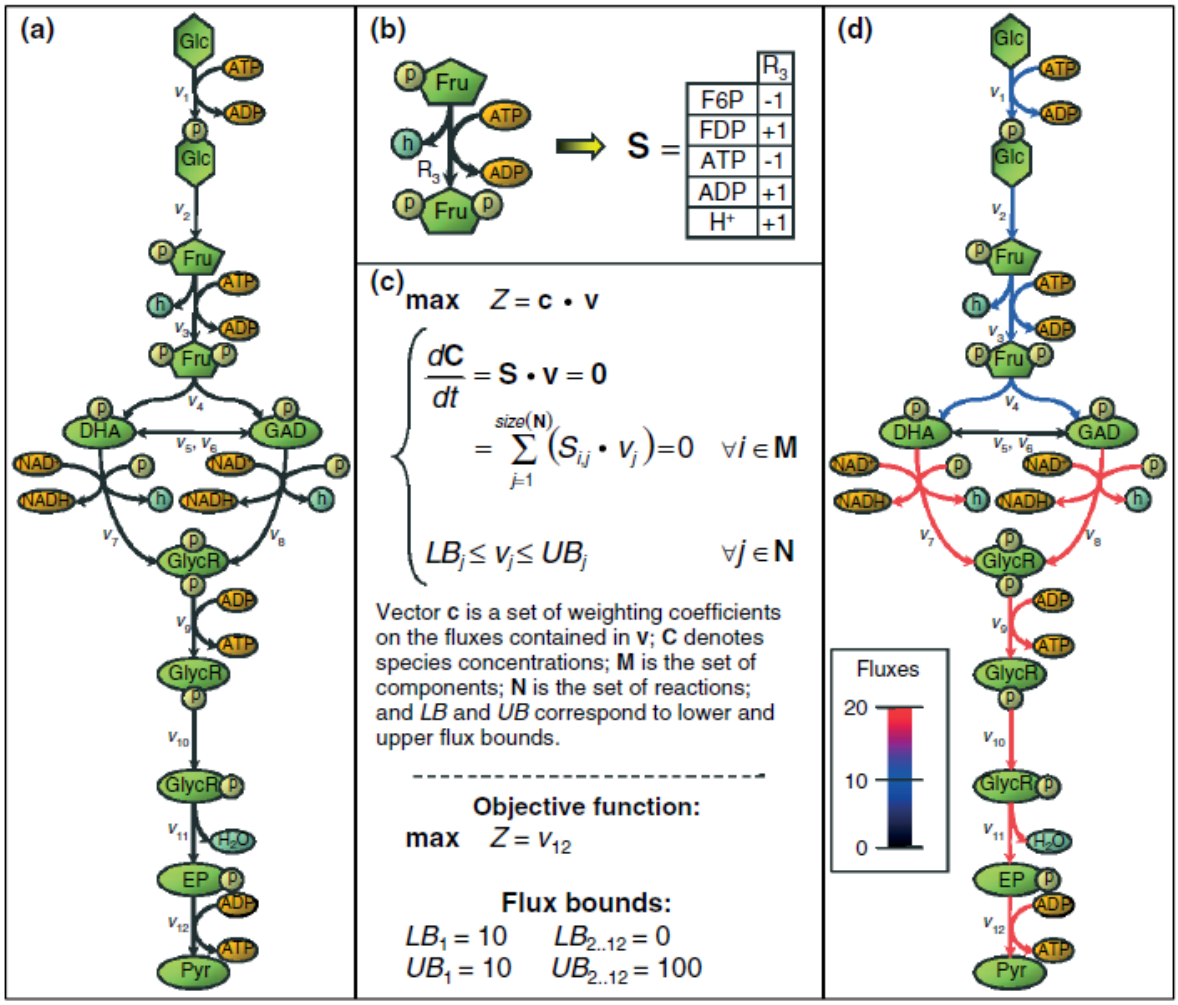

Figure 8. Flux balance analysis approach. A) Representation of a metabolic pathway, in this case glycolysis. Reactions are labelled from v_1 to v_{12}. B) Stoichiometric matrix (*S*) is generated (image shows just one reaction for illustrative purpose, but matrix *S* contains every reaction and metabolite). C) Mathematic FBA formulation. In this case, the objective function is the maximization of last reaction of the pathway (v_{12}) flux. Lower and upper bounds are set as well. D) Representation of the solution of FBA. **Source:** Gianchandani, Chavali *et al.* "The application of flux balance analysis in systems biology."

Gap Filling

As introduced before, our limited knowledge of the annotated genomes impacts the quality of the models. There are two types of missing information in metabolic networks reconstruction. On the one hand, there can be gaps in a network, to wit, places where a

reaction that consumes or produces a metabolite is missing, causing a dead-end. On the other hand, there can be “orphan reactions”. These are reactions that are known to exist, but the corresponding gene or genes have not been identified. Within gaps, metabolites which are produced and not consumed, or consumed but not produced are called, respectively, “root non-consumption metabolites” and “root non-production metabolites”. When performing FBA, reactions involving these metabolites can never carry flux. These reactions are said to be “blocked”, and any reactions upstream or downstream from them will also be unable to carry flux under steady-state conditions (Orth and Palsson 2010).

To solve this lack of information, several gap filling methods have been developed over the recent years. These methods use external databases for adding new reactions that complete the model. Many GEM builders have integrated gap filling algorithms to their pipelines. However, automated gap filling isn’t perfect at all and manual curation is highly recommended (Karp, Weaver et al. 2018).

In order to predict metabolic gene functions, this is, addressing orphan reactions, several methods have been developed, being **SEED**, **ADOMETA** and **PathoLogic Pathway Hole Filler** three of the most relevant.

In the case of gap filling methods, we will deepen next, since this work is mainly based on implementing a gap filling algorithm into a modelling pipeline.

Gap Filling Algorithms

GapFind and **GapFill** (Satish Kumar, Dasika et al. 2007) are two mixed-integer linear algorithms (MILP) developed in parallel. **GapFind** has the ability to identify every gap in a GEM, identifying all blocked metabolites. Although blocked metabolites identification seems trivial, some cases like cycles can be tricky, so GapFind is a very useful tool. **GapFill**, for its part, attempts to minimize the number of gaps by reversing the directionality of existing reactions, adding new reactions from MetaCyc database, and adding transport reactions between compartments, using the lowest possible number of modifications.

In many cases, GEMs are used to maximize the biomass production (i.e., growth). **GrowMatch** (Kumar and Maranas 2009) is an algorithm which uses experimentally determined gene essentiality to correct wrong model predictions. If a model predicts growth but experiments show that organism cannot grow due to the knockout of an essential gene, the is a growth-no growth inconsistency (GNG). If the opposite takes place, namely, the

model predicts no growth but experiments show growth, there is a no growth-growth (NGG) inconsistency. GrowMatch tries to solve these two types of errors by different ways. In the case of GNG, the model has some non-realistic features which need to be removed or constrained. In the case of NGG, the model is missing some fluxes that allow growth. GrowMatch makes use of GapFill and its three strategies mentioned before. Corrections in GNG and NGG models can be either global (resolve one inconsistency without generating new ones) or conditional (resolve one inconsistency while creating new ones in other strains/models).

Gap Filling Algorithm	Required Data
GapFill/GapFind	Database of potential reactions
GrowMatch	Gene essentiality data, database of potential reactions
SMILEY	Growth phenotype data, database of potential reactions
OMNI	Metabolic flux data, database of potential reactions
BNICE	Set of generalized enzyme reactions

Table 2. Classic gap filling algorithms. Besides the specified required data shown in table, every method obviously needs a metabolic network. **Source:** Orth and Palsson. "Systematizing the Generation of Missing Metabolic Knowledge".

An important aspect of metabolic simulations is the substrate where these organisms are supposed to be cultivated. When a model is predicted not to grow on some substrate, but experiments show the opposite, the model probably lacks one or more reactions involved in the consumption of this substrate. **SMILEY** (Reed, Patel et al. 2006) is another MILP algorithm which tries to identify a flux distribution that allows growth in the substrate of interest while minimizing the number of added reactions (from a universal database derived from KEGG). Thus, it follows the principle of maximum parsimony (the simplest explanation is the most likely to be true). This method does not directly perform gap filling in the model, but its results can end up doing it.

As with the previous methods, **OMNI** (Reed, Patel et al. 2006) uses MILP to compare predictions with experimental data to improve a constraint-based model. This algorithm compares predicted fluxes with experimentally measured fluxes in order to obtain a model which minimizes the differences between both.

The last of the discussed classic methods is **BNICE** (Hatzimanikatis, Li et al. 2005), an approach very different from the previous ones. This algorithm identifies all possible reactions that can occur between two metabolites, based on their thermodynamic properties. For that purpose, metabolites are represented as bond-electron matrices (BEMs) and reactions are represented by matrix addition. It is possible to model a reaction with some substrates by adding the same reaction (involving other substrates) to different BEMs. By doing this, *de novo* reactions, which can correct the model, are obtained.

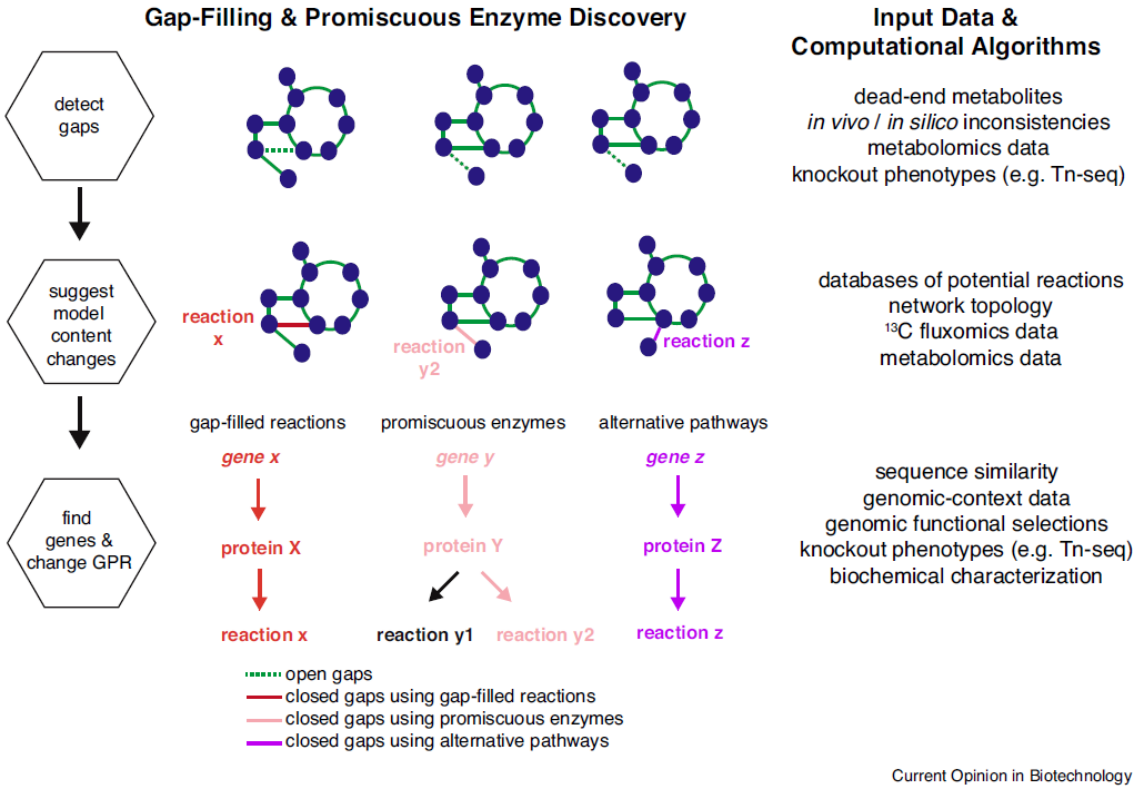

Figure 9. Typical steps of classic gap filling algorithms. These methods usually follow this scheme. 1) Algorithms identify dead-end metabolites or *in silico-in vivo* inconsistencies. 2) They add potential reactions from a database which can solve these gaps. 3) Some gap filling algorithms discover genes responsible for these added reactions. **Source:** Pan and Reed. "Advances in gap-filling genome-scale metabolic models and model-driven experiments lead to novel metabolic discoveries"

Although classic gap filling algorithms are still the most used, over the recent years, researchers have developed more sophisticated methods. **FASTGAPFILL** (Thiele, Vlassis et al. 2014) is an efficient method that computes a nearly minimal set of added reactions for a compartmentalized model. **GLOBALFIT** (Hartleb, Jarre et al. 2016) solves the computational limitations derived from MILP by reformulating gap filling into a bi-level linear problem. **GAUGE** (Hosseini and Marashi 2017) was the first algorithm to exploit flux

coupling analysis (FCA), that detects if two reactions are dependant of each other. Using FCA, GAUGE finds gaps involving genes associated with fully dependant reactions but show uncorrelated reaction patterns. The drawback of this method is its need for a well-defined GPR associations network. There also are algorithms based on hidden Markov models or machine learning techniques, but they are still under development.

Although newer methods have proven to be promising, there are still some limitations that should be solved. First, most of these methods cannot resolve false positive predictions due to the complexity of the cell. Moreover, models tend to be overfitted when tried to resolve dead-end metabolites or resolve false positive predictions. Finally, solutions can be non-unique and hard to rank (Pan and Reed 2018).

COBRApy

COBRApy (Ebrahim, Lerman et al. 2013) is a software with such a central importance in this work that deserves its own section. As stated in its website (https://opencobra.github.io/cobrapy/), it is Python package, derived from COBRA Toolbox (originally developed for MATLAB), that provides a simple interface to metabolic constraint-based reconstruction and analysis. This package includes modules for GEM reconstruction or loading from different file formats (JSON, SBML or MATLAB), FBA, flux variability analysis (FVA), and gene deletion analysis. It also integrates a gap filling algorithm which will be explained later.

The core classes of COBRApy are Model (organism), Metabolite, Reaction and Gene. The Model class acts as a container for a set of Reactions associated with Metabolites and Gene products. A diagram of COBRApy classes is shown in *Fig. 10.*

COBRApy make use of different optimization solvers, which are software that calculate the solution of mathematical problems. **GLPK** (GNU Linear Programming Kit) (https://www.gnu.org/software/glpk/) is a free software that is used by default. **CPLEX** (https://www.ibm.com/es-es/products/ilog-cplex-optimization-studio) and **Gurobi** (https://www.gurobi.com/) may also be used but both need a license (Gurobi can be used under a renewable academic license). All of these solvers can solve linear programming (FBA), MILP (gap filling) and quadratic programming problems.

Moving on to the integrated gap filling algorithm, it is based on SMILEY (mainly) and GrowMatch, both introduced before. MILP is used to obtain the lowest number of reactions

needed to be added for a user-defined collection of reactions. The problem to solve is as follows:

Minimize:

$$\sum_i c_i * z_i$$

subject to:

$$Sv = 0$$
$$v^* \geq t$$
$$l_i \leq v_i \leq u_i$$
$$v_i = 0 \; if \; z_i = 0$$

Where *l*, *u* are lower and upper bounds for reaction *i* and *z* is an indicator variable that is zero if the reaction is not used and otherwise 1, *c* is a user-defined cost associated with using the i^{th} reaction, v^* is the flux of the objective and *t* a lower bound for that objective.

The objective function can either be set as growth/biomass production (default) or a certain reaction to optimize. Different reaction sets can be obtained by having the algorithm go through multiple iterations.

COBRApy also includes a function to add a growth media to a model as nutrients availability is known to have a major impact on metabolic fluxes.

COBRApy has many benefits which make it a very good choice to work with GEMs. First, it employs an object-oriented programming approach that is more suited to representing increasingly complex models of biological networks. Furthermore, being a Python software, it is relatively easy to integrate models with databases. Finally, COBRA Toolbox (and consequently COBRApy) is currently a standard framework for reconstruction and analysis of GEMs which carries an active community and the compatibility with other relevant sites like ModelSEED or BiGG.

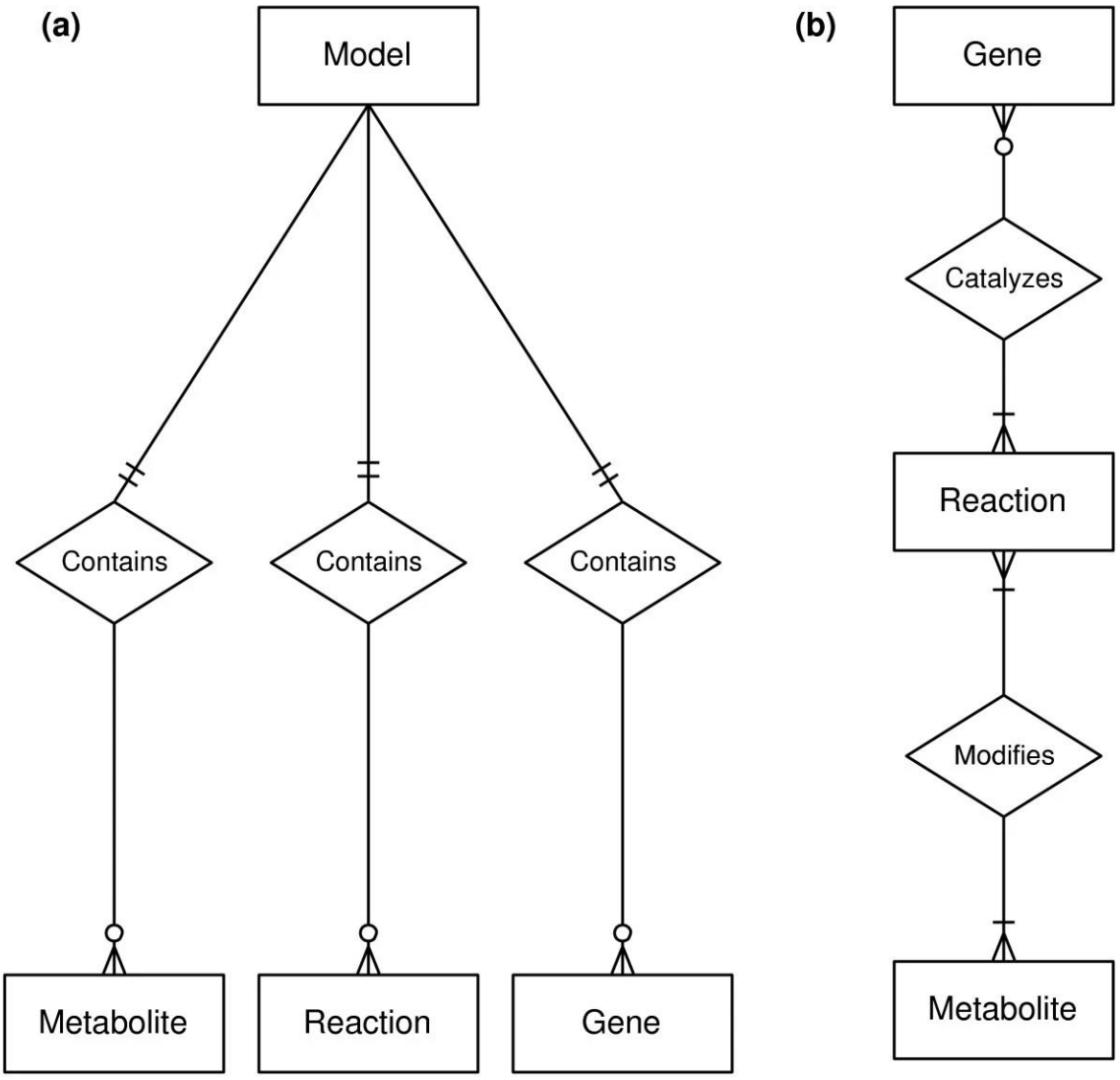

Figure 10. Entity relationships diagrams for core classes in COBRApy. A) A Model contains Metabolites, Reactions and Genes. B) A Reaction may be catalysed by zero (e.g., diffusion reaction), one or more Genes. A Reaction modifies one or more Metabolites. **Source**: Ebrahim, Lerman *et al.* "COBRApy: COnstraints-Based Reconstruction and Analysis for Python"

Gallant

Gallant is a software for the automated reconstruction of multi-strain GEMs, developed by **Systems Biotechnology Group** at Centro Nacional de Biotecnología (**CNB**). Gallant is a *snakemake* pipeline which, given a set of NCBI IDs (reference + homologous genomes), retrieves the genomes, performs several BLASTs to obtain orthologous genes, reconstructs their GEMs, finally getting a multi-strain GEM. Gallant has some features which make it a powerful tool in this matter. For example, its modular design allows the integration of new independent functionalities. Moreover, this pipeline makes use of parallel processing, treating each sample at once, which drastically reduces time consumption.

Objectives

One of the main deficiencies of some GEM builders, including Gallant, is the lack of a gap filling module, which can resolve incomplete models. Gallant is capable of generating a multi-strain GEM, a kind of model which has not been fully explored. In fact, no food industry microorganism multi-strain GEM has been published yet, despite its possible applications. This project tries to solve the mentioned limitations, having the following objectives:

1. **Creation of a gap filling module (GGF) for its integration in Gallant workflow**. GGF will be composed of two submodules, **Homology Gap Filling**, which uses homology models as databases, and **ModelSEED Access,** which provides an alternative source of models.
2. **Gap filling application over a multi-strain GEM of *Lactococcus lactis***. This bacteria species is a standard at food industry, being used in the manufacture of dairy products, pickles and other foods.

Methods

In this section decision-making, partial objectives and methodology during the development of the **gap filling module GGF** will be presented and discussed. As told before, this module will be integrated within **Gallant** tool, which workflow will be shown later. Two submodules have been designed. The first one, called **Homology gap filling**, is made upon **COBRApy gap filling algorithm**, with some improvements. It basically takes a query model, and a set of sorted by identity template models, which are used sequentially to perform gap filling on the query model until no reactions are added. The second submodule is called **ModelSEED access**, and its purpose is to have an additional source of models. It takes a set of **PATRIC** IDs, reconstructs their corresponding GEMs and translates them into BiGG nomenclature if desired. These models can be used either as templates or query.

The **GitHub repository of this project** can be accessed through the link below:

https://github.com/fcomnozz/GALLANT_gapfilling

Computer Features

All software development and testing were made in a Linux-based personal computer with the following features:

Operative System: Ubuntu 18.04.5 LTS Bionic

Processor: Intel Core i7 8750H, 2.20GHz, 6 cores, 12 threads

Memory: 32GB DDR4

Hard Drive Disk: SATA ST1000LM049 1TB (400GB partition)

Database Information

BiGG Models (http://bigg.ucsd.edu/)

BiGG Models was used as a reference database for GEMs. Homology gap filling submodule has been developed specifically for GEMs with BiGG nomenclature. All the homology gap filling testing files have been downloaded from BiGG Models as well as the three organisms used as query in the experimental part, which will be seen later. These three models are the following:

- *Lactococcus lactis* subsp. cremoris MG1363 (http://bigg.ucsd.edu/models/iNF517)
- *Pseudomonas putida* KT2440 (http://bigg.ucsd.edu/models/iJN1463)
- *Escherichia coli* str. K-12 substr. MG1655 (http://bigg.ucsd.edu/models/iML1515)

PATRIC (http://patricbrc.org/)

PATRIC is a repository containing, among other resources and functionalities, thousands of genomes from which GEMs can be reconstructed. Genomes' IDs are passed to ModelSEED-access submodule to obtain their correspondent GEMs.

ModelSEED (https://modelseed.org/genomes/ | https://github.com/ModelSEED)
ModelSEED is a reference database and platform which is able to reconstruct GEMs from PATRIC genomes. A whole submodule has been developed in order to programmatically access to both resources, reconstructing new models and gap-filling them with ModelSEED algorithms.

MetaNetX (https://www.metanetx.org/mnxdoc/mnxref.html)

The last database implied in this project is MetaNetX, particularly, two tabular files named *chem_xref.tsv* and *reac_xref.tsv*. These files were useful to generate translation tables between BiGG and SEED nomenclature. The management and modification of these files can be seen on the project's GitHub repository (seed2bigg directory).

Software Information

GitHub (https://github.com)

Used as a platform for script developing and repository.

Python (https://www.python.org/) – version 3.8.5

Used as a programming language for the development and testing of all the functions.

Jupyter notebook (https://jupyter.org/) – version 6.1.4

Used as Python interface, testing and results presentation.

Snakemake (https://snakemake.readthedocs.io/en/stable) – version 6.4.1

Used for the assembling of Gallant.

Python packages

COBRApy (https://github.com/opencobra/cobrapy) – version 0.19.0

Python package for COBRA Toolbox. Used for GEM loading, management, calculations (FBA) and base gap filling.

Gurobipy (https://gurobi.com/) – version 9.1

Python version of Gurobi solver. Used under academic license for FBA and gap filling calculations.

Mackinac (https://github.com/mmundy42/mackinac) – version 9.1

Used for ModelSEED access.

Pandas (https://github.com/pandas-dev/pandas)

Pandas have been used for dataframe management in the obtention of SEED-BiGG translation tables, and for presentation purposes on testing and result notebooks.

Workflow

Gallant

In this section, a general scheme showing the designed workflow will be presented. Gallant is a snakemake workflow which takes as initial input a table of NCBI Nucleotide IDs, containing a reference genome (first row) and multiple homologous genomes. First, all the genomes are retrieved from NCBI, and their genes and proteins are extracted. Then, several BLASTs are performed between the reference genome and the rest in order to get a matrix of orthologous genes. Next, strain-specific GEMs are reconstructed with the genes coming from orthologous matrix. Two more steps are carried out to finish the pipeline. First, gap filling, which is the central part of this project, is performed, using homologous GEMs as potential reactions databases. The last part consists on checking the quality and consistency of the final model, using diverse strategies, like FASTCC (Vlassis, Pacheco et al. 2014) and MEMOTE (https://memote.readthedocs.io/en/latest/) .

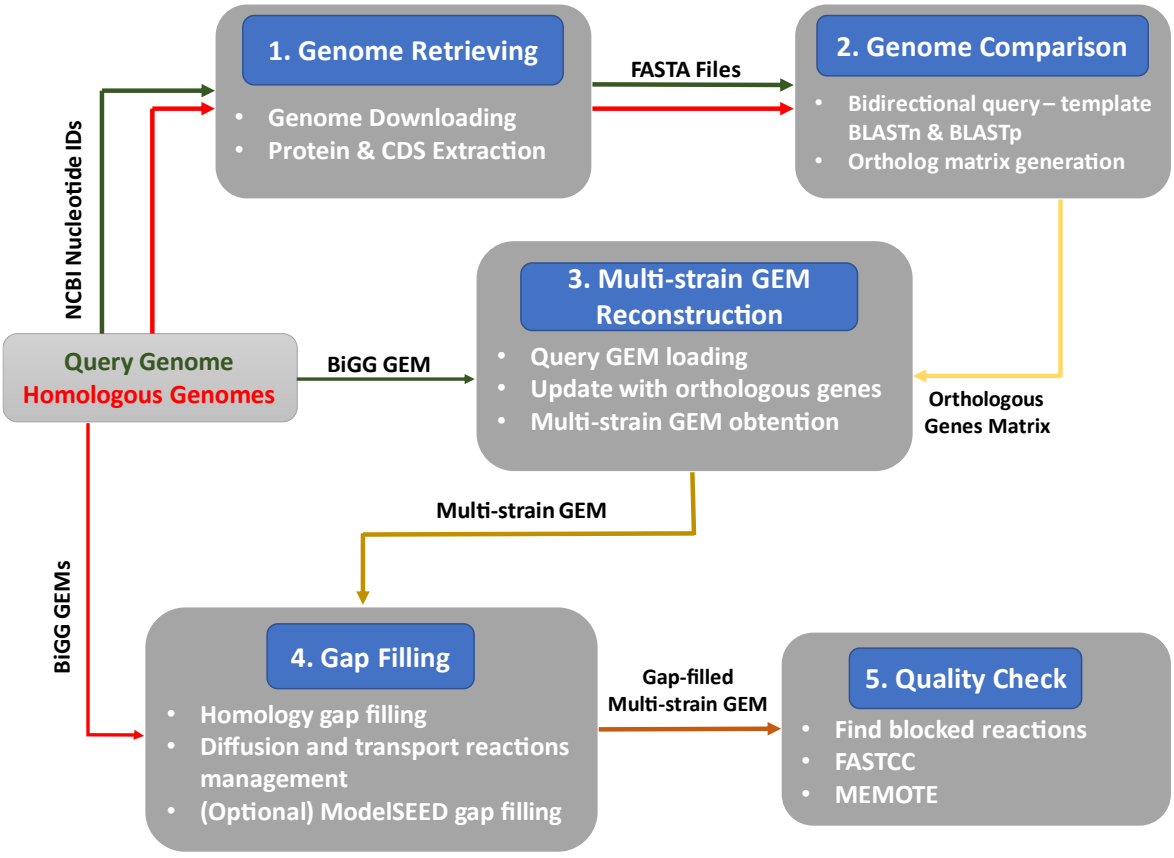

Figure 11. Simplified Gallant workflow. **1)** NCBI Nucleotide IDs for reference (henceforth query) and sorted by identity homologous genomes (henceforth samples) are passed as an input. Genomes are downloaded and proteins and CDS (coding sequence) are extracted for each ID. **2)** Bidirectional BLASTp and BLASTn are performed over every query-sample pair, obtaining a matrix representing samples and orthologous genes. **3)** Query GEM is loaded from a BiGG file (previously downloaded)

with COBRApy. Orthologous genes with more than 80% of similarity are added to query model in order to get a multi-strain GEM. Furthermore, genes which are not considered homologous are removed from the model. **4)** Homology gap filling is performed over multi-strain GEM, using sample associated BiGG GEMs as templates. Transport and diffusion reactions can be added if their metabolites are present in the query GEM. Optionally, ModelSEED models can be retrieved and used as templates if an alternative source is needed. **5)** The gap-filled multi-strain GEM is subjected to a quality check, in which blocked reactions are searched, and a consistency analysis (COBRA FASTCC) and MEMOTE analysis are performed.

Gap Filling

Two submodules of gap filling have been implemented. The first one is called homology gap filling submodule. It takes a query GEM as input (being the multi-strain model in Gallant) and a set of template-BiGG GEMs. These templates act as reactions database when gap-filling the query model. For that purpose, the SMILEY-based COBRApy gap filling algorithm has been implemented and extended, by adding transport reactions and diffusion reactions options. Additionally, in the ModelSEED access submodule, PATRIC IDs can be passed in order to reconstruct their models and gap fill them via ModelSEED and Mackinac (Mundy, Mendes-Soares et al. 2017). These new models can be either used as templates in the homology gap filling function or for addition of exchange/transport reactions. A general workflow of GGF is showed in *Fig. 12*.

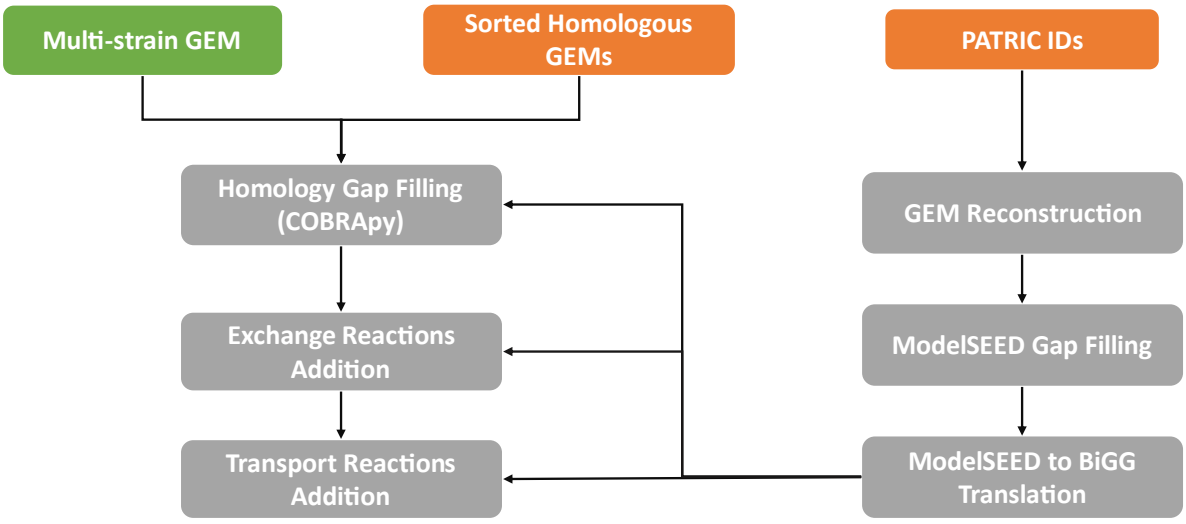

Figure 12. GGF workflow. Vertical workflow on the left represents the homology gap filling function. The multi-strain GEM, and a list of homologous GEMs sorted by identity make up the input for this function. COBRApy gap filling is performed sequentially using first the most similar GEM as template, going down until no reactions are added. Then, exchange and transport reactions from the templates can be added if they meet the metabolites requirements in the query GEM. On the right, ModelSEED access submodule is showed. It takes a list of PATRIC IDs as input, which belong to genomes, reconstructs their GEMs, gap fills them using a universal template hosted by ModelSEED and finally translates the GEMs to BiGG nomenclature. These models can be incorporated at the start

of the homology gap filling process, being fully used as templates, or just for exchange and/or transport reactions addition.

Homology Gap Filling

The homology gap filling function has been designed following certain criteria. The COBRApy algorithm was selected because its wide use, and its compatibility with the rest of the pipeline. It is also a relatively simple algorithm which is subject to possible changes in the core. The main assumption of this function is the following: a more (genetically) similar model will be more likely to contain appropriate GPR to add to the query. Consequently, gap filling is performed through templates in decreasing order of similarity, while reactions are added. When a template doesn't contribute with any reaction (i.e., the algorithm does not consider its reactions to be needed in the query model), gap filling process stops, outputting the gap-filled multi-strain GEM and a list with the used templates and reactions added. This assumption, though, may not be real at all, as we know that in biology just a very few things are exact. Thus, an option allowing to use all templates has been added. The basic workflow of the function is the following:

1) Query and a list of template GEMs are passed as input. Objective function to optimize is selected.
2) Initial flux of the model is calculated using FBA by COBRApy.
3) Function iterates over each template (steps 4-7 are carried out for each template).
4) If selected, exchange reactions are added to the query. These reactions must appear in the template, and their corresponding metabolites must be in the query model. The exchange reactions addition function is integrated within the main function, but can be used independently.
5) If selected, transport reactions are added to the query. First, the function must determine whether a reaction is a transport one (user can choose between two criteria). Then, similarly to the previous step, if the suited metabolites are present in the query, corresponding transport reaction is added. This function can be used independently as well.
6) COBRApy gap filling is executed using template as database. Its basis was explained earlier and in summary it tries to minimize the number of reactions needed to optimize the flux. Selected reactions are added to the model.

7) After gap filling, FBA is performed again. Depending on the result, the workflow may continue in four ways (if use_all_templates = False):
    a. If initial flux value = 0 and new flux value = 0, the function iterates over the next template in order to get a functional model.
    b. If initial value < new value, the new value is set as initial value for the next loop.
    c. If initial value > 0 and initial value = new value, the function stops as it is supposed to be no improvement in the model.
    d. If initial value > new value, there are two possibilities. If the new value falls below a relative threshold set by user, the added reactions are removed from the model and the function stops. If the new value is higher than that threshold, lower and upper bounds of the reactions are set to 0, flux is calculated again, initial value is updated with the new value, and the function continues to the next loop. Setting bounds to 0 is useful to maintain the reaction in the model as it can be relevant, without altering the general flux.
8) If all templates are selected to be used regardless the results, the decisions above are not applied.
9) Gap-filled model and a python dictionary with all the used templates and added reactions and genes constitute the output.

The solver used for both FBA and gap filling is Gurobi through Python package Gurobipy. It was selected over GLPK and Scipy because it is compatible with integrality parameter, which can be useful if models are failed to validate.

Function Parameters

Next, the parameters of the homology_gapfilling() function are shown:

model : cobra.Model

The GEM onto which gap filling will be performed.

templates : [cobra.Model1, cobra.Model2, ...]

A list of models which will be used as templates.

model_obj : None, “biomass”, “{specific_reaction}”

The objective function to optimize in the FBA. It can be either None (default), which assumes there is an already set objective, “biomass”, that optimize biomass production, and a specific reaction present in query model which is desired to be optimized.

use_all_templates : False, True

False by default. Must be set to True if wanted to use all templates.

integer_threshold : float

Integrality threshold. This parameter establishes the threshold at which value is considered non-zero. When a model is failed to validated, lowering this value can solve it. It is 1e-6 by default and can be set down to 1e-9.

force_exchange : False, True

False by default. Must be set to True if exchange reactions are wanted to be added mechanically.

force_transport : False, True

False by default. Must be set to True if transport reactions are wanted to be added mechanically.

t_all_compounds : False, True

This parameter specifies how a reaction is considered to be a transport reaction. If set to False (default), at least one of the compounds of the reaction must appear on both sides (reactants and products) of it. If set to True, all the reactants must appear as products.

t_ignore_h : False, True

Another parameter for transport reactions. If set to True, protons (H^+) are not taken into account when deciding if a reaction is a transport reaction.

value_fraction : float

By default, 0.8. It is the fractional threshold at which an added by gap filling reaction, which lowers the flux value, is removed from the model. For example, for a value_fraction = 0.8, and an initial flux of 1, if flux descends to 0.5, reaction causing lowering is removed. If flux descends to 0.9, reaction is kept, but its bounds are set to 0.

ModelSEED Access and Gap Filling

ModelSEED access submodule has been built upon a package called Mackinac. Mackinac is described as a bridge between ModelSEED and COBRApy. It was used for reconstructing GEMs from PATRIC genomes and gap-fill them. This submodule was designed in order to get an alternative source of templates, for example, if a desired GEM is not uploaded to BiGG Models. The workflow is as follows:

1) A list of PATRIC IDs is passed as input. A PATRIC account is needed to access to this database, and a password will be required when executing the function.
2) ModelSEED models are reconstructed using Mackinac function reconstruct_modelseed_model(). This is the most time-consuming part of the process.
3) ModelSEED models are converted into COBRA-compatible models with the function create_cobra_model_from_modelseed_model().
4) If desired, these models are gap-filled using ModelSEED algorithm and its universal reaction database. ModelSEED gap filling function is based on penalties similar to the one in COBRApy, where number of added reactions is minimized. The employed function is gapfill_modelseed_model().
5) Finally, the model content (reactions and metabolites) is translated to BiGG nomenclature in order to be used as templates. For this purpose, two SEED-BiGG translation tables have been generated from files in MetaNetX.
6) The output is a dictionary with model names as keys and models themselves as values. These models are then prepared for template usage.

Function Parameters

id_list : [“ID1”, “ID2”, ...]

List of PATRIC IDs which GEMs are desired to be reconstructed.

gapfilling : True, False

True by default. Performs ModelSEED gap filling over the models.

tobigg : True, False

True by default. Indicates if BiGG translation will be carried out.

M_table : pandas.DataFrame

Translation two-column dataframe for metabolites. Headers must be “BiGG” and “SEED”. A translation table, name “compounds.tsv” has been generated for this purpose.

R_table : pandas.DataFrame

Translation two-column dataframe for reactions. Headers must be “BiGG” and “SEED”. A translation table, name “reactions.tsv” has been generated for this purpose.

Transport and Exchange Reactions

Exchange (diffusion) and, specially, transport reactions have proved to be a bit tricky on gap filling. Many of these where not added when performing gap filling during function testing, even knowing they would improve the model. Thus, it was decided to treat these reactions differently.

In the case of exchange reactions, a function was designed to insert those template reactions which metabolites are present in the query model. It is a simple search by name, since all exchange reactions in BiGG and SEED nomenclature start with “EX_”.

In the case of transport reactions, things are way more complex. A transport reaction can be described as a process in which one or more compounds get through a biologic membrane (cytoplasmic membrane, nuclear envelope, mitochondrion membrane, etc) with the aid of a transport protein, namely, involving one or more genes. In GEMs, unlike exchange reactions, the transport reaction names are too diverse to extract a regex rule to determine if they are, actually, transport reactions. Thus, we must look at the reaction itself to establish its type. A simple definition for a transport reaction can be the following:

“A reaction which reactants and products are the same, but changing cell compartment.”

This definition includes a vast part of transport reactions, even some complex reactions, in which two compounds switch sides (e.g., Sodium-potassium pump). However, there are still transport reactions that don’t meet the previous statement. One example is a transporter which requires ATP consumption, in which occurs the sub-reaction ATP → ADP + Pi, which is a chemical transformation.

Other possible definition, which include the previous example is the one below:

“A reaction in which at least one of the reactants appears in the products, but changing cell compartment.”

This definition appears to solve the previous issues, but brings a new one. Some non-transport reactions like ATP synthesis, catalysed by ATP synthase, uses proton (H^+) pumping to obtain the energy needed for the ATP assembly to occur. In this case, there is a molecule that switches compartment, resulting, by definition above, in a false positive. A parameter meant to ignore protons have been implemented in the function to solve proton-pumping cases. Nonetheless, there are probably other scenarios where transport reactions may be ignored and non-transport reactions may be mis-tagged. Thus, both approaches have remained in the module, giving the user the chance to elect the most suitable one.

Results

Validation of the Method

The core of GGF is based on COBRA's gap filling function, which in turn is based on SMILEY. In order to determine if GGF surpasses COBRA, a simple comparison test have been designed and executed. This test is meant to measure the number and type of reactions added to models and its execution time. It basically consists on the elimination of 1, 2, 5 or 10 random reactions from three different models (*E. coli, L. lactis, P. putida*), under two different carbon sources (glucose and ribose), with the later gap filling. The test design is as follows:

1. Three different BiGG models were selected: ***E. coli*** str. K-12 substr. MG1655 (iML1515), ***L. lactis*** subsp. cremoris MG1363 (iNF517), and ***P. putida*** KT2440 (iJN1463). *L. lactis* was rather constrained, which could affect feasibility of the model. Thus, all uptake reactions lower bounds were set to -30 (except glucose and ribose) and all secretion reactions upper bounds were set to 1000.
2. The objective function was set as biomass production in all three models.
3. The models were duplicated, and each copy were forced to grow on a different carbon source, being glucose and ribose. Flux values were calculated (see *Table 3*).
4. A COBRA flux variability analysis (FVA) was performed to each of the 6 models (3 species * 2 carbon sources). The output of this analysis is the range of possible fluxes of every reaction in a model given an objective function, in this case, biomass. This was carried out in order to select a set of relevant reactions of each model which

removal would affect the biomass production. The number of reactions can be seen on *Table 3.*

Species	Total Reactions	Flux Value (Glc)	Flux Value (Rib)	Relevant Reactions
E. coli	2712	0.876997	0.688913	430
L. lactis	754	0.801207	0.735738	325
P. putida	2927	0.586117	0.806667	596

Table 3. Original data for GGF validation. Flux units are mmol/(gDW·h). The last column represents the number of relevant reactions for biomass production, to wit, all reactions with at least one of the maximum or minimum fluxes nonzero.

5. A function for the elimination of a desired number of random reactions was designed. This function takes as input a model, a set of reaction IDs (the set of relevant reactions of that model) and the number of reactions of that set wanted to be removed from the model. The tests consist on **removing a different number of random reactions of the models, and observing how many reactions are reincorporated with COBRA and GGF**.
6. Four tests were performed, removing 1, 2, 5 and 10 reactions from the models. To assure robustness, 10 iterations were made over every test/species/carbon source combination. That means that a total of 24 gap filling procedures were performed by COBRA, and other 24 were carried out by GGF, each one of them repeated 10 times. In both cases, the template used for gap filling is the original model.

The full process can be accessed through the next link:

https://github.com/fcomnozz/GALLANT_gapfilling/blob/main/validation/validation.ipynb

The results are shown in tables 4-7.

Organism	Carbon source	COBRA						GGF					
		SM	AR	Tr	Ex	O	Time	SM	AR	Tr	Ex	O	Time
E. coli	Glucose	40	40	0	0	100	2min 40s	60	60	0	0	100	3min 17s
	Ribose	40	40	0	0	100	2min 43s	70	70	28.6	0	71.4	3min 22s
L. lactis	Glucose	40	40	0	0	100	47.2s	80	80	12.5	12.5	75	58.4s
	Ribose	30	30	0	0	100	47.5s	80	80	25	12.5	62.5	58.1s
P. putida	Glucose	0	0	-	-	-	2min 58s	50	50	40	0	60	3min 26s
	Ribose	0	0	-	-	-	3min 1s	20	20	50	50	0	3min 12s

Table 4. Validation test 1: elimination of 1 reaction. For each pair organism/carbon source 20 iterations of reaction elimination – gap filling has been carried out, 10 for each of the algorithms (COBRA, GGF). Legend: **SM) Solved Models (%)**, percentage of models reincorporating reactions. **AR) Added Reactions (%)**, percentage of the removed reactions which have been re-added with gap filling. **Tr) Transport Reactions (%)**, percentage of the added reactions with transport function. **Ex) Exchange Reactions (%)**, percentage of the added reactions with exchange function (diffusion). **O) Other Reactions (%)**, percentage of the added reactions with a function other than transport of exchange. **Time) Time employed on the execution of 10 gap filling procedures**.

As seen on *Table 4*, when removing one reaction, COBRA is able to fix 0 - 40% of the models, while GGF does it with 20 - 80% of them. As only one reaction is deleted from the models, the added reactions and solved models' percentages are the same (this will change in the next tests). It is worth mentioning that all reactions incorporated by cobra are non-transport and non-exchange (hereinafter general), while GGF is able to insert all three kinds of reactions. A noticeable case is *P. putida*, which incorporates few reactions (that will be more notorious in the next tests). This is arguably due to the metabolic robustness in this organism (Nogales, Gudmundsson et al. 2017), which possess several pathways for glucose degradation (Castillo, Ramos et al. 2007). Then, when an important reaction is removed, another one substitutes its function, and gap filling is thus not needed.

GGF exchange and transport additions are not solver-dependent, therefore, can be added even if the model fails to validate. GGF has added both transport/exchange and general reactions, the first type due to the mentioned before, and the second type arguably for the lower integrality threshold of GGF, which helps avoiding infeasibility problems.

Organism	Carbon source	COBRA						GGF					
		SM	AR	Tr	Ex	O	Time	SM	AR	Tr	Ex	O	Time
E. coli	Glucose	50	35	0	0	100	2min 46s	90	65	15.4	7.7	76.9	3min 36s
	Ribose	60	40	0	12.5	87.5	2min 41s	100	75	6.7	20	73.3	3min 39s
L. lactis	Glucose	90	55	18.2	18.2	63.6	47.1s	100	80	25	18.8	56.2	1min 2s
	Ribose	40	30	0	16.7	83.3	47.4s	60	40	0	25	75	57.5s
P. putida	Glucose	0	0	-	-	-	3min 4s	50	30	0	16.7	83.3	3min 30s
	Ribose	0	0	-	-	-	2min 59s	10	5	0	0	100	3min 8s

Table 5. Validation test 2: elimination of 2 reactions.

Looking at *Table 5*, COBRA performance seems to have improved if compared to test 1. In this case it has been able to insert exchange and transport reactions. The best working model is *L. lactis* growing on glucose, with a success of 90%, inserting 55% of reactions (11/20). Nevertheless, GGF has better stats for this case, solving 100% of models, inserting 80% of reactions. The other results are also better than COBRA, filling models with al kinds of

reactions, mostly general. *P. putida* keeps giving poor results, and the will go even worse over the next two tests.

Organism	Carbon source	COBRA						GGF					
		SM	AR	Tr	Ex	O	Time	SM	AR	Tr	Ex	O	Time
E. coli	Glucose	10	8	25	0	75	2min 48s	80	56	21.4	10.7	67.9	3min 35s
	Ribose	10	6	0	0	100	2min 44s	70	52	26.9	0	73.1	3min 27s
L. lactis	Glucose	40	22	9.1	18.2	72.7	46.2s	80	60	23.3	13.3	63.4	58.9s
	Ribose	40	26	7.7	7.7	84.6	46.5s	90	78	23.1	12.8	64.1	1min 1s
P. putida	Glucose	0	0	-	-	-	2min 58s	10	4	0	0	100	3min 6s
	Ribose	0	0	-	-	-	3min	10	6	33.3	0	66.7	3min 8s

Table 6. Validation test 3: elimination of 5 reactions.

In test 3, differences between COBRA and GGF rise even more, with a 10-40% of models solved by COBRA, and a 70-90% by GGF (both aside *P.putida*). The number and type-variability of added reactions is rather higher in GGF as well.

Organism	Carbon source	COBRA						GGF					
		SM	AR	Tr	Ex	O	Time	SM	AR	Tr	Ex	O	Time
E. coli	Glucose	10	4	25	0	75	2min 44s	80	47	17	12.8	70.2	3min 33s
	Ribose	0	0	-	-	-	2min 42s	70	45	17.8	11.1	71.1	3min 25s
L. lactis	Glucose	20	14	14.3	7.2	78.5	46.2s	50	41	17.1	14.6	68.3	57.3s
	Ribose	20	16	12.5	0	87.5	48s	40	34	20.6	11.8	67.6	55s
P. putida	Glucose	0	0	-	-	-	3min 3s	0	0	-	-	-	3min 4s
	Ribose	0	0	-	-	-	2min 58s	0	0	-	-	-	3min

Table 7. Validation test 4: elimination of 10 reactions.

The results of test 4 prove that COBRA presents some troubles gap-filling very incomplete models, with a maximum of 20% success. The percentage of added reactions is also rather low, not surpassing 20% in any case. GGF lowers it performance as well. However, the results are still better than COBRA's, and up to 80% of models have been solved. In both cases, *P. putida* has become totally infeasible.

In conclusion, GGF has prove to outshine COBRA in every single case, filling more models and adding more reactions. One of the keys of GGF is its handling of transport/exchange reactions, which seem to be problematic when using traditional gap filling algorithms, based on an optimization function. At times, the addition of these reactions has facilitated the addition of other reactions, which can improve or fix the model. Regarding execution time, as expected, it is lower in COBRA as it doesn't have an additional management of

transport/exchange reactions. The differences in time rise as the size of the models do it, with an average difference of 19.68%. However, the improvement of gap filling results makes this higher execution time worth it.

Comparisons against other gap filling algorithms like GrowMatch or FASTGAPFILL could have been convenient, but unfortunately, the requisites of these methods make impossible the usage of this particular set.

Gap Filling Application to a Multi-Strain GEM

Although the designed module can be applied on any SBML model, it has been specifically designed for its implementation in a multi-strain GEM generation pipeline. To show a possible application of this pipeline, which also presents a great relevance, **a multi-strain GEM from *Lactococcus lactis* have been generated with Gallant and gap-filled with GGF.**

The BiGG model for *L. lactis* (iNF517), as well as 43 NCBI Nucleotide IDs belonging to different strains of this organism were used by Gallant as input for the reconstruction of 43 strain-specific models which constitute a multi-strain GEM. These 43 models were the input for GGF.

The whole process can be found on a Jupyter notebook in the following link:

https://github.com/fcomnozz/GALLANT_gapfilling/blob/main/experiment/Experiment.ipynb

As the reference model iNF517 was very constrained, all uptake and secretion reactions flux bounds (except for glucose uptake) were enlarged, in order to give the model freedom degrees. The same procedure was performed onto all the models constituting the multi-strain GEM. Uptake lower bounds were set to -30, and secretion upper bounds were set to 1000. Glucose uptake lower bound kept -2.12. All models shared the same objective function, biomass production. Only 4 of 43 models presented a flux at the start of the process (see *Table 8*), with 15 infeasible models, and 24 with null flux. The reaction number vary between 669 and 749, while the reference model contains 754 reactions. The above all variability of the strains can be seen in *Fig. 13*.

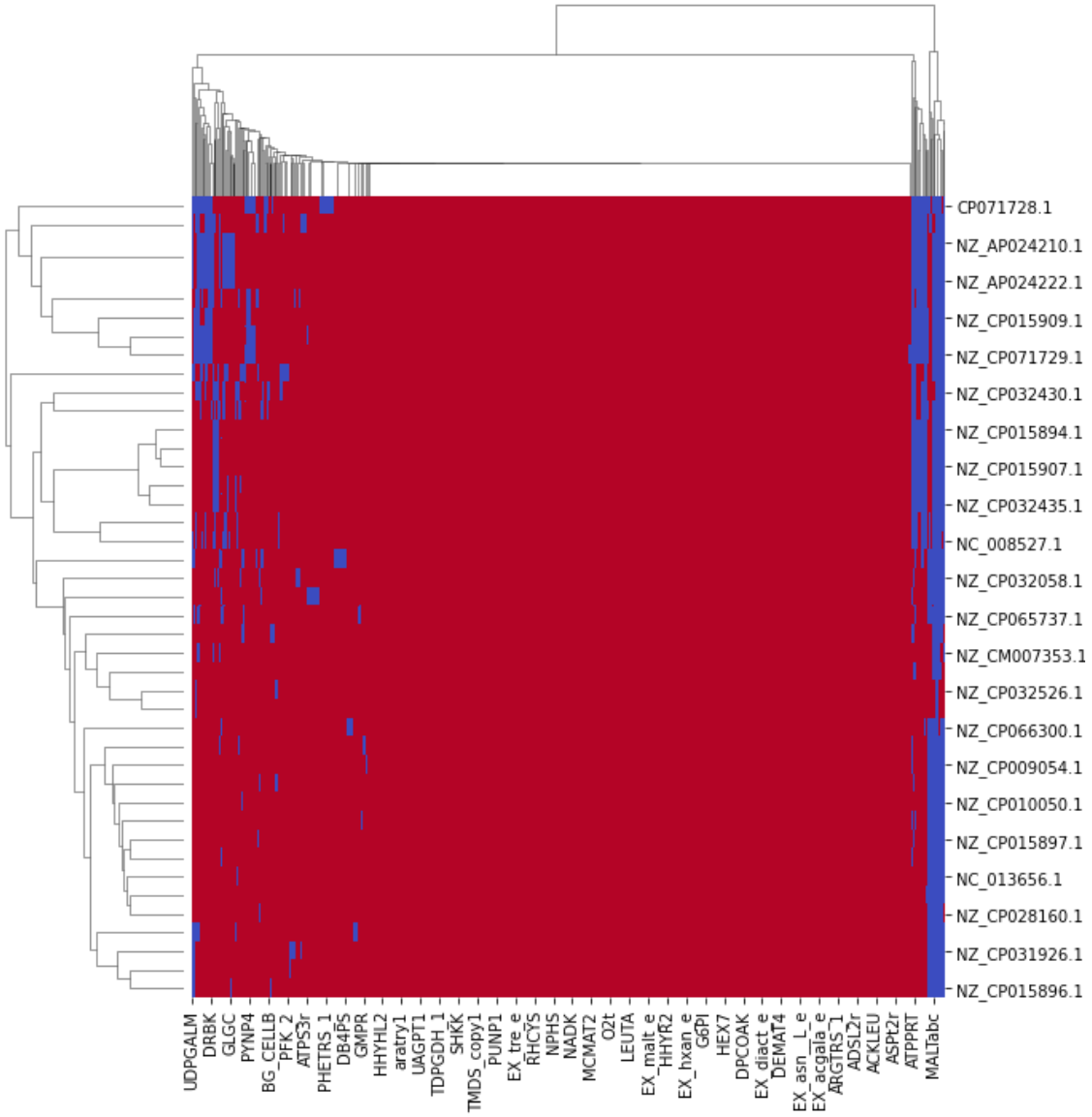

Fig. 13. Initial strain variability. Axis' labels do not represent the totality of strains or reactions to assure the clearness of the figure. Blue spots represent missing reactions. These 43 strain-specific models were reconstructing using Gallant workflow, employing the model iNF517 as a reference. Looking at the left vertical axis, approximately 5 clusters of related strains can be observed.

Fluxes were calculated and registered, and gap filling was performed upon each using the following settings:

- Template: iNF517 (reference *L. lactis*, 754 reactions)
- Integer threshold: 1-e9

- Exchange reactions addition: yes
- Transport reactions addition: yes, default options

Results are shown on *Table 8.*

NCBI ID	Strain	Initial Flux	NR	TR Added	OR Added	Post-GF Flux
NC_009004.1 (reference)	*L. lactis* subsp. cremoris MG13163	0.801207	754	-	-	-
NZ_CP066300.1	*L. lactis* subsp. lactis Wikim0098	0.0	729	0	5	0.800490
NZ_CP015899.1	*L. lactis* subsp. cremoris JM1	Infeasible	694	3	7	0.619201
NZ_CP015900.1	*L. lactis* subsp. cremoris JM2	Infeasible	699	6	2	0.575862
NZ_AP018499.1	*L. lactis* subsp. cremoris C4	0.0	693	3	2	0.649308
NC_020450.1	*L. lactis* subsp. lactis IO-1	0.0	735	0	2	0.649477
NZ_CP015894.1	*L. lactis* subsp. cremoris 158	Infeasible	716	1	2	0.613494
NZ_CP015901.1	*L. lactis* subsp. cremoris JM3	0.0	716	2	1	0.649308
NZ_LT599049.1	*L. lactis* subsp. lactis A12	0.0	732	0	2	0.800490
NC_008527.1	*L. lactis* subsp. cremoris SK11	0.649298	706	4	0	0.649298
NZ_CP015902.1	*L. lactis* subsp. lactis UC06	0.0	732	1	2	0.727794
NZ_CP010050.1	*L. lactis* subsp. lactis S0	0.0	734	0	2	0.800490
NZ_CP032526.1	*L. lactis* subsp. cremoris 4B0	0.0	746	1	2	0.801206
NZ_CP032430.1	*L. lactis* subsp. cremoris W34	Infeasible	702	4	2	0.567316
NZ_AP024222.1	*L. lactis* subsp. cremoris EPSC	0.0	693	3	2	0.649308
NC_019435.1	*L. lactis* subsp. cremoris UC509.9	Infeasible	717	1	2	0.568396
NZ_CP070856.1	*L. lactis* subsp. cremoris GR0507	0.0	735	0	0	0.0
NZ_CP015907.1	*L. lactis* subsp. cremoris UC109	Infeasible	718	0	2	0.568396
NZ_CP061322.1	*L. lactis* subsp. lactis S50	0.0	729	2	2	0.800490
NZ_AP024210.1	*L. lactis* subsp. cremoris G3-2	0.0	693	3	2	0.649308
NZ_CP015896.1	*L. lactis* subsp. lactis 229	0.0	726	0	0	0.0
NZ_CM007353.1	*L. lactis* subsp. cremoris IBB477	0.644121	739	2	0	0.800490
CP071728.1	*L. lactis* subsp. cremoris FM-YL12	Infeasible	669	0	0	Infeasible
NZ_CP032058.1	*L. lactis* subsp. lactis 267	0.0	719	1	2	0.749508
NZ_CP015909.1	*L. lactis* subsp. cremoris JM4	Infeasible	703	2	2	0.649467
NZ_CP051518.1	*L. lactis* subsp. cremoris F	Infeasible	713	1	2	0.566875
NC_017492.1	*L. lactis* subsp. cremoris A76	Infeasible	697	5	4	0.536657
NZ_CP032435.1	*L. lactis* subsp. cremoris B26	Infeasible	715	1	2	0.566875
NZ_CP065737.1	*L. lactis* subsp. lactis FDAARGOS_865	0.0	722	2	2	0.781589
NZ_CP015897.1	*L. lactis* subsp. lactis 275	0.0	733	1	2	0.800490
NC_013656.1	*L. lactis* subsp. lactis KF147	0.0	735	0	2	0.800490
NZ_CP031926.1	*L. lactis* subsp. lactis 223	0.0	724	3	3	0.800490
NZ_CP059049.1	*L. lactis* subsp. lactis N8	0.0	732	1	2	0.800490
NC_022369.1	*L. lactis* subsp. cremoris KW2	0.801206	743	0	0	0.801206
NZ_CP028160.1	*L. lactis* subsp. lactis 14B4	0.0	736	0	2	0.801206
NZ_CP020604.1	*L. lactis* subsp. lactis FM03	0.0	720	3	2	0.800490
NC_017949.1	*L. lactis* subsp. cremoris NZ9000	0.801206	749	0	0	0.801206
NZ_CP031538.1	*L. lactis* subsp. cremoris 3107	Infeasible	691	3	3	0.649467
NZ_CP032148.1	*L. lactis* subsp. cremoris 1196	Infeasible	708	1	2	0.569810
NZ_CP009054.1	*L. lactis* subsp. lactis 2118	0.0	732	1	2	0.800321
NZ_CP024954.1	*L. lactis* subsp. lactis F44	Infeasible	719	1	3	0.201151
NZ_CP015908.1	*L. lactis* subsp. lactis UL8	0.0	730	0	3	0.800490
NZ_CP071729.1	*L. lactis* subsp. cremoris FM-YL11	Infeasible	690	3	3	0.649467
NZ_CP053671.2	*L. lactis* subsp. lactis G121	0.0	708	0	0	0.0

Table 8. Gap filling results for the multi-strain GEM of *L. lactis*. Fluxes values have been rounded for display. The first four columns represent the original strain-specific models (ID, strain name, initial flux value and number of reactions). All of these were reconstructed using iNF517 (754 reactions) as reference. Fifth and sixth columns show the number of transport and general reactions added to each model through GGF. No exchange reaction has been added to any model. Seventh column represents the biomass production flux after gap filling. The first row shows the reference model's number of reactions and flux for comparison. Flux units are mmol/(gDW·h). Models with no flux/infeasible are marked in red.

As seen on the table above, 39 of 43 models wee functional after GGF execution. The four models being functional before gap filling remain the same, except strain IBB477, which even raised its flux. Strain models GR0507, 229, FM-YL11 and G121, were not solved, though. Regarding the added reactions, 43.62% were transport reactions and 56.38% were other reactions, while no exchange reactions were added (*Fig. 14*).

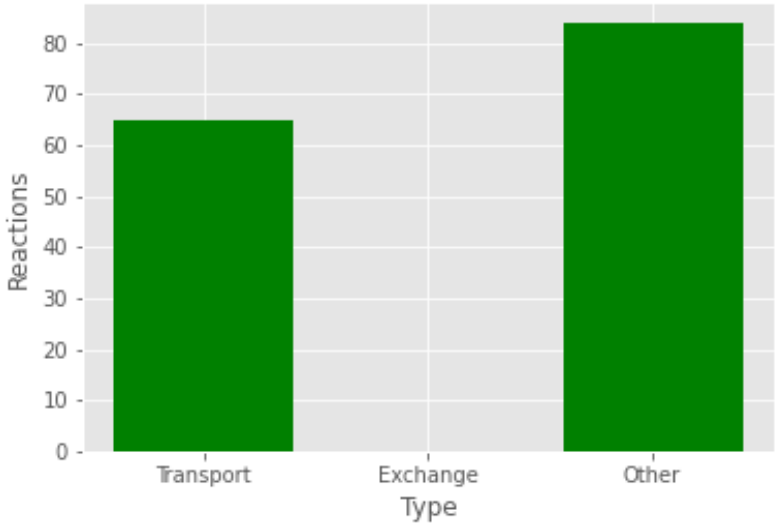

Fig. 14. Total number of added reactions. No exchange reactions were added.

In order to address the four non-functional models, an additional step was applied over these GEMs. Using the ModelSEED access submodule, 10 new *L. lactis* models were reconstructed from PATRIC genomes (see *Table 9*), which were used as templates in a new gap filling step over these non-functional models.

Genome Name	Genome ID
Lactococcus lactis strain Olga-2	1.358.142
Lactococcus lactis strain IPLA1064	1.358.538
Lactococcus lactis subsp. cremoris A17	1359.20
Lactococcus lactis subsp. cremoris strain NCDO763	1359.42
Lactococcus lactis subsp. lactis NCDO 2118	1117941.3
Lactococcus lactis Bpl1	1358.50

Lactococcus lactis subsp. cremoris strain B40	1359.39
Lactococcus lactis subsp. lactis JCM 5805	1360.89
Lactococcus lactis subsp. lactis YF11	1149132.3
Lactococcus lactis strain SP2C4	13.581.193

Table 9. PATRIC IDs used for reconstruction of templates for non-functional models. These models were reconstructed at June 19 2021, 22:39.

A total of 10 *L. lactis* models were reconstructed from PATRIC, gap-filled by ModelSEED using its universal database and translated to BiGG nomenclature. After gap-filling the four non-functional models using these PATRIC models as templates (forcing the use of all templates for every model), no reaction was added, and models remained unfunctional.

A total of 149 reactions were added to 39 of the 43 samples during the whole gap filling process. However, these weren't unique reactions, but a total of 31 different reactions. In *Fig. 15* a clustered heatmap showing reactions added to each model can be seen. It also shows the possible relationship between models or reactions by clustering.

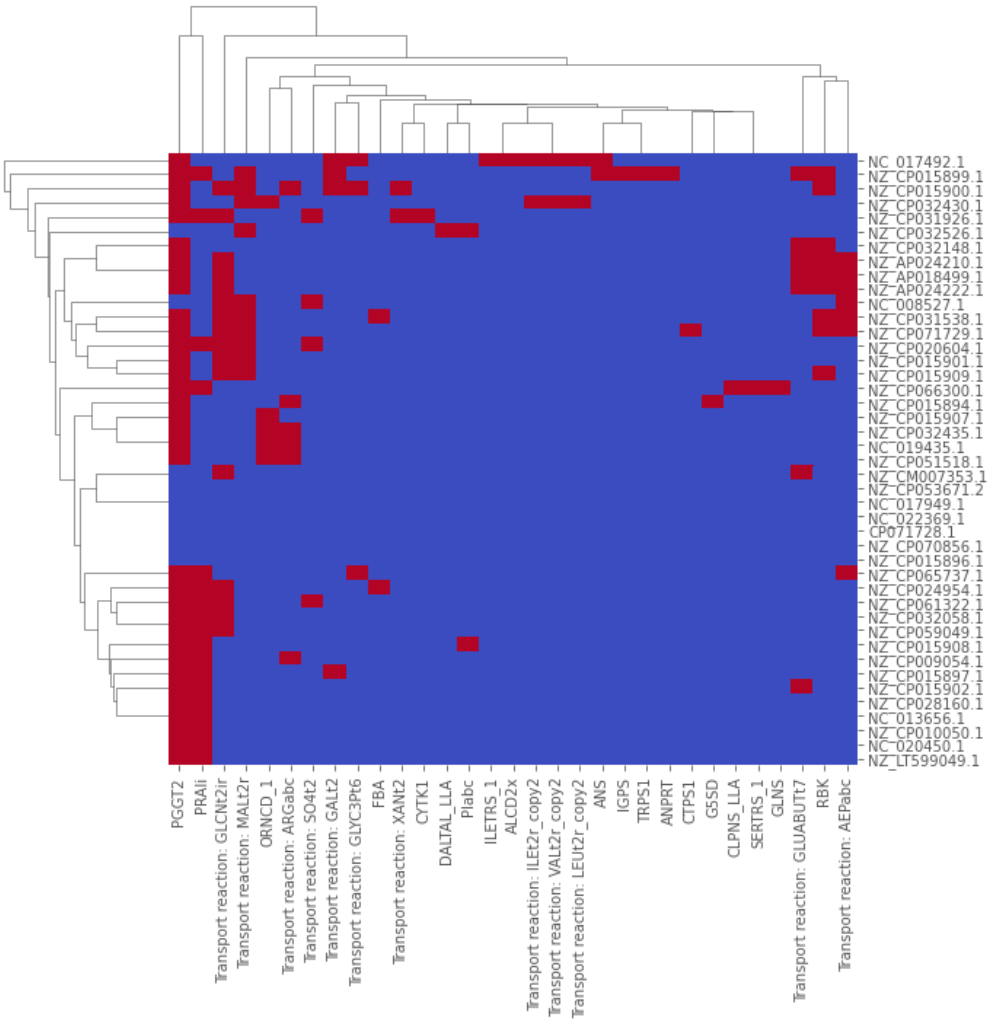

Fig. 15. Clustered heatmap of models and reactions. This graph represents which reactions have been added to models (red). The vertical axis shows the different models, while the horizontal axis shows all different reactions. Although model clusters are not exactly the same as those made by measuring all reactions in models, it still keeps a similar shape, reinforcing the relationship between strains.

Discussion

Advantages of GGF

Looking at the results, it can be concluded that Gallant has been improved with the implementation of this module. Most of the models have been fixed or enhanced. The key of this method is the usage of two different approaches. On the one hand, a classic MILP algorithm, based on an optimization problem, is used to add the least possible numbers of reactions in order to improve it flux. This is a conservative approach where few reactions are added. Being a MILP algorithm, the execution takes some time, but it is not something immeasurable; it usually goes from minutes to a few hours, depending on the number and size of the models, and the features of the host computer/server. For example, during the method validation, 480 independent gap filling processes were performed in approximately 2 hours; that is an average of a process every 15 seconds. On the other hand, as transport and exchange reactions proved to be less prone to be added, a more "mechanical" approach was developed: if a metabolite is present in a query model, and there is an exchange/transport reaction involving that metabolite in a template, that reaction is added to the query model. It is worth remembering that this module was design to use homologous models as templates, so this last approach makes more sense when using this kind of templates. That is the reason why these steps are optional, and should be used carefully. Finally, a step that practically is a pipeline on its own adds more flexibility to the module by allowing the reconstruction of new models which can be used as templates, involving the resources of PATRIC, ModelSEED and Mackinac.

Although the core of this module is built upon COBRA gap filling algorithm, it has surpassed it, having the following advantages:

- **Modularity**: the structure of this module consists in two principal functions, composed of smaller functions, which can be combined or used independently. Being all written in Python language, the implementation of one or more of these functions in a pipeline is not a problem.

- **Flexibility**: the module has a variety of functions, options and combinations, meant to satisfy the preferences of the user. Gap filling can be performed using all templates or just the most similar ones, exchange and transport reactions can be forced to be added, even choosing between three different criteria to determine whether a reaction is considered to be a transport one, and additional models can be used as templates.
- **Better error management**: all possible errors during execution have been taken into account, avoiding potential crashes. This allows the user to execute the workflow overnight having the certainty that the whole process will be completed.
- **Added reactions and genes log:** all reaction and gene names added by gap filling are stored into dictionaries or lists, which allows user to have a register of which reactions have been added, which template come they from, or if it was added during standard gap filling or exchange/transport reactions addition. These logs can be helpful in the generation of automated reports, as well as the curation of the models.
- **ModelSEED access**: although previously commented, this function adds a valuable tool to this module as it can reconstruct a GEM from any PATRIC genome and gap-fill it using ModelSEED resources. Taking into account that PATRIC contained ~250000 different genomes by 2019, this submodule really expands the possibilities.

***Lactococcus lactis* multi-strain GEM gap filling**

As said before, multi-strain GEMs have not been fully explored. In fact, only three of these models have been published in BiGG, being *Acinetobacter baumannii* AYE (Norsigian, Kavvas et al. 2018), *Salmonella Typhimurium* (Seif, Kavvas et al. 2018) and *Klebsiella pneumoniae* (Norsigian, Attia et al. 2019). Other published multi-strain GEMs include *Escherichia coli* (Monk, Charusanti et al. 2013) or *Staphylococcus aureus* (Bosi, Monk et al. 2016). It is noticeable how all these models have some kind of medical relevance. That would make our *L. lactis* the first multi-strain GEM with relevance in food industry ever constructed. The development of this model could allow the design of new applications of *L. lactis* in industry, growth optimization or fermentation enhancing, among other.

Species	BiGG Accession	Applications
Klebsiella pneumoniae	http://bigg.ucsd.edu/models/iYL1228	Study of pathogenicity, drug resistance, gene essentiality, serovars, adaptation to environments
Salmonella Typhimurium	http://bigg.ucsd.edu/models/STM_v1_0	
Acinetobacter baumannii	http://bigg.ucsd.edu/models/iCN718	
Escherichia coli	-	
Staphylococcus aureus	-	

Lactococcus lactis	-	Fermentation enhancing, flavour formation of dairy products, designing of new industrial applications

Table 10. Comparison between published and *L. lactis* multi-strain GEMs applications.

The gap filling has improved at least 36 strain-specific *L. lactis* GEMs, fixing most of them. The four failed models could not be solved even using PATRIC/ModelSEED models as templates. This arguably means that the problem is not in the templates but in the own model (or maybe the solver's limitations). Despite reference model was a *L. lactis* subsp. cremoris, unfunctional models belong equally to this subspecies and subspecies lactis. The only infeasible model was the one with the least number of reactions. These models should probably be checked and compared with the functional ones, in order to identify the missing reactions that make the other models work. These reactions could be manually added then.

NCBI ID	Species/Strain	Number of Reactions	Flux Status
NZ_CP070856.1	*L. lactis* subsp. cremoris strain GR0507	735	0
NZ_CP015896.1	*L. lactis* subsp. lactis strain 229	726	0
CP071728.1	*L. lactis* subsp. cremoris strain FM-YL12	669	Infeasible
NZ_CP053671.2	*L. lactis* subsp. lactis strain G21	708	0

Table 11. Unsolved models.

In regard to added reactions, only 31 different reactions (see *Fig.* 15) were added to the models, fixing them. Therefore, these reactions must be important for the biomass production. In *Fig. 16* the 10 most added reactions are shown.

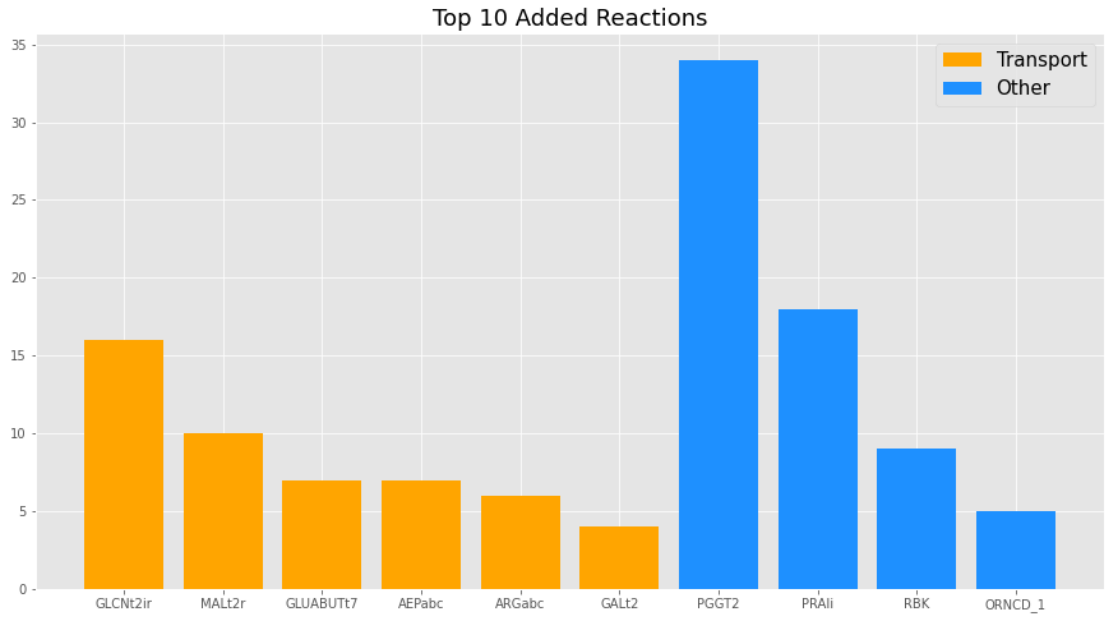

Fig. 16. Top 10 added reactions. Y axis represents the total number of reactions in axis X added to the multi-strain GEM.

In the bar-plot above, shows how PGGT2 reaction was added in 34 of 37 models (KW2 and NZ9000 were not needed to be gap-filled), followed by PRAIi, inserted in 18 models, transport reactions GLCNt2ir and MALt2r, and RBK. Next, these five reactions will be discussed.

PGGT2 (http://bigg.ucsd.edu/universal/reactions/PGGT2)

This reaction, catalysed by a peptidoglycan glycosyltransferase (EC 2.4.1.129), is essential in the formation of peptidoglycan in *L. lactis*. As a Gram-positive bacteria, *L. lactis* uses peptidoglycan as major component for its cell wall, which is needed to maintain structural integrity and stability. It is reasonable to think that bacteria unable to produce their cell wall won't grow at all. Thus, the addition of this reaction will allow bacteria to solve this issue.

PRAIi (http://bigg.ucsd.edu/universal/reactions/PRAIi)

PRAIi belongs to phosphoribosylanthranilate isomerase (EC 5.3.1.24), an enzyme involved in the biosynthesis of phenylalanine, tyrosine and tryptophan (https://www.genome.jp/dbget-bin/www_bget?lla:L0050), which as amino acids are needed for anabolism and therefore, essential for growth.

GLCNt2ir (http://bigg.ucsd.edu/universal/reactions/GLCNt2ir)

This reaction consists on the D gluconate transport via proton symport. D-gluconate can be used as carbon source for *L. lactis* (Passerini, Coddeville et al. 2013), so it seems that its transport through membrane would be able to rise biomass production.

MALt2r (http://bigg.ucsd.edu/universal/reactions/MALt2r)

MALt2r stands for L-malate transport via proton symport. Like the previous case, it can be used as carbon source since it is part of citric acid cycle, an energy-producing pathway.

RBK (http://bigg.ucsd.edu/universal/reactions/RBK)

Ribokinase (EC 2.7.1.15) catalyses the phosphorylation of D-ribose, consuming ATP. This enzyme belongs to the pentose phosphate pathway, which produces lactic acid in *L. lactis (Shinkawa, Okano et al. 2011)*, being its main industrial fermentation product. Therefore, the addition of this reaction wouldn't just be profitable for these bacteria's metabolism, but also for industrial purposes.

Conclusion

- **The designed gap filling module has notoriously improved *L. lactis* multi-strain model.** Its usage has allowed the functioning of the most part of the strain-specific models.

- **GGF has proven to surpass COBRA's gap filling algorithm in a multi-strain GEM generation scope.** Its strengths are its modularity, flexibility, transport/exchange reactions management and PATRIC/ModelSEED access. However, it has only be tested on processes where query and template models belong to the same species, and it would be convenient to test it over other types of datasets.

- **One of the first multi-strain GEMs with relevance in food industry has been generated and gap-filled.** This model can be used in order to improve fermentation, biomass production, flavour development and other strain-dependent aspects.

- **Another application for gap filling not explored within this work is the identification of suitable genes for genetic transformation.** If an organism other the query model is used as template, reactions/genes that improve a certain pathway can be identified.

- **Although the gap filling results have been satisfactory, some improvements could be made in the future.** Some of these upgrades may be the implementation of other gap filling algorithms or the use of other solvers, like CPLEX.

Abbreviations

GEM: Genome-scale Metabolic Model

FBA: Flux Balance Analysis

GGF: Gallant Gap Filling

SBML: Systems Biology Markup Language

GPR: Gene-Protein-Reaction

MILP: Mixed-Integer Linear Programming

Bibliography

(EFFCA), E. F. F. C. A. "Microorganisms in food production." Retrieved May 11, 2021, from https://effca.org/microbial-cultures/food-production.

Adak, A. and M. R. Khan (2019). "An insight into gut microbiota and its functionalities." Cell Mol Life Sci **76**(3): 473-493.

Arakawa, K., Y. Yamada, K. Shinoda, Y. Nakayama and M. Tomita (2006). "GEM System: automatic prototyping of cell-wide metabolic pathway models from genomes." BMC Bioinformatics **7**(1): 168.

Arkin, A. P., R. W. Cottingham, C. S. Henry, N. L. Harris, R. L. Stevens, S. Maslov, P. Dehal, D. Ware, F. Perez, S. Canon, M. W. Sneddon, M. L. Henderson, W. J. Riehl, D. Murphy-Olson, S. Y. Chan, R. T. Kamimura, S. Kumari, M. M. Drake, T. S. Brettin, E. M. Glass, D. Chivian, D. Gunter, D. J. Weston, B. H. Allen, J. Baumohl, A. A. Best, B. Bowen, S. E. Brenner, C. C. Bun, J.-M. Chandonia, J.-M. Chia, R. Colasanti, N. Conrad, J. J. Davis, B. H. Davison, M. DeJongh, S. Devoid, E. Dietrich, I. Dubchak, J. N. Edirisinghe, G. Fang, J. P. Faria, P. M. Frybarger, W. Gerlach, M. Gerstein, A. Greiner, J. Gurtowski, H. L. Haun, F. He, R. Jain, M. P. Joachimiak, K. P. Keegan, S. Kondo, V. Kumar, M. L. Land, F. Meyer, M. Mills, P. S. Novichkov, T. Oh, G. J. Olsen, R. Olson, B. Parrello, S. Pasternak, E. Pearson, S. S. Poon, G. A. Price, S. Ramakrishnan, P. Ranjan, P. C. Ronald, M. C. Schatz, S. M. D. Seaver, M. Shukla, R. A. Sutormin, M. H. Syed, J. Thomason, N. L. Tintle, D. Wang, F. Xia, H. Yoo, S. Yoo and D. Yu (2018). "KBase: The United States Department of Energy Systems Biology Knowledgebase." Nature biotechnology **36**(7): 566-569.

Aziz, R. K., D. Bartels, A. A. Best, M. DeJongh, T. Disz, R. A. Edwards, K. Formsma, S. Gerdes, E. M. Glass, M. Kubal, F. Meyer, G. J. Olsen, R. Olson, A. L. Osterman, R. A. Overbeek, L. K.

McNeil, D. Paarmann, T. Paczian, B. Parrello, G. D. Pusch, C. Reich, R. Stevens, O. Vassieva, V. Vonstein, A. Wilke and O. Zagnitko (2008). "The RAST Server: Rapid Annotations using Subsystems Technology." BMC Genomics **9**(1): 75.

Baeshen, N. A., M. N. Baeshen, A. Sheikh, R. S. Bora, M. M. Ahmed, H. A. Ramadan, K. S. Saini and E. M. Redwan (2014). "Cell factories for insulin production." Microb Cell Fact **13**: 141.

Bazzani, S. (2014). "Promise and reality in the expanding field of network interaction analysis: metabolic networks." Bioinformatics and biology insights **8**: 83-91.

Bogdanov, M., E. Mileykovskaya and W. Dowhan (2008). "Lipids in the assembly of membrane proteins and organization of protein supercomplexes: implications for lipid-linked disorders." Subcell Biochem **49**: 197-239.

Bosi, E., J. M. Monk, R. K. Aziz, M. Fondi, V. Nizet and B. Palsson (2016). "Comparative genome-scale modelling of Staphylococcus aureus strains identifies strain-specific metabolic capabilities linked to pathogenicity." Proc Natl Acad Sci U S A **113**(26): E3801-3809.

Cameron English, K. S. (2020). "Where are GMO crops grown? GLP infographics document the global growth of agricultural biotechnology innovation." Retrieved May 17, 2021, from https://geneticliteracyproject.org/2020/04/28/where-are-gmo-crops-grown-glp-infographics-document-the-global-growth-of-agricultural-biotechnology-innovation/.

Castillo, T. d., J. L. Ramos, J. J. Rodríguez-Herva, T. Fuhrer, U. Sauer and E. Duque (2007). "Convergent Peripheral Pathways Catalyze Initial Glucose Catabolism in Pseudomonas putida: Genomic and Flux Analysis." **189**(14): 5142-5152.

Deckers, M., D. Deforce, M.-A. Fraiture and N. H. J. F. Roosens (2020). "Genetically modified micro-organisms for industrial food enzyme production: An overview." **9**(3): 326.

Ebrahim, A., J. A. Lerman, B. O. Palsson and D. R. Hyduke (2013). "COBRApy: COnstraints-Based Reconstruction and Analysis for Python." BMC Syst Biol **7**: 74.

Edwards, J. S. and B. O. Palsson (1999). "Systems properties of the Haemophilus influenzae Rd metabolic genotype." J Biol Chem **274**(25): 17410-17416.

Elcacho, J. (2017). "España concentra el 95% de los cultivos transgénicos de Europa." Retrieved May 17, 2021, from https://www.lavanguardia.com/natural/20170506/422312083177/informe-mundial-cultivos-transgenicos-espana-lider-europa.html.

Ercolini, D. and V. Fogliano (2018). "Food Design To Feed the Human Gut Microbiota." Journal of agricultural and food chemistry **66**(15): 3754-3758.

Faria, J. P., M. Rocha, I. Rocha and C. S. Henry (2018). "Methods for automated genome-scale metabolic model reconstruction." Biochem Soc Trans **46**(4): 931-936.

Gianchandani, E. P., A. K. Chavali and J. A. Papin (2010). "The application of flux balance analysis in systems biology." Wiley Interdiscip Rev Syst Biol Med **2**(3): 372-382.

Gillespie, J. J., A. R. Wattam, S. A. Cammer, J. L. Gabbard, M. P. Shukla, O. Dalay, T. Driscoll, D. Hix, S. P. Mane, C. Mao, E. K. Nordberg, M. Scott, J. R. Schulman, E. E. Snyder, D. E. Sullivan, C. Wang, A. Warren, K. P. Williams, T. Xue, H. S. Yoo, C. Zhang, Y. Zhang, R. Will, R. W. Kenyon and B. W. Sobral (2011). "PATRIC: the comprehensive bacterial bioinformatics resource with a focus on human pathogenic species." Infect Immun **79**(11): 4286-4298.

Gu, C., G. B. Kim, W. J. Kim, H. U. Kim and S. Y. Lee (2019). "Current status and applications of genome-scale metabolic models." Genome biology **20**(1): 121-121.

Hartleb, D., F. Jarre and M. J. Lercher (2016). "Improved Metabolic Models for E. coli and Mycoplasma genitalium from GlobalFit, an Algorithm That Simultaneously Matches Growth and Non-Growth Data Sets." PLoS Comput Biol **12**(8): e1005036.

Hatzimanikatis, V., C. Li, J. A. Ionita, C. S. Henry, M. D. Jankowski and L. J. J. B. Broadbelt (2005). "Exploring the diversity of complex metabolic networks." **21**(8): 1603-1609.

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

Hosseini, Z. and S. A. Marashi (2017). "Discovering missing reactions of metabolic networks by using gene co-expression data." Sci Rep **7**: 41774.

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

Kapur, R. (2019). "History and Scope of Food Microbiology."

Karp, P. D., D. Weaver and M. Latendresse (2018). "How accurate is automated gap filling of metabolic models?" BMC systems biology **12**(1): 73-73.

King, Z. A., J. Lu, A. Dräger, P. Miller, S. Federowicz, J. A. Lerman, A. Ebrahim, B. O. Palsson and N. E. Lewis (2016). "BiGG Models: A platform for integrating, standardizing and sharing genome-scale models." Nucleic acids research **44**(D1): D515-D522.

Kumar, V. S. and C. D. Maranas (2009). "GrowMatch: an automated method for reconciling in silico/in vivo growth predictions." PLoS computational biology **5**(3): e1000308-e1000308.

McCarty, N. S. and R. Ledesma-Amaro (2019). "Synthetic Biology Tools to Engineer Microbial Communities for Biotechnology." Trends in Biotechnology **37**(2): 181-197.

Monk, J. M., P. Charusanti, R. K. Aziz, J. A. Lerman, N. Premyodhin, J. D. Orth, A. M. Feist and B. Palsson (2013). "Genome-scale metabolic reconstructions of multiple Escherichia coli strains

highlight strain-specific adaptations to nutritional environments." Proc Natl Acad Sci U S A **110**(50): 20338-20343.

Mundy, M., H. Mendes-Soares and N. J. B. Chia (2017). "Mackinac: a bridge between ModelSEED and COBRApy to generate and analyze genome-scale metabolic models." **33**(15): 2416-2418.

Nogales, J., S. Gudmundsson, E. Duque, J. L. Ramos and B. O. Palsson (2017). "Expanding the computable reactome in Pseudomonas putida reveals metabolic cycles providing robustness." 139121.

Norsigian, C. J., H. Attia, R. Szubin, A. S. Yassin, B. Palsson, R. K. Aziz and J. M. Monk (2019). "Comparative Genome-Scale Metabolic Modeling of Metallo-Beta-Lactamase-Producing Multidrug-Resistant Klebsiella pneumoniae Clinical Isolates." Front Cell Infect Microbiol **9**: 161.

Norsigian, C. J., X. Fang, Y. Seif, J. M. Monk and B. O. Palsson (2020). "A workflow for generating multi-strain genome-scale metabolic models of prokaryotes." Nature protocols **15**(1): 1-14.

Norsigian, C. J., E. Kavvas, Y. Seif, B. O. Palsson and J. M. Monk (2018). "iCN718, an Updated and Improved Genome-Scale Metabolic Network Reconstruction of Acinetobacter baumannii AYE." Front Genet **9**: 121.

Norsigian, C. J., N. Pusarla, J. L. McConn, J. T. Yurkovich, A. Dräger, B. O. Palsson and Z. King (2020). "BiGG Models 2020: multi-strain genome-scale models and expansion across the phylogenetic tree." Nucleic acids research **48**(D1): D402-D406.

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

Olempska-Beer, Z. S., R. I. Merker, M. D. Ditto and M. J. DiNovi (2006). "Food-processing enzymes from recombinant microorganisms--a review." Regul Toxicol Pharmacol **45**(2): 144-158.

Orth, J. D. and B. Ø. Palsson (2010). "Systematizing the generation of missing metabolic knowledge." Biotechnology and bioengineering **107**(3): 403-412.

Pan, S. and J. L. Reed (2018). "Advances in gap-filling genome-scale metabolic models and model-driven experiments lead to novel metabolic discoveries." Curr Opin Biotechnol **51**: 103-108.

Passerini, D., M. Coddeville, P. Le Bourgeois, P. Loubière, P. Ritzenthaler, C. Fontagné-Faucher, M.-L. Daveran-Mingot and M. Cocaign-Bousquet (2013). "The carbohydrate metabolism signature of lactococcus lactis strain A12 reveals its sourdough ecosystem origin." Applied and environmental microbiology **79**(19): 5844-5852.

Reed, J. L., T. R. Patel, K. H. Chen, A. R. Joyce, M. K. Applebee, C. D. Herring, O. T. Bui, E. M. Knight, S. S. Fong and B. O. Palsson (2006). "Systems approach to refining genome annotation." Proceedings of the National Academy of Sciences of the United States of America **103**(46): 17480-17484.

Satish Kumar, V., M. S. Dasika and C. D. Maranas (2007). "Optimization based automated curation of metabolic reconstructions." BMC bioinformatics **8**: 212-212.

Seaver, S. M. D., S. Gerdes, O. Frelin, C. Lerma-Ortiz, L. M. T. Bradbury, R. Zallot, G. Hasnain, T. D. Niehaus, B. El Yacoubi, S. Pasternak, R. Olson, G. Pusch, R. Overbeek, R. Stevens, V. de Crécy-Lagard, D. Ware, A. D. Hanson and C. S. Henry (2014). "High-throughput comparison, functional annotation, and metabolic modeling of plant genomes using the PlantSEED resource." Proceedings of the National Academy of Sciences of the United States of America **111**(26): 9645-9650.

Seif, Y., E. Kavvas, J. C. Lachance, J. T. Yurkovich, S. P. Nuccio, X. Fang, E. Catoiu, M. Raffatellu, B. O. Palsson and J. M. Monk (2018). "Genome-scale metabolic reconstructions of multiple Salmonella strains reveal serovar-specific metabolic traits." Nat Commun **9**(1): 3771.

Sgobba, E. and V. F. Wendisch (2020). "Synthetic microbial consortia for small molecule production." Curr Opin Biotechnol **62**: 72-79.

Shinkawa, S., K. Okano, S. Yoshida, T. Tanaka, C. Ogino, H. Fukuda and A. Kondo (2011). "Improved homo L-lactic acid fermentation from xylose by abolishment of the phosphoketolase pathway and enhancement of the pentose phosphate pathway in genetically modified xylose-assimilating Lactococcus lactis." Appl Microbiol Biotechnol **91**(6): 1537-1544.

Simeonidis, E. and N. D. Price (2015). "Genome-scale modeling for metabolic engineering." Journal of industrial microbiology & biotechnology **42**(3): 327-338.

Thiele, I., N. Vlassis and R. M. Fleming (2014). "fastGapFill: efficient gap filling in metabolic networks." Bioinformatics **30**(17): 2529-2531.

Vlassis, N., M. P. Pacheco and T. Sauter (2014). "Fast Reconstruction of Compact Context-Specific Metabolic Network Models." PLOS Computational Biology **10**(1): e1003424.

Wanjek, C. (2011). "An Intellectual Resource for Integrative Biology." Retrieved May 21, 2021, from https://irp.nih.gov/catalyst/v19i6/systems-biology-as-defined-by-nih#:~:text=Systems%20biology%20is%20an%20approach,involves%20taking%20the%20pieces%20apart.

Wattam, A. R., D. Abraham, O. Dalay, T. L. Disz, T. Driscoll, J. L. Gabbard, J. J. Gillespie, R. Gough, D. Hix, R. Kenyon, D. Machi, C. Mao, E. K. Nordberg, R. Olson, R. Overbeek, G. D. Pusch, M. Shukla, J. Schulman, R. L. Stevens, D. E. Sullivan, V. Vonstein, A. Warren, R. Will, M. J. C. Wilson, H. S. Yoo, C. Zhang, Y. Zhang and B. W. Sobral (2014). "PATRIC, the bacterial bioinformatics database and analysis resource." Nucleic Acids Research **42**(D1): D581-D591.

Wunderlich, S. and K. A. Gatto (2015). "Consumer perception of genetically modified organisms and sources of information." Adv Nutr **6**(6): 842-851.