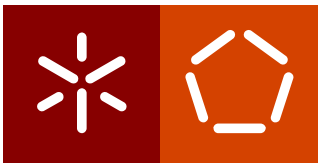
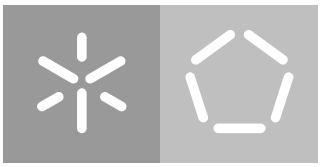


Universidade do Minho
Escola de Engenharia
Departamento de Informática

José Miguel Dantas de Sousa

In silico objective-driven optimization of microbial communities

November 2019



Universidade do Minho
Escola de Engenharia
Departamento de Informática

José Miguel Dantas de Sousa

In silico objective-driven optimization of microbial communities

Dissertation
Master Degree in Bioinformatics

Supervisors
Oscar Manuel Lima Dias
Miguel Francisco de Almeida Pereira da Rocha

November 2019

ACKNOWLEDGEMENTS

Por mais individual que seja a dissertação, como auge do meu percurso neste mestrado, esta seria impossível sem a ajuda e suporte de muitos intervenientes. Desse modo, serve este espaço para agradecer a todos os que contribuíram, direta ou indiretamente, para a realização desta dissertação.

Sem os quais nada seria possível, não posso deixar de agradecer esta oportunidade aos meus pais, que sempre me apoiaram nos estudos e os tornaram possíveis, inclusive na mudança de rota que me permitiu chegar aqui.

À Doutora Sara Correia, por me introduzir a este tema no projeto e, consequentemente, me ter desafiado a continuá-lo nesta dissertação.

Aos Professores Doutor Oscar Dias e Doutor Miguel Rocha, por me terem aceite como orientando nesta dissertação e pela paciência com os meus atrasos quanto aos prazos de escrita.

Ao Emanuel, Óscar e Diogo pelas inúmeras horas a fazer esgotantes trabalhos de grupo, discutir dissertações e, no geral, por partilharmos este final de uma autêntica maratona.

Aos membros do BiSBII, por me acolherem nestes últimos meses no grupo, quer como mestrando, quer como bolseiro, onde não me faltou ajuda, gargalhadas e lanches "ligeiramente" prolongados. Agradeço, especialmente: ao Diogo, novamente, por termos partilhado também este percurso, agora profissional, e por ter aturado infindáveis deambulações sobre assuntos que não interessam a ninguém; à Sophia, por toda a preciosa ajuda que me deu, já desde a altura do projeto, apesar de não estar oficialmente envolvida; à Marta, por todo o trabalho que cobriu por mim para eu conseguir acabar esta dissertação; e sem esquecer, também, ao Fernando, Davide e Bastos.

Por fim, mas sem menor importância, aqueles com quem nunca me separei apesar da distância ou trabalho, e que me ajudam a por um senso de normalidade neste mundo; ao Mário, João, Rita, Cibrão, André e Rui e ao Mendes, Rodrigues, Guilherme, Rita e Cláudia.

A todos, um enorme obrigado.

* * *

Search-ON2: Revitalization of HPC infrastructure of UMinho (NORTE-07-0162-FEDER-000086), co-funded by the North Portugal Regional Operational Programme (ON.2-O Novo Norte), under the National Strategic Reference Framework (NSRF), through the European Regional Development Fund (ERDF).

ABSTRACT

Metabolic models have been widely used in studies of areas that interface with metabolic engineering. Metabolic modeling has proven itself useful in phenotype prediction, having been used to simulate the behaviors and interactions of microbial organisms with *in silico* techniques such as *Flux Balance Analysis (FBA)*. Throughout time, the increase in information available and model quality allowed shifting the attention to community metabolic models, as microbial organisms rarely appear alone in the wild.

In silico metabolic engineering strategies are also used for optimizing microbial phenotypes, to achieve or improve the production of the desired target compound with microbial organisms. Following along with the interest in community models, optimization strategies for community models have also started to appear in recent years.

Metaheuristic algorithms, such as *Evolutionary Algorithms (EAs)*, have been developed in the last decades and deliver problem-independent cost-efficient strategies to solve optimization problems. These algorithms are highly flexible and can adapt to most optimization problems.

In this work, a Python framework was developed to perform *in silico* optimization of the composition of microbial communities given a specified objective with EAs. The objective of this work came to fruition by building community metabolic models and then knocking out organisms during simulations through constraints. This strategy was shown to exhibit advantages over creating a new community metabolic model for every candidate of the *EA*, while being interchangeable regarding the output, as well as perform faster than available options.

The work was successfully validated by testing examples found in the literature and obtaining the desired microbial community replicated in the outcome from the algorithm. The framework also exhibited the possibility to expand this work to multi-objective EAs strategies in the future.

Keywords: Microbial Communities, Community Metabolic Models, Optimization, Evolutionary Algorithms

RESUMO

Os modelos metabólicos têm sido amplamente usados em estudos nas áreas relacionadas com a engenharia metabólica. A modelação metabólica tem provado ser útil para previsão de fenótipos, tendo já sido usada para simular comportamentos e interações entre microrganismos usando técnicas *in silico* como a *Análise de Balanço de Fluxos*. Ao longo do tempo, o aumento da informação disponível e a qualidade dos modelos construídos permitiu mudar o foco de investigação para a construção de modelos metabólicos de comunidades microbianas, visto que os microrganismos raramente surgem sozinhos na natureza.

As estratégias de engenharia metabólica *in silico* são também usadas para otimizar fenótipos de microrganismos de modo a atingir ou melhorar a produção de um composto de interesse produzido por microrganismos. Este facto, aliado ao crescente interesse em modelos de comunidades microbianas, criou a necessidade de se desenhar estratégias de otimização para modelos de comunidades.

Para resolver estes problemas de otimização, nas últimas décadas têm sido utilizados algoritmos metaheuristicos, como os algoritmos evolucionários, sendo estratégias eficientes e independentes do problema. Estes algoritmos são altamente flexíveis e conseguem adaptar-se à maior parte dos problemas de otimização.

Neste trabalho, foi desenvolvida uma plataforma computacional em Python para executar optimização *in silico* da composição de comunidades microbianas segundo objectivos pré-definidos através de *Algoritmos Evolucionários*. O objetivo deste trabalho foi concretizado construindo modelos metabólicos de comunidades e eliminando organismos durante simulações através da adição de restrições. Esta estratégia mostrou apresentar vantagens face a construir um novo modelo de comunidades para cada candidato do algoritmo evolucionário, sendo intercambiável no que diz respeito ao resultado, assim como apresenta resultados mais rápidos do que alternativas existentes.

Este trabalho foi validado com sucesso ao testar exemplos encontrados na literatura e obtendo a comunidade microbiana replicada no resultado do algoritmo. A plataforma abre também a possibilidade de expandir este trabalho com algoritmos evolucionários multi-objetivo no futuro.

Palavras-chave: Comunidades Microbianas, Modelos Metabólicos de Comunidades Microbianas, Otimização, Algoritmos Evolucionários

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

A

AGORA Assembly of Gut Organisms through Reconstruction and Analysis.

B

BRENDA BRaunschweig ENzyme DAtabase.

C

CFBA Community Flux Balance Analysis.

COBRA COnstraint-Based Reconstruction and Analysis.

COMIDA Community Metabolism Design Algorithm.

D

D-OPTCOM dynamic OptCom.

DFBA Dynamic Flux Balance Analysis.

E

EAS Evolutionary Algorithms.

EMO Evolutionary Multi-objective Optimization.

F

FBA Flux Balance Analysis.

FLYCOP FLexible sYnthetic Consortium OPtimization.

FRAMED FRAmework for Metabolic Engineering and Design.

FVA Flux Variability Analysis.

I

ID IDentifier.

K

KEGG Kyoto Encyclopedia of Genes and Genomes.

L

LP Linear Programming.

M

MOMA Minimization Of Metabolic Adjustment.

N

NSGA Non-dominated Sorting Genetic Algorithm.

R

ROOM Regulatory On/Off Minimization.

S

SBML Systems Biology Markup Language.

V

VMH Virtual Metabolic Human.

X

XML eXtensible Markup Language.

1

INTRODUCTION

1.1 CONTEXT AND MOTIVATION

Genome-scale metabolic models have been used for the last decades to study cell behavior under different conditions and to predict the impact that genetic modifications have in the metabolism [1].

Metabolic models, specifically stoichiometric models, are defined by a m x n matrix (S) comprised of m metabolites and n metabolic reactions involving said metabolites, and a vector, v, comprised of n reaction rates [2], with the purpose of mimicking the organism behavior *in silico*. The internal metabolite concentrations are assumed to be in a quasi-steady state, as internal metabolic reactions are relatively faster than growth rates and environmental changes. Thus, all fluxes that lead to the formation or degradation of internal metabolites are assumed to be mass balanced [3]. This system can be represented as:

$$S \cdot v = 0$$

As all possible phenotypes are described as a steady-state flux space, this system can be explored using a *Linear Programming (LP)* approach [4]. Constraint-based modeling approaches, such as *Flux Balance Analysis (FBA)*, introduce lower and upper bounds to the flux rates, thus limiting the solution space of the LP problem, which, given an objective, will then find the respective phenotype solution [5].

Usually metabolic models only consider one organism. However, most of *in vivo* microorganisms live in complex communities [6]. Therefore, studying microbial communities provides better insight into each microorganism behavior in the community as well as the interactions between the members of the community. With that in mind, an increasing number of studies concerning microbial community metabolic models are being published [7]. As the reconstruction of community models is challenging, the focus lies on the high-quality reconstruction of single organism metabolic models to be combined into community models, such as the *Assembly of Gut Organisms through Reconstruction and Analysis (AGORA)* provides [8].

Naturally, to respond to this need, several methods for community metabolic model simulation have appeared as well, such as *Community Flux Balance Analysis (cFBA)* [9] and SteadyCom [10].

Several different strategies have been developed over time to manipulate organisms towards a specific objective. Metabolic engineering focuses on the manipulation of the metabolism of a given organism to improve its properties [11]. With community models, this can be achieved by defining the best community composition for the desired outcome.

Defining the best composition for a community can be seen as an optimization problem. *Evolutionary Algorithms (EAs)*, inspired by natural selection, form a set of optimization methods [12] that have already been used for strain design of single organism models in order to maximize production of industrial interest [13]. However, until now no methods are provided to find the best community composition in a systematic way in order to optimize a given objective of the community.

This work aims to use EAs to find the best community composition of a given microbial community for a defined objective.

1.2 GOALS

The purpose of this work is the development of methods that given a collection of metabolic models, would be capable of using *Evolutionary Algorithms* to optimize the composition of a microbial community for a defined objective.

To achieve that purpose, the goals of this work are:

- Reviewing the state of the art in community metabolic models and related methods and software tools;
- Developing methods to create community metabolic models from single organism metabolic models that allow excluding organisms from *in silico* communities;
- Developing methods using evolutionary algorithms that optimize the composition of a microbial community, for a given purpose;
- Validating the results obtained from the developed methods with other strategies and current available alternatives;
- Testing the performance of the developed methods and validating the optimization with a case study.

1.3 DOCUMENT ORGANIZATION

This document is structured as follows:

- State of the Art
  - Synopsis of the overall subject, which frames this work, Systems Biology.
  - Overview of metabolic models encompassing constraint-based modeling, community metabolic models and model simulations.
  - Detailing metabolic engineering in the context of this work, highlighting EAs.
- Materials and Methods
  - Detailing of the used tools.
  - Overall view of the developed work.
  - Descriptive detailing of all of the components of the developed work.
- Results
  - Performance analysis of the developed work against existing alternative.
  - Validating the optimization methods with a case study.
- Conclusions
  - Takeaways of the dissertation.

2

STATE OF THE ART

2.1 SYSTEMS BIOLOGY

Systems biology is the field of biology that targets the development of a system-level understanding of biological systems, requiring a series of rules and methods that make the connection between the behavior of molecules to the characteristics and functions of a biological system [14].

The system-level approach to biology has been a recurring theme in the scientific community for a long time. One of the first supporters was Norbert Wiener, whose book led, in 1948, to the birth of biological cybernetics [15]. In 1968, based on the concept of homeostasis proposed by Cannon [16], a general systems theory was proposed by Ludwig von Bertalanffy, in an effort to establish a general theory for systems biology [17]. The work was, however, too abstract.

While the pursuit of system-level understanding in biology is not new, only recently, the knowledge framework based on the molecular level has been sufficient to grant the opportunity to study biological systems at the level of genes and proteins. Past attempts did not make the connection between system level description and molecular level knowledge. The essence of system-level understanding of biological systems is obtained from four main points [18]:

i ***Structure Identification:*** The relationships that make up the structure of the system need to be identified. Primarily, regulatory relationships of genes and protein interactions responsible for signal transduction and metabolic pathways and the physical structure of the biological system.

ii ***Behavior Analysis:*** Metabolic, sensitivity and dynamic analysis methods allow for an understanding of the system-level characteristics that underlie specific behaviors.

iii ***Control:*** Minimization of malfunctions by modulating mechanisms that systematically control the state of the biological system required to use the information obtained about its structure and behavior.

iv ***Design:*** Strategies with definite design principles and simulations can be devised to construct systems with desired properties.

One essential characteristic of biological systems is robustness, the ability to preserve certain characteristics in spite of uncertain components or the environment [19]. The characteristics of robust systems can be divided into three categories:

i ***Adaptation:*** Pertains to the capability to endure environmental changes;

ii ***Parameter Insensitivity:*** An indicator of a system's relative sensitivity to kinetic parameters;

iii ***Graceful Degradation:*** After a system takes damage, its functions degrade slowly, rather than catastrophically.

Regarding systems, the robustness is obtained with system control, redundancy of functions, structural stability promoters and, as sub-systems are insulated, modularity [14].

As any full description of a biological system is extremely extensive, systems biology manages to deal with this vast quantity of information by condensing the current knowledge into quantitative and qualitative descriptors and incorporating those into metabolic models [20].

2.1.1 *Metabolic Models*

Mathematical modeling is a very useful tool used in a diverse range of scientific areas to interpret and predict natural events [21]. Given that, modeling any kind of natural phenomena comes down to a simplification, depending on the available data and objectives, it is possible to construct different mathematical models for the same event [22].

The sudden increase in biological experimental data propelled several attempts to construct mathematical models for biological systems. In areas such as biotechnology, there is a specific focus on cellular metabolism, with the objective of exploring the production of certain compounds [23].

2.1.1.1 *Kinetic Models vs Stoichiometric Models*

There are two main different metabolic model strategies, kinetic models, based on microbial or enzyme kinetics as well as stoichiometry, and stoichiometric models, based on invariable characteristics of metabolic networks:

i ***Kinetic:*** Kinetic models introduce reaction fluxes and metabolite concentration changes as a function of time. Requiring more details than stoichiometric models, these usually reach several metabolic reactions and transport processes within few pathways [24].

This type of model includes details from mass action, reaction mechanisms such as Michaelis-Menten reaction, information on types of inhibition and more, and parameters including the catalytic constant, maximal rate of reactions, Michaelis-Menten constant. Kinetic models provide a chance to simulate the perceptible variations of metabolite concentration and flux values over time [25]. Some detailed metabolic models at pathway-scale have been constructed, *e.g.* glycolysis [26] and Entner-Doudoroff pathway [27].

ii ***Stoichiometric:*** Stoichiometric models are capable of covering a biological system genome-wide requiring less detail on individual reactions than kinetic models [28]. These models are used to analyze feasible steady states with just information on the stoichiometry of the reactions present, by assuming extracellular dynamics to be virtually unchanged compared to intracellular ones. Additional parameters such as the direction of reactions, bounds of fluxes and more, improve the accuracy of the prediction of the model. Having less information per reaction allows the creation of large models [25]. Hundreds of stoichiometric models at genome-scale have been manually constructed for different organisms, along with thousands of automatic reconstructions [29, 30].

The main component in constructing a stoichiometric metabolic model is the stoichiometric matrix [2]. Assuming a set of n reactions that establish relationships between an m number of metabolites, we can create a matrix, S, with the size of m x n, comprised of the stoichiometry from the respective reactions, as depicted in Figure 1. Every flux rate from the defined reactions is stored in a vector, v [3]. This is shown in the form of:

$$S \cdot v = 0 \tag{1}$$

where the product between the matrix S and the vector v, corresponding to the net metabolite uptake, is usually zero, due to the assumption that intracellular dynamics are vastly faster than extracellular ones, *i.e.* quasi-steady state [31].

2.1.1.2 *Constraint-Based Modeling*

Constraint-based modeling is based upon the notion that the behavior of a biological system, such as a cell, will be limited by its natural properties. These properties can be classified into 4 distinct categories [33]:

i ***Physicochemical:*** Inviolable hard constraints on cellular functions such as energy, mass and momentum that must be conserved. Cells are densely packed and so its viscosity can be hundreds of times of that of water. This makes the diffusion rates of some molecules, depending on molecular size, limiting and slow [34]. The confinement of

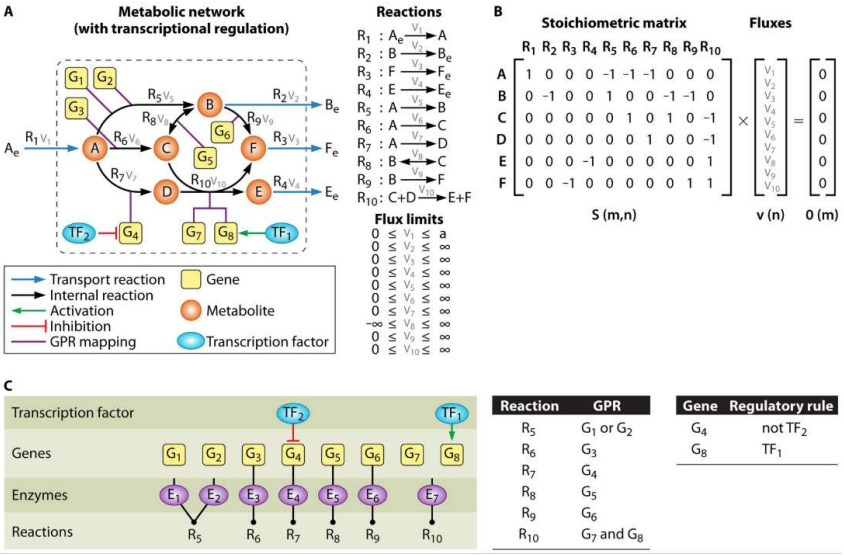


Figure 1: Diagram illustrating how the stoichiometric matrix is formed from the corresponding metabolic network. (A) Sample Network; (B) Stoichiometric Matrix; (C) Gene-Protein-Reaction and Regulation Rules. Source: *Maia, P., Rocha, M. and Rocha, I.* [32]

molecules in semi-permeable membranes demands mechanisms to deal with osmotic pressure [35]. Enzyme-turnover values are usually below 10^4 s^{-1} and maximal reaction rates equal to enzyme concentration multiplied by the turnover [36]. Additionally, reactions proceeding in the forward direction require negative free-energy.

ii ***Topological:*** Fitting molecules inside cells leads to three-dimensional constraints the affecting form and function of the biological system. Physical arrangement of molecules inside the cell follow two clashing needs, to be packed tightly but also to be easy to access [37], *e.g.* bacterial DNA is 1000 times longer than a cell. At pH of 7.6, *E. coli* has around 16 hydrogen ions [38]. These low numbers make the tracking of hydrogen ions crucial, as it turns the discussion around bulk average pH of the intracellular spaces meaningless [39].

iii ***Environmental:*** Dependent on time and condition, environmental constraints include nutrient availability, osmolarity, temperature, pH and presence of electron acceptors. These constraints are fundamental for quantitative analysis of microorganisms. Well documented media environments are necessary to incorporate data from various sources into descriptive and predictive quantitative models. Experiments with a media composition undefined are of limited use in quantitative models [33].

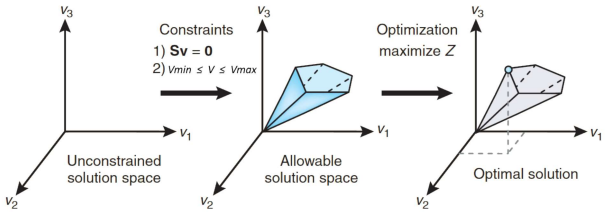


Figure 2: Conceptual basis of constraint-based modeling, showing the solution space constrained by mass balance (1) and lower and upper bounds (2), and the optimal solution found by FBA through the optimization of an objective function. Source: *Orth, J., Thiele, I. and Palsson, B.* [5]

.

iv ***Regulatory:*** Unlike the previous categories, regulatory constraints are self-imposed and subject to evolutionary change. In contrast to the hard constraints, these can be referred as regulatory restraints. These constraints permit the cell to eliminate suboptimal phenotypes and confine to behaviors with better fitness [40].

The constraints can be represented mathematically in metabolic models. These constraints work either as balances, values linked with conserved quantities, such as mass and energy, and events such as osmotic pressure, solvent capacity, or bounds, numerical ranges that limit parameters and variables, such as fluxes and concentrations, of the metabolic model [33].

The most common example of a balance constraint is the conservation of mass. When assuming a quasi-steady state in a metabolic network, every metabolite must be produced and consumed at the same rate. This flux balance is represented trough the abovementioned equation 1 [41].

Bounds are represented trough limit values that constrain individual variables. To every individual flux from the vector v, a lower and upper limit is defined as $v_{min} \leq v \leq v_{max}$. Elementary irreversible reactions have $v_{min} = 0$. Upper limits based on enzyme capacities might be imposed on reactions [40].

Every set of flux rates, vector v, describes a phenotype of the biological system corresponding to the metabolic model. Thus, constraint-based modeling limits the available phenotypes possible for a metabolic model. This allows for simulation methods such as FBA to find the best real phenotype for a defined objective, as depicted in Figure 2.

2.1.1.3 *Community Models*

While microorganisms have a vital role in various industrial processes, it cannot go unnoticed that in the vast majority of cases, microorganisms do not work in isolation, but rather, in microbial communities. Microbial communities are multi-species aggregations where organisms interact with each other [42]. These communities exist in various sizes

from dozens of "species" [43] to hundreds in human microbiomes [44] to tens of thousands per gram of soil [45].

The importance of studying microbial communities over single microorganisms sparkled the emergence of community metabolic models [7]. In terms of constructing community metabolic models, there are two main strategies [46]:

i ***Supra-Organism:*** One of the options is to consider the microbial community as a modeling unit that performs various functions. Through this lens, the microbial community is viewed as the collection of reactions and genes from all species in the community without considering cell boundaries and not as a set of different species. Therefore, it works as a single species prokaryotic model, with one internal compartment and one extracellular compartment. This type of approach is mostly used to study the interactions of the microbial community with the environment [47].

ii ***Population-based:*** In this approach, every species in the model is interpreted as an interacting modeling unit. The internal states and functions are assumed to be unchangeable across organisms from the same species. This involves containing every species in its specific compartment of the model, similarly to the way eukaryotic models assume quasi-steady state between inter- and intra-compartmental reactions [46].

However, when considering population-based models, it is possible for species to grow at a faster rate than others, changing significantly the community structure. Accordingly, considering the basis of constraint-based modeling, quasi-steady state assumptions might not hold [48]. Thus, to predict non-steady state interactions, multi-species dynamic models are required, adding exchange rates for each individual species [47]. These community metabolic model reconstruction strategies can be represented as illustrated in Figure 3. Other community metabolic model reconstruction approaches include graph-based [49] and network expansion strategies [50].

The construction of reliable community metabolic models is a difficult exercise. Even for single species models, genome-scale metabolic network reconstruction is a process that takes a significant time [51]. Community modeling reconstruction is, therefore, even more challenging, given the interactions present between the organisms in the biological system. Instead, it is conventional to focus the construction of community models in the reconstruction of high-quality individual metabolic models. The combination of these models should result in quantitative predictions of the interactions and behaviors of the community [52]. This strategy proves unfruitful if there is a lack of enough data required for individual models. Several projects have focused on the construction of these high-quality metabolic models. Magnúsdóttir *et al.* created AGORA, having generated 773 genome-scale models, claiming it could provide the starting point for generating high-quality curated

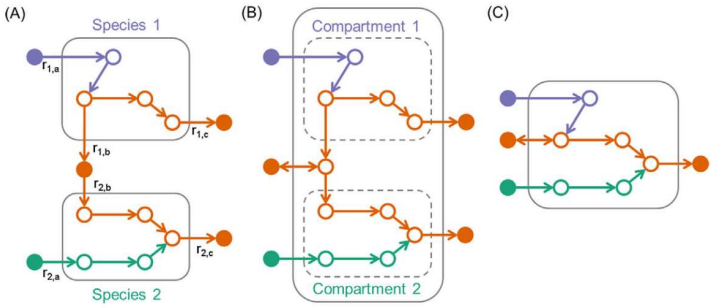


Figure 3: Alternative strategies for the construction of community models: (A) Multi-species dynamic modeling; (B) Population-based modeling; (C) Supra-organism modeling. Colors correspond to metabolic pathways associated with species 1 (purple), species 2 (green) and common for both organisms (orange). Solid circles are extracellular metabolites, open circles are intracellular ones. Solid lines boxes correspond to compartments outside of which quasi-steady state assumptions are not guaranteed, dashed lines boxes are compartments inside of which quasi-steady state assumptions do hold true. In **A** kinetically modeled exchange rates are represented. Source: Henry *et al.* [47].

metabolic reconstructions [8]. AGORA models, however, are constructed from the human microbiome and, therefore, may vary from the species found in soil microbial communities, especially from extreme environments, as conditions such as temperature, pH and salinity may influence the behavior of the species.

2.1.2 *Model Simulation*

The main objective of reconstructing *in silico* constraint-based metabolic models of biological systems is to predict the interactions and behavior of such systems. While the definition of an S matrix defines reactions and metabolites present in a biological system, the constraints applied to said system limit the possible phenotypes. Hence, model simulation strategies work, given a biologically relevant objective function such as biomass growth rate [53], to find the best solution for such an objective. Model simulations, or phenotype predictions, can be achieved through several methods, the most common of which is FBA [3].

FBA attempts to find the optimal solution for an objective by interpreting the issue as an LP problem [54]. Thus, calculating an objective function described by the following equation:

$$\begin{aligned} \text{Maximize} \quad & z = c^T \cdot v \\ \text{Subject to:} \quad & S \cdot v = 0 \\ & v_{j\_min} \leq v_j \leq v_{j\_max} \end{aligned} \tag{2}$$

where c corresponds to a vector responsible for the relation between the fluxes, vector v, and the objective function z. The LP problem is also subject to the constraints defined of stoichiometry consistency and lower and upper bounds for every j metabolite of the vector v [2]. The wide use of FBA has led to several variants of this method that use additional information to surpass limitations of the regular FBA. Regulatory Flux Balance Analysis, developed by Covert *et al.* [55], introduced transcriptional regulatory events as another time-dependent constraint in FBA to study the effects of transcriptional regulation on cellular metabolism. However, these methods require more information than their regular counterparts do, thus fewer models are able to use them.

Flux Variability Analysis (FVA) is able to test the flexibility and robustness of a model by maximizing and minimizing every flux and thus providing a range to each flux rate in which the model still obeys objective function [56], according to the following LP problem.

$$\begin{aligned} \text{Maximize/Minimize} \quad & v_j \\ \text{Subject to:} \quad & S \cdot v = 0 \\ & z_{obj} = c^T \cdot v \\ & v_{j\_min} \leq v_j \leq v_{j\_max} \quad \text{for } j = 1...k \end{aligned} \tag{3}$$

To perform FVA, a first FBA must be performed to define z_{obj}. Then the problem is solved for all k fluxes existent. A significantly more efficient version of FVA, fastFVA, was developed by *Gudmundsson and Thiele* [57]. Using Simplex-type algorithms, instead of starting each LP problem from scratch, it starts from the optimum solution found in the previous LP problem solved, therefore turning the run times from 20 up to 220 times faster than regular FVA, and thus making it feasible to use in larger models.

While the maximal biomass growth rate is assumed in normal conditions, genetic perturbations, such as cases of gene deletion, might override this principle. Different methods have been developed to address this problem. *Minimization Of Metabolic Adjustment (MOMA)* minimizes the sum of squared differences between a reference flux distribution and a mutant one, defining the problem, not as linear, but as quadratic programming [58]. *Regulatory On/Off Minimization (ROOM)* simulates genetic perturbations and then minimizes the amount of significantly changed fluxes compared to the original, with a mixed-integer LP problem [59].

2.1.2.1 *Community Model Simulation*

The development of community models and their implied complexity sparked a need for simulation methods that better fit the needs and specificities of community models.

One of the first steps in this direction is cFBA, a direct translation from single organism FBA to community models, involving only the addition of community-related constraints concerning the preservation of steady state assumptions. The constraint equations added imply that to maintain steady state, one of two scenarios must be true: (1) organism

abundances remain stagnant and the net growth rate is zero, meaning that organisms exchange metabolites while not growing, consistent with periods of dormancy; (2) if all organisms grow at the same rate and exchange rates to the exterior increase as much as the biomass, where the community growth rate does not have to match any of the growth rates of the microorganisms present [9]. cFBA implies the existence of a balanced growth of the microbial organisms involved, therefore, it is apt for studies of stable communities. It predicts flux distribution, growth rates and abundance of each species and exchange fluxes between them and the environment.

In 2012, *Zomorrodi and Maranas* [60] published OptCom, a generalized computational framework to implement cFBA. OptCom generates a multi-level optimization problem. At the inner species-specific level, it creates a separate problem for the biomass growth of each species. The interactions between each species are modeled with constraints in the outer level of the optimization problem, as illustrated in Figure 4.

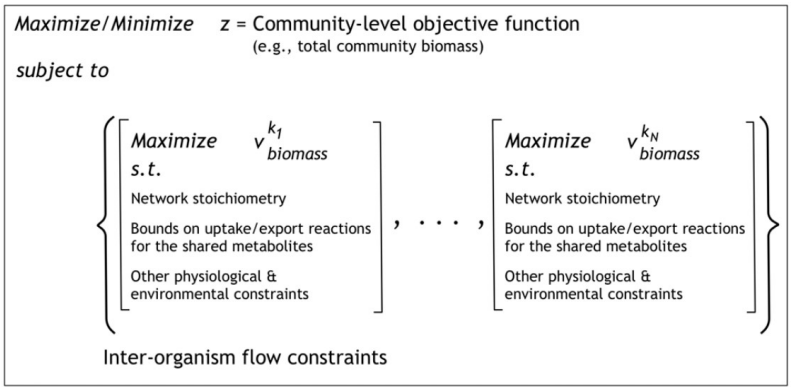


Figure 4: The multi-level structure of the OptCom optimization problem. A separate biomass maximization problem is defined for each species. These inner problems are integrated into the outer level through the inter-organism flow constraint to optimize a community-level objective function. Source: *Zomorrodi, A. and Maranas, C.* [60]

In a following study, *dynamic OptCom (d-OptCom)* was introduced, as an extension to their previous work that used kinetic information to perform dynamic simulations. d-OptCom adds constraints to the outer level of the optimization problem. These constraints concern mass conservation equations for the biomass of each community member, mass conservation equations for shared metabolites with kinetic information, and uptake rates of shared metabolites [61]. This approach was preceded by other dynamic simulation methods such as Dynamic Multi-species Metabolic Modeling [62] and Dynamic Flux Balance Analysis [63].

Recently, SteadyCom was introduced by Chang *et al.* [10], showcasing a simulation method for community models able to predict the flux distributions abiding by the steady-

state requirement. In cFBA, the number of LP problems increases exponentially with the number of organisms present in the community [9]. On the contrary, in SteadyCom, the number of LP problems required depends on the precision of the maximum growth rate and the distance of the solution to the initial estimate, thus providing a clear advantage when simulating larger models. Another advantage is the compatibility with the FVA formulation.

When developing community model simulation methods, it is common to use the sum of the biomass reactions of each species, the community biomass, as the overall objective function [10], which may not be an accurate representation of natural occurrences. The interaction between species might not be beneficial and so the maximization of growth rate would not reflect the behavior of the community.

The development of community model simulation methods is a growing area as new works continue to appear. *FLexible sYnthetic Consortium OPtimization (FLYCOP)*, developed to design synthetic microbial consortia [64], using *Computation of Microbial Ecosystems in Time and Space (COMETS)* [65] to find the best parameters of the microbial community for one or more objective functions. The composition of the community is, however, fixed and the algorithm allows only for a small number of strains. *Community Metabolism Design Algorithm (CoMiDA)*, a graph-based algorithm to find the smallest community composition to yield a target compound [66], considers the metabolites as nodes and reactions as edges through an LP problem, returning the smallest group of species capable of together producing the compound. This method, however, is not optimizing the production of that compound and thus, there can be other, better, alternatives to achieve the same goal, albeit with more organisms.

2.1.3 *Tools*

Systems Biology Markup Language

The *Systems Biology Markup Language (SBML)* is a model definition language based on *eXtensible Markup Language (XML)* [67], a simple machine-readable text-based substrate that has high acceptance in bioinformatics and computational biology [68], as shown in Figure 5. SBML emerged from the necessity of the community to share their models with each other. As several projects were being developed, easy communication between the models generated by each project was necessary, and so SBML served as an intermediary that could be used to facilitate the sharing of information. Despite the initial purpose of it being just a facilitation tool, SBML is the most popular model language [69].

```
<?xml version="1.0" encoding="UTF-8"?>
<sbml xmlns="http://www.sbml.org/sbml/level1"
      level="1" version="2">
  <model name="gene_network_model">
    <listOfUnitDefinitions>
       ...
    </listOfUnitDefinitions>
    <listOfCompartments>
       ...
    </listOfCompartments>
    <listOfSpecies>
       ...
    </listOfSpecies>
    <listOfParameters>
       ...
    </listOfParameters>
    <listOfRules>
       ...
    </listOfRules>
    <listOfReactions>
       ...
    </listOfReactions>
  </model>
</sbml>
```

Figure 5: Depiction of the schematics of an SBML file. Source: Hucka, M. *et al.* [70]

SBML is developed in levels, to maintain the support of earlier versions. SBML Level 2, which is currently the latest fully developed level, includes these features [70]:

i ***Compartment:*** A container where reactions take place.

ii ***Species:*** An entity partaking in a reaction, such as ions or molecules.

iii ***Reaction:*** A description of a transformation, binding or transport process changing one or more species. Includes rate laws describing the way they take place.

iv ***Parameter:*** A nameable quantity. Parameters in SBML can be global or local to certain reactions.

v ***Unit definition:*** A name of a unit used in expressing quantities in the model.

vi ***Rule:*** A mathematical expression added to the equations of the model generated from the reactions.

vii ***Function:*** A mathematical expression used in place of repeated expression throughout the model in rate equations or other formulas.

viii ***Event:*** A set of mathematical expressions used at specified times in the evolution of the system.

Level 3, currently under development, will support more features, including: *Layout*, *Hierarchical Model Composition*, *Arrays*, *Spatial Processes* and *Flux Balance Constraints* [69].

Constraint-Based Modeling Software

One of the most well-known software for constraint-based modeling is the *COnstraint-Based Reconstruction and Analysis (COBRA)* Toolbox [71], an open source software package developed for MATLAB [72] first developed to ease phenotype prediction using the available constraint-based modeling methods at the time. The growth of the community led to more recent versions of the COBRA Toolbox, including methods to simulate and analyze phenotypes and new modeling strategies.

Several other constraint-based modeling software tools have been developed for a different multitude of platforms, including Java [73], R [74], Python [75], C++ [76] and web-based format, as shown in Table 1. Each of these tools was developed with a specific interest in mind and thus their capabilities are not completely overlapping. Constraint-based modeling tools perform a variety of functions including simulation, analysis, design, reconstruction, data integration and visualization.

Table 1: Examples of sofware with contraint based modeling capabilities and respective platform and status [71].

Name	Platform	Interface	Development	Distribution
COBRA Toolbox [71]	MATLAB	Script	open source	git
CellNetAnalyzer [77]	MATLAB	Script/GUI	closed source	zip
OptFlux [78]	Java	Script/GUI	open source	svn
Sybil [79]	R	Script	open source	zip
COBRApy [80]	Python	Script/Narrative	open source	git
CBMPy [81]	Python	Script	open source	zip
FRAMED [82]	Python	Script	open source	zip
SurreyFBA [83]	C++	Script/GUI	open source	zip
KBase [84]	Web-based	Script/Narrative	open source	git

FRAmework for Metabolic Engineering and Design (FRAMED) is a python package capable of analysis and simulation of different metabolic models [82]. While also performing constraint-based modeling strategies, its main focus is to provide support for different modeling approaches: constraint-based models, kinetic models and bioprocess models. It includes several functions for each of the model types supported, including FBA, time-course simulation and *Dynamic Flux Balance Analysis (DFBA).*

2.2 METABOLIC ENGINEERING

The idea of manipulating metabolic pathways to improve cellular processes is old and has been broadly used in the past, from classical breeding and crossing of *Saccharomyces* strains

for beer fermentation to improving penicillin production of *Penicillium chrysogenum* through repeated mutation [85]. The emergence of DNA recombinant technology, however, made it possible to outline more precise, targeted strain improvements. This approach was titled cellular and metabolic engineering. The term was coined in 1991 by *Bailey* [86], describing it as an *"Improvement of cellular activities by manipulation of enzymatic transport and regulatory functions to the cell with the use of recombinant DNA technology."* A more lenient definition was proposed by *Cameron and Tong* as a *"Purposeful modification of intermediary metabolism using DNA techniques"* [87]. Thus, metabolic engineering encompasses the following: *(1)* introducing new pathways in microbial organisms with the objective of producing novel metabolites; *(2)* producing heterologous peptides; *(3)* improving new and existing processes [85].

As with many other engineering subjects, metabolic engineering comprises two main steps: analysis and synthesis [88]. The analysis component of metabolic engineering can be achieved through [85]:

i ***Physiological Studies:*** With the advances in genome characterization, a combination of genetic studies with classic cultivation work comes up as extremely important to identify unknown regulatory compounds and patterns.

ii ***Metabolic Flux Analysis:*** Especially relevant in studies where the objective is to redirect the metabolic production to a certain pathway, metabolic flux analysis outputs valuable information when the quantification of fluxes is estimated at different environmental conditions or with mutant strains.

iii ***Metabolic Control Analysis:*** As several molecular events were identified, such as feedback inhibition, covalent modification and co-operativity in enzymes, identifying the one responsible for flux control often generate disputes, in many cases conflicting. Metabolic Control Analysis reports than all processes influence the flux and that some have higher influence than others.

iv ***Thermodynamic Analysis:*** Information on if a reaction can occur and in which direction can be obtained through the Second Law of Thermodynamics and the Gibbs chemical equilibrium principle. It is also possible to determine a range of concentrations in which each reaction is feasible, thus allowing to identify reactions that compromise the thermodynamic unfeasibility.

v ***Mechanistic Modeling:*** Without needing to assume steady state, dynamic models obtain information from changes in metabolite concentrations over time, providing detail of how individual processes influence the performance of the system.

The synthesis component refers to the processes used to obtain new strains of desired organisms. Several different techniques have been used for this purpose, pertaining to these categories [89]

i ***Construction of Synthetic Metabolic Pathways:*** Many target compounds are not natural to the studied organisms and so strategies to build synthetic pathways are required to attain efficient formation of those targets. With *de novo* pathway design the best candidate enzymes are heterologously or combinatorially introduced to reconstruct a new metabolic pathway [90]. Another approach is enzyme engineering and creation for synthetic pathways. When enzymes for the required pathway are not present, it is possible to develop new enzymes through directed evolution and mutagenesis [91].

ii ***Systems metabolic engineering strategies:*** After assuring the production of the desired target, the objective turns to enhance its production, through the application of systems metabolic engineering strategies. These can be divided into rational-intuitive and systematic-rational-random strategies:

a) *Rational and Intuitive approaches:* Engineering of carbon source utilization has had a lot of focus to enhance its uptake and efficiency to enhance the production of target metabolites [92]. Sometimes the target needs to be excreted, thus minimizing its concentration inside the cell and avoiding inhibitions will maximize the production of the target metabolite from a transporter engineering standpoint [93]. By-product elimination and precursor enrichment are also one of the most used strategies. By removing competing pathways with gene deletion and enhancing the target pathway with gene amplification techniques [94]. Cofactor optimization is often required as they are important to the production of target compounds. Aspects such as the balance of electron-mediating cofactors are controlled with the generation of highly required cofactors [95].

b) *Systematic and rational-random approaches:* These approaches rely on randomness and are driven by a defined rationale, mimicking natural evolution. The introduction of synthetic metabolic pathways, usually from heterologous sources, imbalances the tightly controlled regulation, possibly leading to growth delay or accumulation of metabolites. Optimization and modulation of the metabolic pathways strategies are used to control the expression of enzymes to desired values [96]. Finding the necessary enzyme may not be enough, thus enzyme evolution for activity optimization is crucial in improving the synthesis rate of a target compound [97]. Using the natural selection dogma, metabolite evolution obtains desired phenotypes, which are too complex to enhance using rational strategies [98]. Adaptive evolution with resequencing and re-engineering mimics

natural evolution by exposing cells to rationally devised environmental conditions, which are then sequenced to identify responsible genes [99].

In accordance with the *in silico* use of metabolic models, most of these strategies have *in silico* counterparts and/or complements.

2.2.1 *In silico Metabolic Engineering*

When translated to the *in silico* frame, metabolic engineering focuses on the prediction of modifications to the wild type strain of an organism to design a new strain that will improve the yield of a product of interest. These modifications include gene deletion, gene over or under-expression, heterologous insertion and cofactor binding specificity [32].

Gene deletion is the suppression of metabolic functions through the inactivation of specific genes. This can be obtained by forcing the inactivation of some fluxes when simulating a model and evaluating the effects of that suppression [100]. When inactivating a specific gene can be fatal to the organism, it is possible to use gene over or under expression. Defining bounds close to their theoretical limits, when performing a simulation, allows achieving the desired outcomes without compromising the organism [101]. Heterologous insertion consists of adding new pathways or genes to the model, not native to the organism. It is performed by searching through databases for reactions that could be inserted in the model that fulfil the desired outcome [102]. Cofactor binding specificity can be modulated by simulating changes to the specificity between some reactions and evaluating the predicted phenotypes [103].

In an effort to optimize strains through gene deletion, OptKnock was introduced in 2003 by Burgard *et al.* [100], as a bi-level optimization method that couples drain reactions needed for biomass growth, such as energy and carbon sources, with the production of the target compound as depicted in Figure X. Thus making the target compound a mandatory by-product of biomass growth. OptKnock outputs which reactions should be removed from the metabolic network, which is achieved by deleting the responsible genes.

OptStrain, an extension of OptKnock, was later developed as a method to include nonnative functionalities through the addition of heterologous enzymes [104]. Pulling information from sources such as *BRaunschweig ENzyme DAtabase (BRENDA)* [105], *Kyoto Encyclopedia of Genes and Genomes (KEGG)* [106] and MetaCyc [107], OptStrain is loaded with a reaction database used to find reactions that can be added to a metabolic model.

Likewise, OptReg was introduced a few years later to include gene over or under-expression in strain optimization problems. OptReg works similarly to OptKnock with some additional constraints. Apart from knocked out genes, it introduces upregulated genes and downregulated genes as variables to create new strains [108].

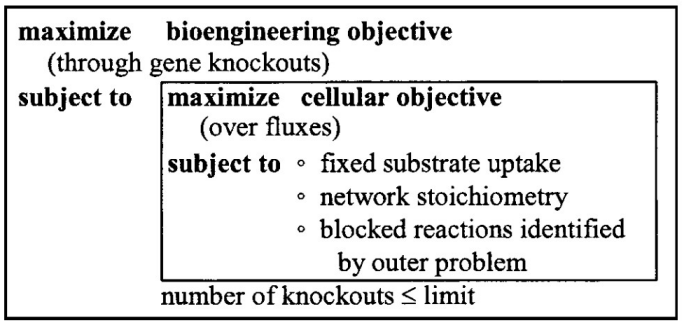


Figure 6: Bi-level optimization framework of OptKnock. The inner problem focuses on a cellular objective such as biomass growth, while the outer layer maximizes the product of interest by restricting the reactions available to the inner layer. Source: Burgard *et al.* [100]

These works brought forth a plethora of similar computational methods to optimize strain design. However, due to the elemental complexity of the computational complexity of these optimization methods, despite outputting an exact solution, they often are limited to a small maximum number of alterations [109].

The limitations of these previous methods led to the utilization of metaheuristic methods. Metaheuristic methods present a less computationally expensive alternative. Despite not being able to guarantee the optimal solution for a problem, they allow for optimization across a larger subset of phenotypes. Some well-known metaheuristic optimization methods include EAs, Simulated Annealing and Tabu Search [110].

2.3 METAHEURISTIC ALGORITHMS

Metaheuristic is a term established by Glover [111] to define higher-level problem-independent framework designed to select heuristics that provide solutions to optimization problems. Metaheuristic algorithms are reliant on stochastic processes and as such are not guaranteed to find the best solution. Regardless, these algorithms are able to efficiently find optimal solutions with less effort than most alternatives [112]. Most metaheuristic algorithms are inspired by natural phenomena, leading to a multitude of different algorithms. The easiness with which these are generated has, however, gathered some criticism [113].

The diversity of existing metaheuristic methods can be classified within the following [114]:

i ***Nature-inspired vs Non-nature-inspired:*** Methods can be inspired by natural events, such as the Genetic Algorithm [115] or not, *e.g.* Tabu Search [111]. This classification is merely indicative and, sometimes, ambiguous.

ii ***Population-based vs single point approach:*** Related to the number of solutions the algorithms works with at the same time. Trajectory methods work with only one solution with local search strategies. Algorithms working with a population of solutions, on the other hand, showcase the evolution of the solutions through the search space.

iii ***Dynamic vs static objective function:*** Some algorithms keep the objective function intact through the whole process, while others will modify it by incorporating information collected throughout the search to escape local minima.

iv ***One vs various neighborhood structures:*** While most metaheuristics work with a single pool of solutions, it is possible for algorithms to jump between neighborhoods, diversifying the search if needed.

v ***Memory usage vs memory-less methods:*** Whether or not the algorithm makes use of its search history. Algorithms can use only the current state of the search, use short-term memory, such as the recently performed moves or long-term memory by accumulating parameters related to the search.

2.3.1 *Evolutionary Algorithms*

Inspired by the evolutionary theory [116], EAs constitute a family of population-based optimization algorithms [12]. Search methods based on natural selection can be traced as far back as 1954 [117].

In 1975, John Holland introduced the genetic algorithm [115], being the first to introduce crossover and recombination techniques to diversify the population of solutions. Around the same time, Fogel *et al.* [118], with the intent to simulate evolution to study artificial intelligence, created a programming technique that featured random mutations to the solutions in each generation. This combination of crossover, recombination and mutations events in the generation of new populations of solutions is the staple in EAs [112].

While EAs are known for their huge flexibility, it is possible to identify five main components [12]:

i *Evaluation function, rating solutions in terms of fitness:* The fitness function is the most flexible component of an EA. Every member of a population is evaluated by it and attributed a score value. If a goal value set beforehand is reached, the algorithm is signaled to stop, as the objective has been fulfilled.

ii *Genetic representation of solutions:* The representation of each solution is, therefore, highly dependent on the evaluating function, as it will consist of the parameters required by evaluating function. Usually, every solution is represented by an array with a set number of elements, which can be represented as any kind of data type,

such as real, integer, boolean or string, depending on the nature of the parameter. Therefore, every element of the solution acts as a chromosome.

iii *Creating a population of solutions:* Every iteration of the evolutionary algorithm will need a different population. The first population is usually set randomly from the possible values every element can take. After going through the evaluating function once, every member of the population will have an associated score, which will influence how the next generation is produced. Creating a new population can be achieved by a potentially infinite number of ways. Using the associated scores, the elements of the best scoring solutions are used to generate solutions for the next generations. New generations can be entirely new or keep a number of members from previous iterations.

iv *Genetic operators that change the composition of children during reproduction:* With the goal of imitating their *in vivo* counterparts, the generation of new individuals is done by choosing two solutions from the current generation and performing a combination of the following strategies, as depicted below: *(1)* recombination: where every element of a new solution is chosen randomly from the parent elements; *(2)* crossover: where one or more points in the sequence are randomly set, marking the intervals of elements given by each parent; *(3)* mutation: singular stochastic alterations to one of the elements of the offspring.

$$\begin{aligned}
\text{Parents:} \quad & P_1[0,0,0,0,0],\ P_2[1,1,1,1,1] \\
\\
\text{Recombination:} \quad & C_1[1,0,1,1,0],\ C_2[0,0,0,1,0] \\
\text{Crossover:} \quad & C_1[1,1,1,0,0] \ \&\ C_2[0,0,0,1,1], \\
& C_3[0,1,1,1,0] \ \&\ C_4[1,0,0,0,1] \\
\text{Mutation:} \quad & C_1[0,0,1,0,0] \ \rightarrow\ C_1^*[1,0,1,0,1]
\end{aligned}$$

v *Values for the parameters:* Finally, the values for the sizes of the population and of the solutions, the stopping criterion, which can be a goal score value and/or a number of iterations, the probabilities of some of the genetic operators acting or not and any other required value of the evolutionary algorithm.

EAs are widely used as they are extremely flexible, allowing them to perform general optimization problems using any representation, fitness function, selection and variation mechanisms, as illustrated in Figure 7, providing them with a tailored advantage over other optimization methods [119].

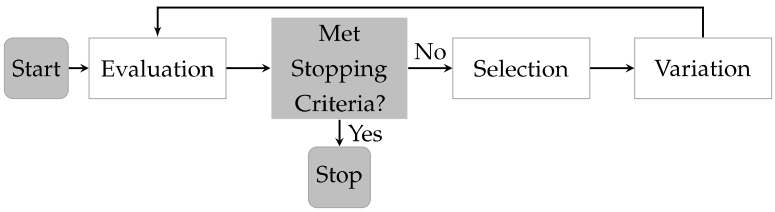


Figure 7: Conceptual basis of evolutionary algorithms.

Strain optimization software has been developed using EAs. For instance, OptGene performs optimization through gene deletion, where each gene is an element of every solution, represented by an array of Boolean elements [120]. The population is therefore defined as a set of different strains of the same organism, in which simulating methods such as FBA or MOMA are used as a fitness evaluation function, returning the objective function score as the associated score for each solution. Variation is performed by simulating crossover of genes between the individuals.

2.3.1.1 *Multi-objective Evolutionary Algorithms*

Evolutionary Multi-objective Optimization (EMO) was judged one of the fastest growing research areas amongst every computational intelligence topic during the World Congress of Computational Intelligence (WCCI) in Vancouver, 2006 [121] which is reflected in the number of multi-objective EAs suggested over the years [122, 123, 124].

One of the main objectives of multi-objective optimization is to find Pareto-optimal solutions, also known as non-dominated, non-inferior or efficient solutions. A solution is called Pareto-optimal, when in a minimization problem, if for all of the values on the solution vector there is not another vector in the solution space in which each value is lower or equal than those of the solution and at least one of the values is lower [125]. Put simply, a solution is Pareto-optimal if there is no other solution possible that perform better in one of the parameters without compromising the others.

The main point of attractiveness of this field is the ability of this sort of algorithms to find multiple Pareto-optimal solutions in a single run, in opposition to other multi-objective strategies. These will generate multiple solutions, which, when plotted will form the recognizable, Pareto-front, as demonstrated in Figure 8

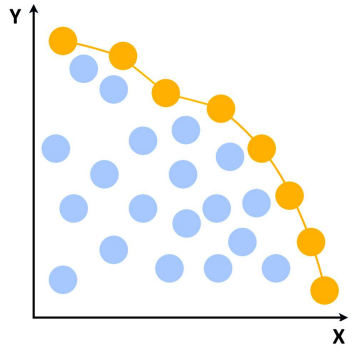


Figure 8: Given a set of solutions with two parameters each, X and Y, represented by circles, in a problem of maximization, the orange circles represent the Pareto front: the set of Pareto-optimal solutions where no solution outperforms the others. Image adapted from wikipedia contributors [126].

Since, however, there can be multiple optimal solutions, no solution can be stated as better than the rest. Therefore, selecting the best solution for the desired objective involves external information and each option will present a trade-off situation to be considered [125].

3

MATERIALS AND METHODS

An evolutionary algorithm was designed to achieve the proposed goals, with community metabolic models in mind. This algorithm creates a community model from an existing list of single organism metabolic models.

Afterwards, it creates an initial population of candidates (described in section 3.1), with which it starts by evaluating every candidate and, subsequently, generating new populations of candidates based on the evaluation outcome.

This is repeated until a predefined number of evaluations is performed or a specific evaluation outcome is hit, as shown in Figure 9 and detailed throughout this chapter.

3.1 TERMINOLOGY

Due to the high plasticity of evolutionary algorithms, it is necessary to detail the terminology used as it is specific to the developed work. In this light, considering a set of 10 metabolic models, labeled 0 to 9, representing 10 different organisms, the following terminology is used:

- ***Candidate:*** A candidate is portrayed as a representation of the composition of a microbial community. Candidates describe the composition of a community either through binary representation (a list of ones and zeros, where the indexes of the ones reflect which organisms are present) or through integer representation (a list of integers corresponding to the models present) as depicted in Figure 10 a).
- ***Population:*** A population is a set of candidates, that is, a collection of representations of possible compositions of microbial communities, as depicted in Figure 10 b); not to be confused with microbial populations.
- ***Generation:*** A generation is one iteration of the evolutionary algorithm, characterized by a population of candidates. After the population is evaluated and its candidates used to generate new ones, a new generation starts, as depicted in Figure 10 c); not to be confused with the reproduction based concept of generation.

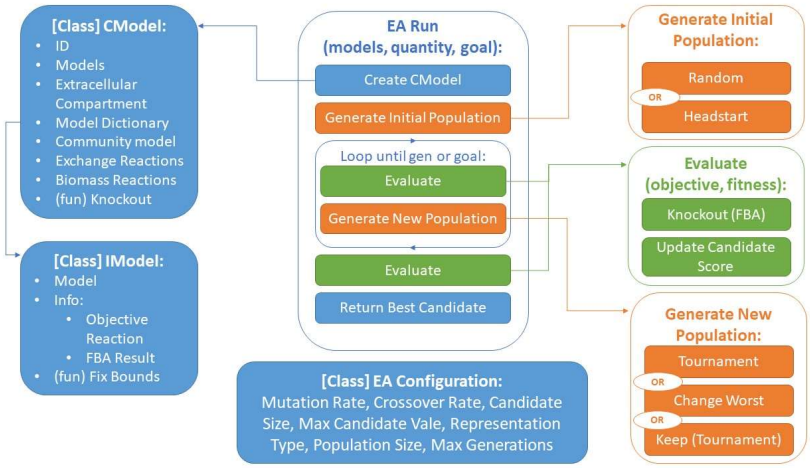


Figure 9: General work flow of the evolutionary algorithm designed to perform optimization of microbial community composition. The algorithm, creates a community model, generates candidates, evaluates them, and continues generating new candidates until it outputs the best performing one. The blue sections pertain to classes developed for the algorithm and its attributes, the orange to candidate generating sections, and the green to candidate evaluating sections.

3.2 HARDWARE AND SOFTWARE

The present work was developed and tested in a computer with an Intel Core i7-500U processor (2.40GHz, 4MB Cache) and 8GB of RAM on a Windows 10 operating system.

The speed performance comparisons were performed in *SErvices and Advanced Research Computing with HTC/HPC clusters (SeARCH)*, a computing infrastructure, from which only 5 out of 54 nodes were used: three with 1024GB of space and supporting Gigabit Ethernet network, one of which with a E5-2683v4 dual CPU, 2.10GHz, 32 cores and 256GB of RAM and two with E5-2660v4 dual CPUs, 2.00GHz, 28 cores and 128GB of RAM; and two with E5-2650 dual CPU, 2.00GHz, 16 cores and 64GB of RAM, supporting Gigabit Ethernet and Myrinet 16Gbps networks. *SeARCH* runs Linux (CentOS 6.3 at the node level and Rocks 6.1 (Emerald Boa) to manage the cluster).

The programming language chosen to develop this work was Python 3.7. All python packages were installed through Anaconda Navigator 1.9.7 [127] and the code was written with PyCharm 2019.2.3 [128].

Two Python packages were essential to the development of this project, namely FRAMED and Inspyred. The former was used to handle the SBML metabolic models within the Python framework. The following classes and functions were imported from the FRAMED package:

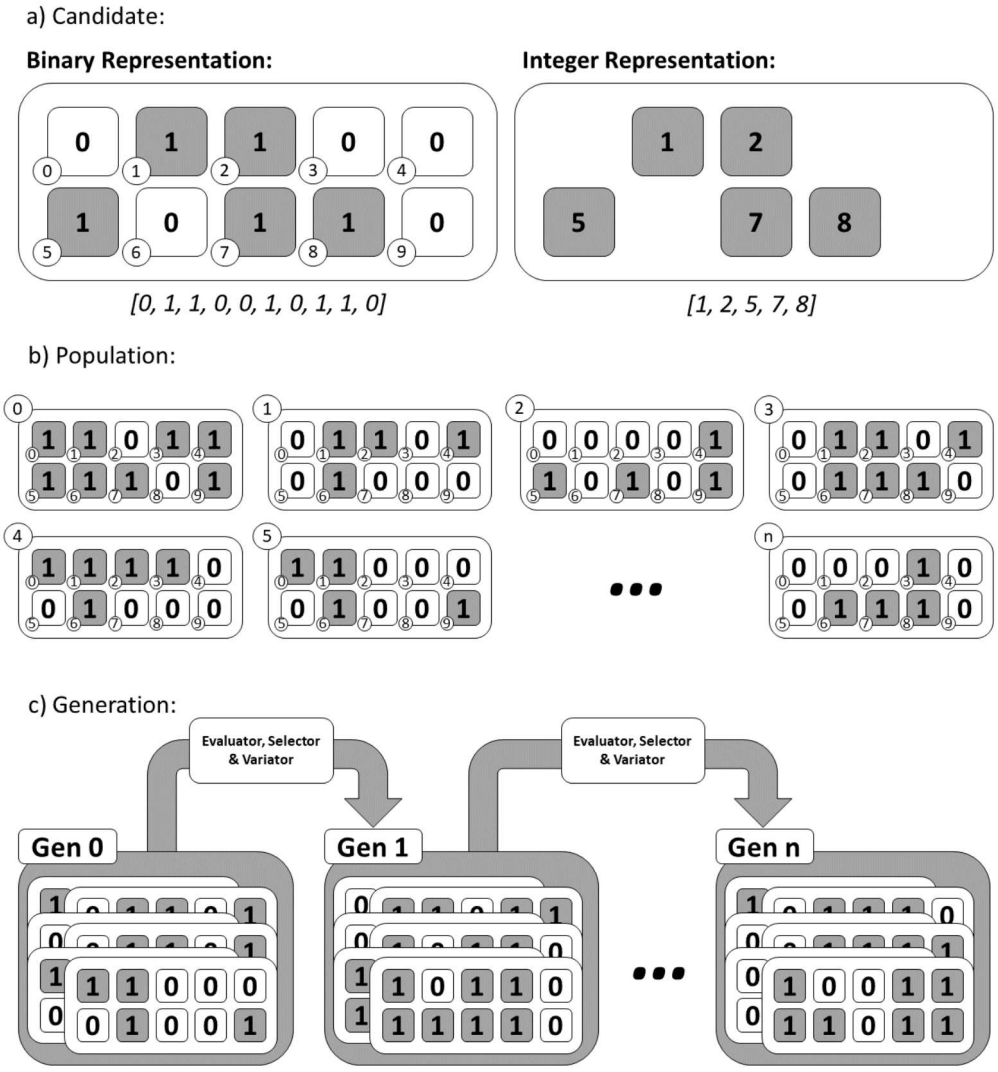


Figure 10: Visual depictions of several concepts used throughout the developed work, considering a pool of 10 models labeled 0 through 9. a) Candidate: A candidate represents a possible composition of a microbial community. It is depicted the composition in both binary and integer representation for a composition with models 1, 2, 5, 7 and 8 present. The filled squares represent which models are present in the candidate. The associated numbered circles indicate the positional index of the model in the binary representation. b) Population: A population refers to a series of candidates. Here depicted is a set of n random representations. c) Generation: A generation is one iteration of the evolutionary algorithm, characterized by its population. After a generation undergoes evaluation, selection and variation processes, a new generation starts, with a new population.

- The *load_model* function loads the model to the Python console, creating a FRAMED Model object. This object creates a metabolic model in python by organizing every metabolite, reaction and compartment in its dedicated object. Creating a Model object requires the file name of the model to be loaded and the *flavor* parameter, indicating the format of the model (*cobra*, *cobra:other*, *seed*, *bigg* or *fbc2*).

- The *FBA* functions performs a FBA on a Model object, returning a CPLEX [129] Solution object, containing the status of said FBA and the values of the calculated fluxes. Running the *FBA* function requires a FRAMED Model object. Additionally, the function can take parameters such as *constraints* to limit the value of certain fluxes and *objective* to define a non default objective for the simulation.

- The *Community* and *Environment* classes create the community model from existing FRAMED Model objects. The *Community* takes, as parameters, an *IDentifier (ID)*, a list of FRAMED Model objects and the ID of the external compartment (which must me the same across all pertaining individual models), and creates the community model as a FRAMED Model object. The *Environment* class is then called to generate a complete growth medium for the community model.

The latter (Inspyred) package was used to perform evolutionary computation, in order to compare an existing alternative with the algorithm developed in this work. The main class imported from this package is the Evolutionary Computation framework, *ec*. Although Inspyred allows users to create their own framework, it also provides several preset evolutionary computation strategies, out of which the following were used:

- ***GA:*** Represents a canonical genetic algorithm; served as comparison to the developed work.

- ***emo.NSGA2:*** Represents the *Non-dominated Sorting Genetic Algorithm (NSGA)*; it was used to perform multi-objective optimization of the microbial composition.

The *evolve* function of the chosen algorithm was called to use these algorithms. The following parameters were used:

- ***pop_size:*** to define the size of each population of candidates;

- ***maximize:*** to decide if the optimization maximizes or minimizes;

- ***bounder:*** to force the values of each candidate inside the desired intervals; and,

- ***max_generations:*** to set the maximum number of generations the algorithm was to endure.

Apart from these, several attributes had to be designed or imported, namely:

- ***Generators:*** These functions generate a single candidate representation, which Inspyred utilizes to generate new candidates for populations of each generation.
- ***Evaluators:*** Evaluator functions take a list of candidate representations and return a list of fitness values, respective for each candidate.
- ***Observers:*** Have a mainly indicative purpose as its only purpose is to print to the console the latest generation's candidates and respective fitness values.
- ***Variators:*** Variators are responsible for creating new candidates from existing ones, usually chosen based on high fitness performance. These can take multiple existing candidates and combining them into new ones, *e.g.* crossing-over, or altering a single existing candidate into a different representation, *e.g.* mutation.
- ***Terminators:*** Functions that determine how or when the evolutionary computation terminates. These can stop the algorithm on a predefined setting, such as time passed, generations evaluated or having hit a fitness goal.

Every evolutionary computation framework from Inspyred has an *evolve* method which needs to be called to perform an evolutionary run. *evolve* takes the generator and the evaluator function, while other parameters are to be attributed to the evolutionary computation framework itself; it is also where the size of the population, the number of generations, the bounder and whether the optimization is a maximization or minimization are defined.

3.3 METABOLIC MODELS

A large number of similarly annotated individual metabolic models are required to perform community composition optimization. In that light, it was decided to use the AGORA models provided by *Virtual Metabolic Human (VMH)* [130]. This collection encompasses 818 different models from the human gut microbiome. The *'AGORA 1.02 - Average European diet constraints'* [131] version of the models, made available in 10/02/2018, was downloaded from the VMH website (https://www.vmh.life) in 12/02/2019.

A second pool of models was selected to perform the validation. These models were collected from several sources, as shown in Table 2.

Converting SBML Format

The SBML version of the AGORA models was incompatible with the loading options provided by FRAMED. To address this issue, every single model was loaded in Python and written in a compatible format using the COBRApy [80] package.

Table 2: List of models used for validation along with organism strain and source.

Model iD	Organism	Reference
iAF1260	*Escherichia coli* str. K-12 substr. MG1655	Feist et al. [132]
iAO358	*Lactococcus lactis* subsp. lactis Il1403	Oliveira et al. [133]
iBif452	*Bifidobacterium adolescentis* L2-32	El-Semman et al. [134]
iBT721_v2	*Lactobacillus plantarum* WCFS1	Goffin et al. [135]
iCR744	*Rhodoferax ferrireducens* T118 (DSM 15236)	Risso et al. [136]
iFap484	*Faecalibacterium prausnitzii* A2-165	El-Semman et al. [134]
iJB785	*Synechococcus elongatus* PCC7942	Broddrick et al. [137]
iJL432	*Clostridium acetobutylicum* ATCC 823	Lee et al. [138]
iJL846	*Lactobacillus casei* LC2W	Xu et al. [139]
iJSPcarbinolicus	*Pelobacter carbinolicus* DSM 2380	Sun et al. [140]
iJSPpropionicus	*Pelobacter propionicus* DSM 2379	Sun et al. [140]
iLB1027	*Phaeodactylum tricornutum*	Levering et al. [141]
iMF721	*Pseudoalteromonas haloplanktis* TAC125	Fondi et al. [142]
iMM904	*Saccharomyces cerevisiae*	Mo et al. [143]
iMZ1055	*Bacillus megaterium* WSH-002	Zou et al. [144]
iNV706	*Enterococcus faecalis* V583	Veith et al. [145]
iRM588	*Geobacter sulfurreducens* PCA (ATCC 51573)	Mahadevan et al. [146]
iRP911	*Methylobacterium extorquens* AM1	Peyraud et al. [147]
iRR1083	*Salmonella typhimurium* LT2	Raghunathan et al. [148]
iSB1139	*Pseudomonas fluorescens* SBW25	Borgos et al. [149]
iSO783	*Shewanella oneidensis*	Pinchuk et al. [150]
iSR432	*Ruminiclostridium thermocellum* ATCC 27405	Roberts et al. [151]
iWZ663	*Ketogulonicigenium vulgare* WSH-001	Zou et al. [152]
iYS432	*Corynebacterium glutamicum*	Shinfuku et al. [152]

Filtering

An initial FBA was performed for each of the AGORA metabolic model gathered, with the maximization of the default biomass reaction as the objective function. However, only some of the models registered unique objective function values, due to the propagating way in which AGORA works. The models whose objective function value is repeated may not be as accurate, and as such, were filtered out. Out of the 818 models that underwent simulation, there were only 314 different biomass flux values; out of which only 277 values registered a single model associated. These 277 were identified to be used whenever it was required to pick models at random.

3.4 Developed Work

The code written for the dissertation is organized in three modules: *Classes*, *Background* and *Evolutionary Algorithm*.

- The *Classes* represent the different types of Python objects created for this project.
- The *Background* is composed of auxiliary functions that create populations and mutate candidates, among other things.
- The *Evolutionary Algorithm* section is composed of the evolutionary algorithm itself, along with the evaluating functions.

3.4.1 *Classes*

Four different interacting classes were created in Python as gears of the developed Evolutionary Algorithm: *IModel*, *CModel*, *EAConfig* and *Candidate*.

IModel

The most basic of the created classes is the *Individual Model (IModel)*. Its purpose is to incorporate individual metabolic models and some of their properties into the community model further down the line. The *IModel* is initiated by providing a FRAMED Model object. It starts by turning any bound set to *None* to a numerical value as to prevent infeasibility issues. Then, it saves a dictionary with the name of the default objective reaction, which should be set to the biomass reaction, and its value when run in a FBA, schematized in Algorithm 1.

Algorithm 1 IModel Constructor - *IModel()*

Input: $FRAMED\ Model$

1: $self.model \leftarrow FRAMED\ Model$
2: Fix *None* bounds
3: $self.info(dic) \leftarrow \{objective\ reaction\ name,\ objective\ reaction\ flux\ value\}$

CModel

The *Community Model (CModel)*, alike the *IModel*, incorporates the community model itself into the code. The *CModel* requires the same three parameters required to call the FRAMED *Community* function: a community ID, a list of FRAMED Model objects and the extracellular compartment ID. Five attributes, which will be necessary in the latter parts of the project, are created, as detailed in Algorithm 2:

- A dictionary with the ID of each individual model as key and the respective *IModel* objects as values.
- The FRAMED Model object of the community model.

- A dictionary with the objective reactions of the community metabolic model.
- A dictionary with the reactions required to knockout an organism, separated by organism. These reactions are stored as constraints to be used by the FRAMED FBA function, detailed in Algorithm 3. These reactions were dubbed *pex reactions*, for pseudo exchange reactions, as they closely match the exchange reactions of the individual metabolic models.
- Lastly, a dictionary with the biomass reactions of each of the individual organisms as keys and the respective models values.

Algorithm 2 CModel Constructor - *CModel()*

Input: $community\_id(str)$, $FRAMED\ Models(list)$, $extracellular\_compartment\_id(str)$

1: $self.community\_id \leftarrow community\_id$
2: $self.models \leftarrow FRAMED\ Models$
3: $self.extracellular\_compartment\_id \leftarrow extracellular\_compartment\_id$
4: $self.model\_dic(dic) \leftarrow \{model\_IDs : IModel\ objects\}$
5: $self.cmodel \leftarrow Community(community\_id, models, extracellular\_compartment\_id)$
6: $self.objectives(dic) \leftarrow \{objectivereactions : 1\}$
7: $self.pex\_cons(dic) \leftarrow \{compartment : pex\ reactions\}$
8: $self.biomass\_reactions(dic) \leftarrow \{biomass\ reactions : model\_IDs\}$

Additionally, the following methods were developed for the *CModel* class:

- ***fba():*** a function that runs the FRAMED FBA function. If a biomass reaction is not an objective or a constraint, its flux will have a lower bound of 10% of the value stored in *IModel.info*.
- ***knockout():*** takes a list of model IDs as a parameter and performs a FBA while knocking out the input organisms, as described in Algorithm 4.

The *EA Configurations (EAConfig)* class coordinates most of the parameters for the functions in the *Background* section, which will be detailed further down in this document. The class includes the following attributes:

- ***mutation rate,*** probability of occurring mutations;
- ***crossover rate,*** probability of occurring crossover;
- ***candidate size,*** the size of the candidate representation;
- ***maximum candidate value,*** the largest value an item of the candidate representation can take;

Algorithm 3 Aggregates the pex reactions of a community model in the format of constraints - *get_pex_constraints()*

```
exch(list) ← [exchange reactions]
ext_comp(str) ← external compartment
exch_metas ← [ ]
for reaction in exch do
    Concatenate substrates and products to exch_metas
end for
pex_reacs ← [ ]
for metabolite in exch_metas do
    Concatenate metabolite reactions to pex_reacs
end for
pex_per_comp(dic) ← { }
for pex reaction in pex_reacs do
    comps ← pex reaction compartments
    for compartment in comps do
        if compartment not ext_comp then
            if compartment not in pex_per_comp then
                pex_per_comp[comp] ← [pex reaction]
            else if compartment in pex_per_comp then
                Add pex reaction to pex_per_comp[comp]
            end if
        end if
    end for
end for
for model name in self.model_dic do
    for reaction in model reactions do
        if all compartment names end with the model name then
            if reaction not in pex_per_comp[model compartment] then
                Add reaction to pex_per_comp[model compartment]
            end if
        end if
    end for
end for
pex_constraints ← { }
for compartment in pex_per_comp do
    pex_constraints[compartment] ← pex_per_comp[compartment] in constraint format
end for
```

Output: *pex_constraints*

- ***representation type,*** the type chosen for the representation, either binary or integer;
- ***population size,*** the size of the population of candidates;
- ***tournament size,*** the size of the tournament when creating a new population from an existing one. Set at $2 + 5\%$ of the *population size*;
- ***maximum generation,*** the maximum number of generations the algorithm runs for before forcibly stopping;
- ***scoredic,*** a dictionary that saves the scores of each candidate;
- ***val_dic,*** a dictionary that saves the values of all reactions for each candidate;
- ***fit_dic,*** a dictionary that saves the values for the reactions to be used as fitness for each candidate.

Algorithm 4 Performs Knockouts when running FBA - *knockout()*

Input: $list\_of\_model\_ids(list)$, $objective\_list(list)$, $constraints(dic)$

if no *objective_list* is given **then**
    $objective\_list(list) \leftarrow [\,]$
end if
if no *constraints* is given **then**
    $constraints(dic) \leftarrow \{\,\}$
end if
for model **in** *list_of_model_ids* **do**
    Update *constraints* with $self.pex\_cons$[model compartment]
    $biom\_reac \leftarrow self.model\_dic$[model].$info$[reaction name]
    Set *biom_reac* objective to zero.
end for
$self.fba(objective\_list, constraints)$

Output: *res*

Lastly, the *Candidate* class; an instance of this class is used for each candidate of a population. It needs, as parameter, a representation, in the form of a list, which is handled by the population generating functions from the *Background* section. Its attributes consist of the *representation*, the *score*, returned by the FBA function, the flux *values* of every reaction simulated in the FBA and the list of flux values for the fitness relevant reactions, *fit_list*. It has two methods:

- ***set_cand_values():*** which sets the values for the candidate attributes, calculated during the evaluation process;

- ***knockout():*** which sets the values from the *EAConfig* dictionaries when the candidate has been evaluated before.

3.4.2 *Background*

Several auxiliary functions support the evolutionary algorithm. These range in purpose and are applied in different sections of the algorithm. Most of these methods have parameters that are set by *EAConfig*. They can be described as:

- ***EAConfig auxiliary:*** A function was created to change the parameters of the EA configurations, instead of changing the default code. A function to reset the dictionaries in *EAConfig*, to run the EA more than once in the same script, is also available.
- ***Representations:*** Functions that create representations, both in binary and integer form, depending on the selected options, were developed. Additionally, some representation converters were needed, namely, a binary to integer converter and a function that returns the inverse representation of a given integer one.
- ***Mutation and Crossover:*** These methods include bit mutations, changing one element of the representation; uniform crossover, which takes two candidates, and randomly attributes each element of the two candidates to two new candidates; and one point crossover, which takes two candidates and joins the beginning of one and the end of the other, thus creating two new candidates.
- ***Populations:*** These include creating starter populations, either completely random or partially random, by forcing specific elements to be included in every representation. There are different strategies to create the subsequent populations. Creating a new population from an existing one by tournament will repeatedly take a certain amount of the candidates of the original population, choose the two best scoring candidates and perform or not crossing over and mutation. There is also a way to create a new population by changing the least performing elements of each candidate for new, random, elements. This is done by comparing the fluxes of the respective precursor reactions of the reaction being used as fitness. Lastly, there is an option that acts exactly as the tournament option albeit forcing specific elements to be present. All of the above mentioned population mechanics are pictured in Figure 11.
- ***Other:*** Some other functions that serve more arbitrary purposes include a sample size checker, to prevent errors downstream, a function to get the precursor reactions of a reaction, and another to get the reactions to be used as fitness from a given exchange reaction.

3.4.3 *Evolutionary Algorithm*

The evolutionary algorithm is supported by the previously mentioned methods. The evolutionary algorithm function, *ea_run*, detailed in Algorithm 5, is aided by the evaluating function (*evaluate*) and the population generating functions. The *ea_run* takes the following parameters:

- ***option list,*** this includes the options to generate populations;
- ***list of models,*** a list of FRAMED Model objects;
- ***cmodel,*** an already created FRAMED community model;
- ***objective list,*** a list of reaction IDs to be used as objective function of the FBA inside the evaluating function;
- ***quantity,*** this defines the number of organisms present in the candidates. If left on default, the number of organisms per candidate is random;
- ***fitness,*** this takes an exchange reaction ID to be used as fitness by the evaluating function;
- ***goal,*** a numeric score to stop the algorithm if met by any candidate during evaluation.

The main method inside the evolutionary algorithm functions is the evaluating function. All parameters utilized are inherited through *ea_run*, either provided by the used or created or modified by it. The function will, for each of the candidates in the population, if no fitness is yet stored, simulate with a FBA and store the designated values, as described in Algorithm 6

3.4.3.1 *Alternative approaches*

Other approaches were developed to be tested against the main strategy.

De Novo

While one of the pillars of this project knocking out organisms, through constraints when simulating a community model, size and scalability might raise performance issues with the introduction of increasingly large sets of models. Therefore, a version of the (*Knockout*) methodology was developed, in which instead of generating a single community metabolic model and knocking out organisms as needed, this version (*De Novo*) will instead create a new community metabolic model for each new candidate with the organisms considered for that candidate.

This version demanded a new *ea_run* and *evaluate* functions to be developed while the *Classes* and *Background* are shared with the original.

Multi-objective

This is a slightly altered version of the *Knockout* version, in which the fitness is composed of two variables, either maximizing the biomass and an exchange reaction, or maximizing the biomass and minimizing the number of organism present. Every generation, at random, one of the two fitness variables is used as fitness.

A *De Novo* version of this alternative was also developed. These versions share only the *Classes* with their single objective counterparts while using slightly altered versions of the *Background* functions, *ea_run*, and *evaluate*.

Inspyred

The Inspyred Python package was used to compare the developed work to already available options. The Genetic Algorithm (*ec.GA*) was the evolutionary computational strategy used for regular optimization, while the NSGA, *emo.NSGA2* was used to test multi-objective evolutionary algorithms.

3.4.4 *Code Availability*

The aforementioned work is, at the time, available in its entirety in the GitLab of the BIOSYSTEMS Research Group Web Services at https://gitlab.bio.di.uminho.pt/sousamd/ea-com-opt.

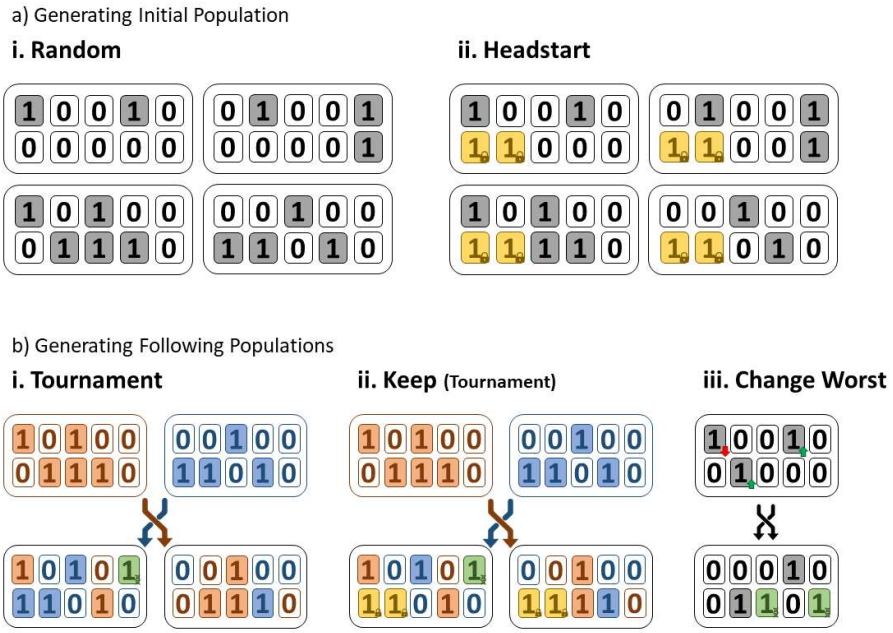


Figure 11: Population Generating Strategies. The initial population generating processes is depicted in a); in *i.* the population generated is completely random; whereas in *ii.* some positions are locked in, represented by the yellow cells with a lock token. The subsequent generation of populations is depicted in b); in *i.* parent candidates are selected based on fitness and generate offspring by combining their representation, while being subject to mutation, represented by the green cells with a DNA token; *ii. Keep* follows the same principle of *Tournament* but forces desired cells to be present; *iii.* compares the contribution of each present organism to the fitness value (represented through arrow tokens, where the downwards red arrow signal a lower relative value than the upwards green arrows), removes the worst performing candidate and introduces a random number of mutations to generate offspring.

Algorithm 5 Runs the Evolutionary Algorithm - *ea_run()*

Input: $option\_list(list)$, $list\_of\_models(list)$, $cmodel(FRAMED\ Community\ Model\ object)$, $obj\_list(list)$, $cons(dic)$, $quantity(int)$, $fitness(str)$, $goal(float)$

Reset existing dictionaries in *EAConfig*
Creates $obj\_list \leftarrow [\,]$ and $cons \leftarrow \{\,\}$ respectively if not provided
if *cmodel* is given **then**
    cmodel is used
else if *list_of_models* is given instead **then**
    A *cmodel* is created from it
end if
EAConfig attributes are set accordingly
Generates fitness reactions, *fit_reacs*, **if** *fitness* is given
Generates initial population, *popu*
Sets $gens \leftarrow 0$ counter and $exit\_flag \leftarrow False$
while $exit\_flag == False$ **do**
    $evaluate(cmodel,\ popu,\ obj\_list,\ cons,\ quantity,\ fit\_reacs)$
    best_score $\leftarrow$ fitness value of the best performing candidate
    if goal is provided **and** $best\_score > goal$ **then**
        $exit\_flag \leftarrow True$
    end if
    if $exit\_flag == False$ **then**
        Reassigns *popu* to a newly generated population.
        Increases *gens* by 1
        if $gens == EAConfig.max\_gen$ **then**
            $exit\_flag \leftarrow True$
        end if
    end if
end while
$evaluate(cmodel,\ popu,\ obj\_list,\ cons,\ quantity,\ fit\_reacs)$
bestcand $\leftarrow$ *Candidate* object with the best performing fitness

Output: $bestcand(Candidate\ object)$

Algorithm 6 Performs the Evaluation of the Candidates - *evaluate()*

Input: $cmodel(FRAMED\ Community\ Model\ object)$, $popu(list)$, $obj\_list(list)$, $cons(dic)$, $quantity(int)$, $fit\_reacs(list)$

if no *obj_list* **then**
    objs ← A copy of the reactions defined as objective in the model
end if
for candidate **in** *popu* **do**
    if candidate **in** $EAConfig.scoredic$ **then**
        Update the candidate attributes with the ones stored
    else if candidate **not in** $EAConfig.scoredic$ **then**
        indexes ← A list of indexes of the models to be knocked out in this candidate
        $list\_of\_model\_ids$ ← The model IDs correspondent of the values in *indexes*
        $model\_ko \leftarrow cmodel.knockout(list\_of\_model\_ids, obj\_list, cons)$
        if no *model_ko.values* **then**
            Updates candidate with minimal attributes
        end if
        if *fit_reacs is provided* **then**
            fit_list ← List of values of the fitness reactions
        else if no *fit_reacs* **then**
            fit_list ← List of values of the objective reactions
        end if
        score ← Sum of *fit_list*
        Updates candidate with the obtained attributes
        Reset the objectives with *objs*
    end if
end for

4

RESULTS AND DISCUSSION

The results obtained in this work will be discussed in this chapter, namely:

- performance comparison of different variables between the different variants developed for this purpose;
- a comparison between *Knockout* and *De novo* versions regarding flux bounds;
- a comparison between the multi-objective variants, and validation with a case study.

4.1 PERFORMANCE SPEED

The main objective of this project was to use the ability to knockout organisms from existing metabolic models as means to perform community composition optimization through EAs. Thus, a comparison with another approach was mandatory, as well as comparing existing alternatives with the developed strategies.

Figure 12 shows the recorded times of these algorithms for the following conditions: 10 generations, population size of 15 candidates, pool size of 10, 20 and 50 models across three different methods: *Knockout* method creates the community model first with all models in the pool and performs the knockouts for every candidate representation, while the remaining methods create a new community model for each candidate representation.

The *Inspyred* based method takes noticeably more time than the developed work to perform the same task, regardless of pool size. As the evaluating function is the same in every approach tested, this difference could be explained by Inspyred not saving solutions to save time when evaluating candidates with the same representation.

Furthermore, it is observable that both *Knockout* and *De novo* strategies have areas where they perform better than the other. With the increase in pool size, *De novo* has an advantage as it does not have to load a community model with the whole pool size. Moreover, the smaller the number of models desired in the final solution, the less important the pool size becomes, the repetition of candidate representations is less probable, thus less time to be

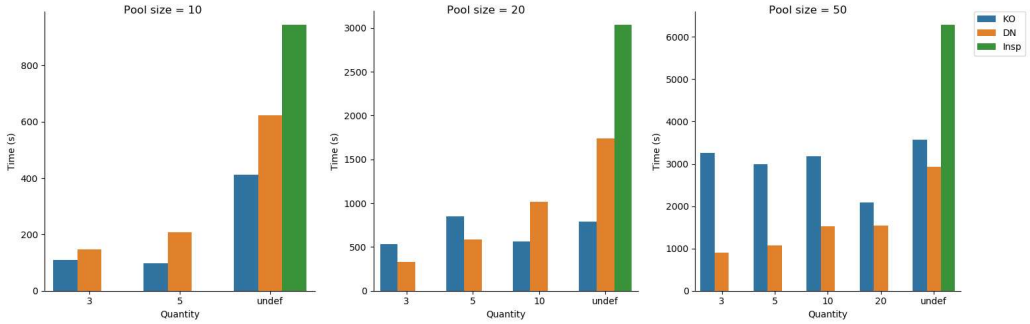


Figure 12: Average performance speed comparisons varying some of the key parameters in an *EA* run. These ran for 10 generations each with populations of 15 candidates and the default biomass as the objective function. Pool size refers to the number of models the algorithms pulls from; Quantity refers to the number of models present in the candidate solutions, where *undef* means that no set number of models present was defined for the candidates; Time in seconds is the time it took the algorithm to load the models and run the evolutionary algorithm until the end of the defined generations. As indicated by the legend, blue bars represent the *Knockout* strategy, orange bars represent the *De novo* strategy and the green bars were performed with the Inspyred Python package.

saved in the run. *Knockout* seems advantageous with a relatively medium sized pool size and amount of models in the solutions.

Finally, it should be noticed that the increase in generation and population sizes will increase the preference of *Knockout* over *De novo*, as the time initially spent has less impact in the time of the run.

4.2 MULTI-OBJECTIVE

The nature of EAs calls for the inclusion of multi-objective strategies. Given the Inspyred Python package options to perform EMO through the NSGA method, the latter was compared to the developed work in this project.

Ten arbitrarily selected models from the AGORA collection constituted the pool of models the algorithms drew from:

- *Yokonella regensburgei* ATCC 43003,
- *Acinetobacter junii* SH205,
- *Clostridiales sp* 1 7 47FAA,
- *Achromobacter xylosoxidans* A8,
- *Achromobacter xylosoxidans* NBRC 15126,
- *Acidaminococcus intestini* RyC MR95,
- *Acidaminococcus sp* D21,

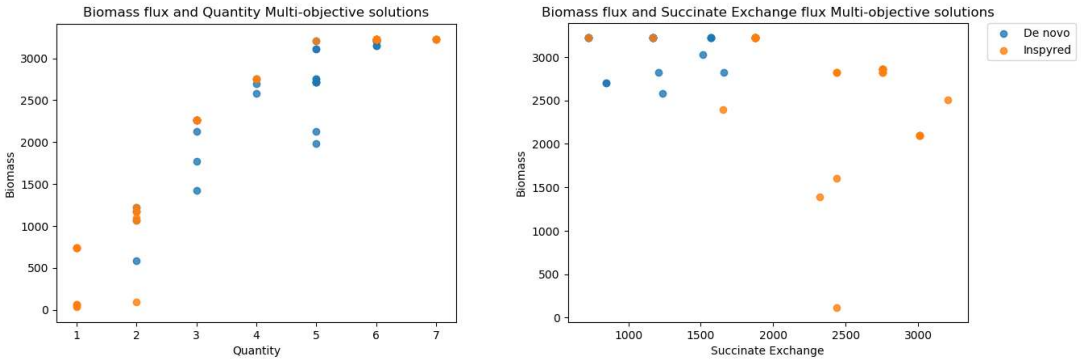


Figure 13: Multi objective results with *De novo* method and Inspyred package. The plot on the left displays the maximization of the biomass flux and the minimization of the quantity of models in the candidate solutions while the plot on the right displays the maximization of the biomass flux and the flux of the *succinate exchange* reaction. Blue dots represent solutions obtained through *De novo* method and orange dots represent solutions obtained with the Inspyred Python package. Flux values units in mmol/gDW/hr.

- *Acinetobacter calcoaceticus* PHEA 2,
- *Acinetobacter lwoffii* WJ10621 and
- *Actinobacillus pleuropneumoniae* L20.

This pool of models was ran in a multi-objective version of the *De novo* strategy, with the Inspyred function for NSGA for 10 generations and a population size of 15. Two versions were tested, one where the objectives were the maximization of the biomass growth rate and the minimization of the amount of models in the candidate solutions, and a second version where the objective was the maximization of the biomass growth rate and the maximization of the, arbitrarily selected, *succinate exchange* reaction flux, as demonstrated in Figure 13.

It is possible to assert that both strategies exhibit similar results for the biomass growth rate and amount of organisms optimization, albeit the Inspyred version admits less solutions outside the Pareto-front, which is to be expected as the *De novo* version does not take Pareto optimality into consideration. When optimizing the biomass growth rate and an exchange reaction, there is a more spread out set of solutions. This stems from the exchange reaction flux being a variable with a larger solution space than the number of organisms in a model, which is limited to positive integers. The distribution along the Pareto-front is similar to the previous example, in which Inspyred places more solutions alongside the frontier. However, in this case, the *De novo* strategy displays solutions more oriented towards the optimization of the biomass growth rate due to it not considering Pareto optimality and the significant number of solutions in which the reaction flux value is negative or null.

It is relevant to note that the time comparisons in 4.1 also apply to the multi-objective versions of the algorithms, making Inspyred a slower method when compared with this work.

4.3 FLUX VALUES OF NON-OBJECTIVE REACTIONS

When choosing fitness reactions different from the reactions defined as objective, a situation arises where there are no guarantees that the fitness will be constant and coherent when simulated. Due to the nature of FBA and LP in general, fluxes can display different values despite reaching the same value in the objective reactions selected. Therefore, there are instances where flux values are different between *Knockout* and *De novo* strategies, despite obtaining the same result in the objective function.

Furthermore, the default flux bound, set when reactions have null values in flux bounds, is relevant to the outcome of the aforementioned situation and small differences can lead to completely different flux values, as demonstrated in Figure 14. To demonstrate these differences, sequential FBA were ran for both *Knockout* and *De novo* strategies, with the biomass maximization as the objective and *Taurine exchange* reaction as fitness (chosen arbitrarily). The default bound value was increased by one between each run, from 0 until 2000.

Apart from the logical increase in flux value, it is possible to note the difference in flux values between the two strategies. These results include intervals where the flux value is present in one version an non-existent the other.

These results show the importance of having high quality metabolic models that, preferably do not contain null values in the reaction bounds. Although, it is important to notice that these community models were loaded with complete medium thus being more likely to produce these results. While with a more restrictive media, it is expected for the flux values to be more coherent. The adoption of *Parsimonious Flux Balance Analysis (pFBA)* [153] as the evaluating tool instead of FBA could also be of use in mitigating this issue.

4.4 VALIDATION

A case study was selected from the work from Ponomarova *et al.* [154], where lactic acid bacteria *Lactococcus lactis* was shown to grow in mutualism with *Lactoccocus plantarum* with help from a nitrogen overflow from *Saccharomyces cerevisiae*, to validate the developed software.

EA runs were performed for 5 generations and population size of 15 candidates, with a pool size of 24 models, with the biomass of *L. lactis* as the objective and fitness reaction. The *headstart* and *keep* options were selected to maintain the *L. lactis* in every solution. A

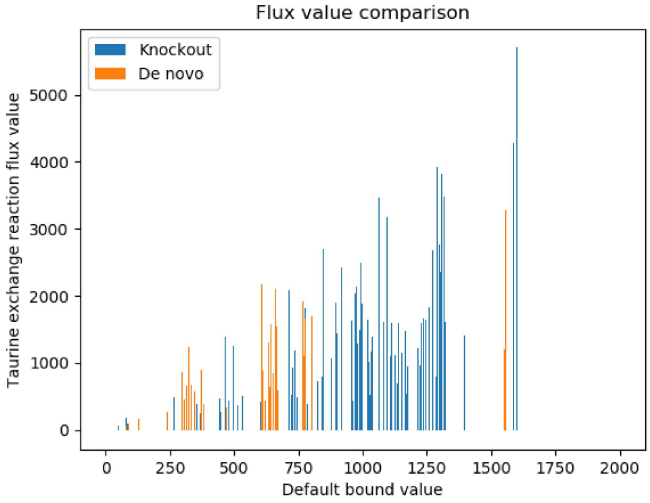


Figure 14: Flux value comparison of an arbitrarily chosen reaction, *Taurine exchange*, not used as objective, with different default bound values. Sequential *FBA* was run increasing the default bound value by one between each run from 0 to 2000, for each of the strategies (*Knockout* in blue, *De novo* in orange. Flux values units in mmol/gDW/hr.

minimal medium indicated for the population described in the work of Ponomarova *et al.* was selected instead of the complete default medium to attain comparable results with the publication.

When running with a selected candidate size of 3 organisms per candidate, the solution output by the algorithm was the community with *L. lactis*, *L. plantarum* and *S. cerevisiae*, with a score of $813.0 mmol/gDW/hr$ (the default value substituted in reactions where the bound is null). A second run was performed, this time with both the biomass of *L. lactis* and *L. plantarum* as the objective and fitness reactions, and again the same solution was output,

Table 3: Final solutions of EA runs based on the works of Ponomarova *et al.* [154]. Run 1 maximizes the biomass growth rate of *L. lactis* and *L. plantarum*, with *L. lactis* present in the solutions. Run 2 maximizes the biomass growth rate of *L. lactis* with *L. lactis* and *L. plantarum* present in the solution. Organisms in bold were forced to be present by the algorithm. Quantity refers to the number of organisms in each candidate. Fitness units in (mmol/gDW/hr)

#	Quantity	Objective Function	Organisms in Final Solution			Fitness
1	3	Maximizing biomass growth rate of *L. lactis* and *L. plantarum*	***L. lactis***	*L. plantarum*	*S. cerevisiae*	1147.6
2	3	Maximizing biomass growth rate of *L. lactis*	***L. lactis***	***L. plantarum***	*S. cerevisiae*	*813.0*

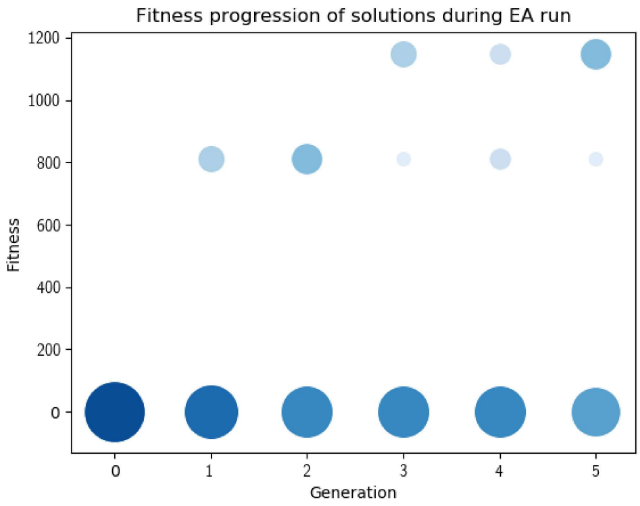


Figure 15: Fitness values progression of the candidates in the *EA* run 1 described in Table 3. Each generation encompasses a population of 15 candidates, distributed through different fitness values. Dot size and color are indicative of the relative proportion of candidates with that fitness value. Fitness value units in mmol/gDW/hr.

this time with a score of $1147.6 mmol/gDW/hr$. These results were replicated with both the *Knockout* and *De novo* strategies. These results are detailed in Table 3.

Due to the nature of the medium chosen for these *EA* runs, it was expected that only communities containing the organisms described would thrive in it. Accordingly, the only candidates with relevant fitness values are communities that share some organisms with the final solution. As such, it is expected a large number of candidates with no fitness, as shown in Figure 15. It is observable how the number of candidates with optimal fitness value increases each run, and how less candidates exhibit no fitness with the passing of generations.

These results in accordance with the publications confirm the biological solidity of the developed work and the interchangeability of the two strategies.

5

CONCLUSIONS

The purpose of this work was to develop an *in silico* method to optimize the composition of a microbial community, given a specified objective, through the implementation of knockouts of certain members of a community metabolic model. This method was to use EAs due to their high plasticity and adaptability to any kind of problem.

The desired objectives were then developed into a framework using Python as the programming language, with FRAMED as the backbone. Different strategies were developed to either address the objective or to serve as comparison, including replicating the work using Inspyred, as an outside control.

Knocking out of organisms inside a community model was shown to provide advantages in certain situations, when compared to the creation of a new community model for every candidate in the *EA*, while maintaining its interchangeability between the two strategies when it comes to the output. The replication of the examples in the work of Ponomarova *et al.* [154] also serve as validation.

This work serves, once more, to stress the need for good curation processes of metabolic models, as well as a strongly defined annotation methods. These needs will only increase the incremental interest in community metabolic models in opposition to single organism metabolic models.

After the initial objectives were ensured, there was also an interest in the introduction of multi-objective strategies for microbial community composition optimization, which were shown to be compatible with the framework developed in this work.

FUTURE WORK

Possible future work in this area could include the use of different evaluating functions in alternative to FBA to achieve more consistent results, namely with pFBA. The further development of the multi-objective evolutionary algorithms for optimizing community composition of microbial communities was shown to have potential. It could also be of interest the inclusion of the medium composition of a microbial community as a variable in the *EA*, alongside the community composition.

BIBLIOGRAPHY

[1] Jeremy S. Edwards, Rafael U. Ibarra, and Bernhard Ø. Palsson. In silico predictions of Escherichia coli metabolic capabilities are consistent with experimental data. *Nature Biotechnology*, 19:125–130, 2001.

[2] Timo R. Maarleveld, Ruchir A. Khandelwal, Brett G. Olivier, Bas Teusink, and Frank J. Bruggeman. Basic concepts and principles of stoichiometric modeling of metabolic networks. *Biotechnology Journal*, 8(9):997–1008, 2013.

[3] Amit Varma and Bernhard Ø. Palsson. Metabolic flux balancing: Basic concepts, scientific and practical use. *Nature Biotechnology*, 12(10):994–998, 1994.

[4] Sharon J. Wiback, Iman Famili, Harvey J. Greenberg, and Bernhard Ø. Palsson. Monte Carlo sampling can be used to determine the size and shape of the steady-state flux space. *Journal of Theoretical Biology*, 228(4):437–447, 2004.

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

[6] John N. Thompson. The Evolution of Species Interactions. *Science*, 284(5423):2116–2118, 1999.

[7] Sergey Stolyar, Steve Van Dien, Kristina Linnea Hillesland, Nicolas Pinel, Thomas J. Lie, John A. Leigh, and David A. Stahl. Metabolic modeling of a mutualistic microbial community. *Molecular Systems Biology*, 3(92):1–14, 2007.

[8] Stefanía Magnúsdóttir, Almut Heinken, Laura Kutt, Dmitry A. Ravcheev, Eugen Bauer, Alberto Noronha, Kacy Greenhalgh, Christian Jäger, Joanna Baginska, Paul Wilmes, Ronan M. T. Fleming, and Ines Thiele. Generation of genome-scale metabolic reconstructions for 773 members of the human gut microbiota. *Nature Biotechnology*, 35(1):81–89, 2017.

[9] Ruchir A. Khandelwal, Brett G. Olivier, Wilfred F. M. Röling, Bas Teusink, and Frank J. Bruggeman. Community Flux Balance Analysis for Microbial Consortia at Balanced Growth. *PLoS ONE*, 8(5), 2013.

[10] Siu Hung Joshua Chan, Margaret N. Simons, and Costas D. Maranas. SteadyCom: Predicting microbial abundances while ensuring community stability. *PLoS Computational Biology*, 13(5):1–25, 2017.

[11] Gregory Stephanopoulos. Metabolic fluxes and metabolic engineering. *Metabolic Engineering*, 1(1):1–11, 1999.

[12] Zbigniew Michalewicz. Genetic Algorithms + Data Structures = Evolution Programs, 1996.

[13] Zachary L. Fowler, William W. Gikandi, and Mattheos A. G. Koffas. Increased malonyl coenzyme A biosynthesis by tuning the Escherichia coli metabolic network and its application to flavanone production. *Applied and Environmental Microbiology*, 75(18):5831–5839, 2009.

[14] Hiroaki Kitano. Systems biology: A brief overview. *Science*, 295(5560):1662–1664, 2002.

[15] Norbert Weiner. *Cybernetics: Or Control and Communication in the Animal and the Machine*. The MIT Press, Cambridge, Massachussets, 2nd edition, 1948.

[16] Water B. Cannon. *The Wisdom of the Body*. W. W. Norton, New York, 1932.

[17] Ludwig von Bertalanffy. *General System Theory: Foundations, Develoment, Applications*. George Braziller, Inc., New York, 1968.

[18] Hiroaki Kitano. *Foundations of Systems Biology*. The MIT Press, Cambridge, Massachussets, 1st edition, 2001.

[19] Marie E. Csete and John C. Doyle. Reverse engineering of biological complexity. *Science*, 295(5560):1664–1669, 2002.

[20] Rainer Breitling. What is systems biology? *Frontiers in Physiology*, 1 MAY(May):1–5, 2010.

[21] Michael D. Resnik and Philip Kitcher. *The Nature of Mathematical Knowledge*. Cambridge University Press, 1999.

[22] James E. Bailey. Mathematical Modeling and Analysis in Biochemical Engineering: Past Accomplishments and Future Opportunities. *Biotechnology Progress*, 32(14):8–20, 1998.

[23] Andreas Karoly Gombert and Jens Nielsen. Mathematical modelling of metabolism. *Current Opinion in Biotechnology*, 11(2):180–186, 2000.

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

[25] Egils Stalidzans, Andrus Seiman, Karl Peebo, Vitalijs Komasilovs, and Agris Pentjuss. Model-based metabolism design: constraints for kinetic and stoichiometric models. *Biochemical Society Transactions*, 46(2):261–267, 2018.

[26] Finn Hynne, Sune Danø, and Preben Graae Sørensen. *Full-scale model of glycolysis in Saccharomyces cerevisiae*, volume 94. 2001.

[27] Reinis Rutkis, Uldis Kalnenieks, Egils Stalidzans, and David A. Fell. Kinetic modelling of the Zymomonas mobilis Entner-Doudoroff pathway: Insights into control and functionality. *Microbiology (United Kingdom)*, 159(PART 12):2674–2689, 2013.

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

[29] Benjamin D. Heavner, Kieran Smallbone, Nathan D. Price, and Larry P. Walker. Version 6 of the consensus yeast metabolic network refines biochemical coverage and improves model performance. *Database*, 2013:1–5, 2013.

[30] Richard A. Notebaart, Frank H. J. van Enckevort, Christof Francke, Roland J. Siezen, and Bas Teusink. Accelerating the reconstruction of genome-scale metabolic networks. *BMC Bioinformatics*, 7:1–10, 2006.

[31] Francisco Llaneras and Jesús Picó. Stoichiometric modelling of cell metabolism. *Journal of Bioscience and Bioengineering*, 105(1):1–11, 2008.

[32] Paulo Maia, Miguel Rocha, and Isabel Rocha. In Silico Constraint-Based Strain Optimization Methods: the Quest for Optimal Cell Factories. *Microbiology and Molecular Biology Reviews*, 80(1):45–67, 2016.

[33] Nathan D. Price, Jennifer L. Reed, and Bernhard Ø. Palsson. Genome-scale models of microbial cells: Evaluating the consequences of constraints. *Nature Reviews Microbiology*, 2(11):886–897, 2004.

[34] Paul B. Weisz. Diffusion and Chemical Transformation. *Science*, 179(4072):433–440, 1973.

[35] Virgilio L. Lew and Robert M. Bookchin. Volume, pH and Ion-content regulation in human red cells. *Journal of Membrane Biology*, 92:57–74, 1986.

[36] Lubert Stryer. *Biochemistry*. W. H. Freeman, New York, 3rd edition, 1988.

[37] R. John Ellis. Macromolecular crowding: Obvious but underappreciated. *Trends in Biochemical Sciences*, 26(10):597–604, 2001.

[38] Antoine Danchin. By way of introduction: Some constraints of the cell physics that are usually forgotten, but should be taken into account for in silico genome analysis. *Biochimie*, 78(5):299–301, 1996.

[39] Jennifer L. Reed, Thuy D. Vo, Christophe H. Schilling, and Bernhard Ø. Palsson. An expanded genome-scale model of Escherichia coli K-12 (i JR904 GSM/GPR). *Genome Biology*, 4(9):R54, 2003.

[40] Bernhard Ø. Palsson. *Systems Biology: Properties of Reconstructed Networks. Cambridge University Press, New York.* New York, 2006.

[41] Markus W. Covert, Christophe H. Schilling, Iman Famili, Jeremy S. Edwards, Igor I. Goryanin, Evgeni Selkov, and Bernhard Ø. Palsson. Metabolic modeling of microbial strains in silico. *Journal of Theoretical Biology*, 271(3-4):493, 2001.

[42] Allan E. Konopka. What is microbial community ecology. *ISME Journal*, 3(11):1223–1230, 2009.

[43] Philip L. Bond, Steven P. Smriga, and Jillilan F. Banfield. Phylogeny of microorganisms populating a thick, subaerial, predominantly lithotrophic biofilm at an extreme acid mine drainage site. *Applied and Environmental Microbiology*, 66(9):3842–3849, 2000.

[44] J. Gregory Caporaso, Christian L. Lauber, Elizabeth K. Costello, Donna Berg-Lyons, Antonio Gonzalez, Jesse Stombaugh, Dan Knights, Pawel Gajer, Jacques Ravel, Noah Fierer, Jeffrey I. Gordon, and Rob Knight. Moving pictures of the human microbiome. *Genome Biology*, 12(5), 2011.

[45] Cameron Wagg, S. Franz Bender, Franco Widmer, and Marcel G. A. van der Heijden. Soil biodiversity and soil community composition determine ecosystem multifunctionality. *Proceedings of the National Academy of Sciences of the United States of America*, 111(14):5266–5270, 2014.

[46] Hyun-Seob Song, William Cannon, Alexander S. Beliaev, and Allan E. Konopka. Mathematical Modeling of Microbial Community Dynamics: A Methodological Review. *Processes*, 2(4):711–752, 2014.

[47] Christopher S. Henry, Hans C. Bernstein, Pamela Weisenhorn, Ronald C. Taylor, Joon Yong Lee, Jeremy Zucker, and Hyun-Seob Song. Microbial Community Metabolic Modeling: A Community Data-Driven Network Reconstruction. *Journal of Cellular Physiology*, 231(11):2339–2345, 2016.

[48] Giovanni Bacci, Maria Teresa Ceccherini, Alessia Bani, Marco Bazzicalupo, Maurizio Castaldini, Marco Galardini, Luciana Giovannetti, Stefano Mocali, Roberta Pastorelli,

Ottorino Luca Pantani, Paola Arfaioli, Giacomo Pietramellara, Carlo Viti, Paolo Nannipieri, and Alessio Mengoni. Exploring the dynamics of bacterial community composition in soil: the pan-bacteriome approach. *Antonie van Leeuwenhoek, International Journal of General and Molecular Microbiology*, 107(3):785–797, 2015.

[49] Elhanan Borenstein, Martin Kupiec, Marcus W. Feldman, and Eytan Ruppin. Large-scale reconstruction and phylogenetic analysis of metabolic environments. *Proceedings of the National Academy of Sciences of the United States of America*, 105(38):14482–14487, 2008.

[50] Thomas Handorf, Oliver Ebenhöh, and Reinhart Heinrich. Expanding metabolic networks: Scopes of compounds, robustness, and evolution. *Journal of Molecular Evolution*, 61(4):498–512, 2005.

[51] Ines Thiele and Bernhard Ø. Palsson. A Protocol for Generating High Quality Genome-Scale Metabolic Reconstruction. *Nat Protoc*, 5(1):93–121, 2010.

[52] Saeed Shoaie, Pouyan Ghaffari, Petia Kovatcheva-Datchary, Adil Mardinoglu, Partho Sen, Estelle Pujos-Guillot, Tomas De Wouters, Catherine Juste, Salwa Rizkalla, Julien Chilloux, Lesley Hoyles, Jeremy K. Nicholson, Joel Dore, Marc E. Dumas, Karine Clement, Fredrik Bäckhed, and Jens Nielsen. Quantifying Diet-Induced Metabolic Changes of the Human Gut Microbiome. *Cell Metabolism*, 22(2):320–331, 2015.

[53] Robert Schuetz, Lars Kuepfer, and Uwe Sauer. Systematic evaluation of objective functions for predicting intracellular fluxes in Escherichia coli. *Molecular Systems Biology*, 3(119), 2007.

[54] Joanne M. Savinell and Bernhard Ø. Palsson. Optimal selection of metabolic fluxes for in vivo measurement. I. Development of mathematical methods. *Journal of Theoretical Biology*, 155(2):201–214, 1992.

[55] Markus W. Covert, Christophe H. Schilling, and Bernhard Ø. Palsson. Regulation of Gene Expression in Flux Balance Models of Metabolism. *Journal of Theoretical Biology*, 213(3):73–88, 2001.

[56] Radhakrishnan Mahadevan and Christophe H. Schilling. The effects of alternate optimal solutions in constraint-based genome-scale metabolic models. *Metabolic Engineering*, 5(4):264–276, 2003.

[57] Steinn Gudmundsson and Ines Thiele. Computationally efficient flux variability analysis. *BMC Bioinformatics*, 11(2):2–4, 2010.

[58] Daniel Segrè, Dennis Vitkup, and George M. Church. Analysis of optimality in natural and perturbed metabolic networks. *Proceedings of the National Academy of Sciences of the United States of America*, 99(23):15112–15117, 2002.

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

[60] Ali R. Zomorrodi and Costas D. Maranas. OptCom: A multi-level optimization framework for the metabolic modeling and analysis of microbial communities. *PLoS Computational Biology*, 8(2), 2012.

[61] Ali R. Zomorrodi, Mohammad Mazharul Islam, and Costas D. Maranas. D-OptCom: Dynamic Multi-level and Multi-objective Metabolic Modeling of Microbial Communities. *ACS Synthetic Biology*, 3(4):247–257, 2014.

[62] Kai Zhuang, Mounir Izallalen, Paula Mouser, Hanno Richter, Carla Risso, Radhakrishnan Mahadevan, and Derek R. Lovley. Genome-scale dynamic modeling of the competition between Rhodoferax and Geobacter in anoxic subsurface environments. *ISME Journal*, 5(2):305–316, 2011.

[63] Timothy J. Hanly and Michael A. Henson. Dynamic metabolic modeling of a microaerobic yeast co-culture: Predicting and optimizing ethanol production from glucose/xylose mixtures. *Biotechnology for Biofuels*, 6(1):1–16, 2013.

[64] Beatriz García-Jiménez, José Luis García, and Juan Nogales. FLYCOP: Metabolic modeling-based analysis and engineering microbial communities. *Bioinformatics*, 34(17):i954–i963, 2018.

[65] William R. Harcombe, William J. Riehl, Ilija Dukovski, Brian R. Granger, Alex Betts, Alex H. Lang, Gracia Bonilla, Amrita Kar, Nicholas Leiby, Pankaj Mehta, Christopher J. Marx, and Daniel Segrè. Metabolic Resource Allocation in Individual Microbes Determines Ecosystem Interactions and Spatial Dynamics. *Cell Reports*, 7:1104–1115, 2014.

[66] Alexander Eng and Elhanan Borenstein. An algorithm for designing minimal microbial communities with desired metabolic capacities. *Bioinformatics*, 32(13):2008–2016, 2016.

[67] Tim Bray, Jean Paoli, C.M. Sperberg-McQueen, Eve Maler, and François Yergeau. Extensible markup language (XML) 1.0 (second edition), W3C Recommendation 26-November-2008, 2013.

[68] Frédéric Achard, Guy Vaysseix, and Emmanuel Barillot. XML, bioinformatics and data integration. *Bioinformatics*, 17(2):115–125, 2001.

[69] Michael Hucka, Andrew Finney, Benjamin J. Bornstein, Sarah M. Keating, Bruce E. Shapiro, Joanne Matthews, Benjamin L. Kovitz, Maria J. Schilstra, Akira Funahashi, John C. Doyle, and Hiroaki Kitano. Evolving a lingua franca and associated software infrastructure for computational systems biology: the Systems Biology Markup Language (SBML) project. *Systems Biology*, 1(1), 2004.

[70] Michael Hucka, Andrew Finney, Herbert M. Sauro, Hamid Bolouri, John C. Doyle, Hiroaki Kitano, Adam P. Arkin, Benjamin J. Bornstein, Dennis Bray, Athel Cornish-Bowden, Autumn A. Cuellar, Serge Dronov, Ernst D. Gilles, Martin Ginkel, Victoria Gor, Igor I. Goryanin, Warren J. Hedley, Charles Hodgman, Jan-Hendrik S. Hofmeyr, Peter J. Hunter, Nick S. Juty, Jay L. Kasberger, Andreas Kremling, Ursula Kummer, Nicolas Le Novère, Les M. Loew, Daniel Lucio, Pedro Mendes, Eric Minch, Eric D. Mjolsness, Yoichi Nakayama, Melanie R. Nelson, Poul Michael Fønss Nielsen, Takeshi Sakurada, James Choate Schaff, Bruce E. Shapiro, Thomas S. Shimizu, Hugh D. Spence, Joerg Stelling, Koichi Takahashi, Masaru Tomita, John Wagner, and Jian Wang. The systems biology markup language (SBML): A medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–531, 2003.

[71] Laurent Heirendt, Sylvain Arreckx, Thomas Pfau, Sebastián N. Mendonza, Anne Richelle, Almut Heinken, Hulda S. Haraldsdóttir, Jacek Wachowiak, Sarah M. Keating, Vanja Vlasov, Stefanía Magnúsdóttir, Chiam Yu Ng, German Preciat, Alise Žagare, Siu Hung Joshua Chan, Maike K. Aurich, Catherine M. Clancy, Jennifer Modamio, John T. Sauls, Alberto Noronha, Aarash Bordbar, Benjamin Cousins, Diana C. El Assal, Luis V. Valcarcel, Iñigo Apaolaza, Susan Chaderi, Masoud Ahookhosh, Marouen Ben Guebila, Andrejs Kostromins, Nicolas Sompairac, Hoai M. Le, Ding Ma, Yuekai Sun, Lin Wang, James T. Yurkovich, Miguel A. P. Oliveira, Phan T. Vuong, Lemmer P. El Assal, Inna Kuperstein, Andrei Zinovyev, H. Scott Hinton, William A. Bryant, Francisco J. Aragón Artacho, Francisco J. Planes, Egils Stalidzans, Alejandro Maass, Santosh Vempala, Michael Hucka, Michaels A. Saunders, Costas D. Maranas, Nathan E. Lewis, Thomas Sauter, Bernhard Ø. Palsson, Ines Thiele, and Ronan M. T. Fleming. Creation and analysis of biochemical constraint-based models: the COBRA Toolbox v3.0. *accepted in Nature Protocols*, page 78, 2018.

[72] MathWorks. MATLAB Documentation, 2019.

[73] James Gosling, Bill Joy, Guy J. Steele, Gilad Bracha, and Alex Buckley. *The Java Language Specification, Java SE 8 Edition*. Addison-Wesley Professional, 1st edition, 2014.

[74] R Core Team. R: A Language and Environment for Statistical Computing, 2018.

[75] Guido Van Rossum and Fred L Drake. *Python Tutorial*. 1995.

[76] Bjarne Stroustrup. *The C++ Programming Language*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 3rd edition, 2000.

[77] Steffen Klamt, Julio Saez-Rodriguez, and Ernst D. Gilles. Structural and functional analysis of cellular networks with CellNetAnalyzer. *BMC Systems Biology*, 1(2):1–13, 2007.

[78] Isabel Rocha, Paulo Maia, Pedro Evangelista, Paulo Vilaça, Simão Soares, José P. Pinto, Jens Nielsen, Kiran Raosaheb Patil, Eugénio C. Ferreira, and Miguel Rocha. OptFlux: an open-source software platform for in silico metabolic engineering. *BMC Systems Biology*, 4(45):12, 2010.

[79] Gabriel Gelius-Dietrich, Abdelmoneim Amer Desouki, Claus Jonathan Fritzemeier, and Martin J. Lercher. SOFTWARE Open Access sybil – Efficient constraint-based modelling in R. *BMC Systems Biology*, 7, 2013.

[80] Ali Ebrahim, Bernhard Ø. Palsson, Joshua A. Lerman, and Daniel R. Hyduke. COBRApy: COnstraints-Based Reconstruction and Analysis for Python. *BMC Systems Biology*, 7(74):6, 2013.

[81] Brett G. Olivier, Johann M. Rohwer, and Jan-Hendrik S. Hofmeyr. Modelling cellular systems with PySCeS. *Bioinformatics*, 21(4):560–561, 2005.

[82] Daniel Machado, Sergej Andrejev, Kai Zhuang, Marta Matos, Sara Correia, and Nikolaus Sonnenschein. cdanielmachado/framed: First python 3 compatible release, 2018.

[83] Albert Gevorgyan, Michael E. Bushell, Claudio Avignone-Rossa, and Andrzej M. Kierzek. SurreyFBA: A command line tool and graphics user interface for constraint-based modeling of genome-scale metabolic reaction networks. *Bioinformatics*, 27(3):433–434, 2011.

[84] Adam P. Arkin, Robert W. Cottingham, Christopher S. Henry, Nomi L. Harris, Rick L. Stevens, Sergei Maslov, Paramvir Dehal, Doreen Ware, Fernando Perez, Shane Canon, Michael W. Sneddon, Matthew L. Henderson, William J. Riehl, Dan Murphy-Olson, Stephen Y. Chan, Roy T. Kamimura, Sunita Kumari, Meghan M. Drake, Thomas S. Brettin, Elizabeth M. Glass, Dylan Chivian, Dan Gunter, David J. Weston, Benjamin H. Allen, Jason Baumohl, Aaron A. Best, Ben Bowen, Steven E. Brenner, Christopher C. Bun, John-Marc Chandonia, Jer-Ming Chia, Ric Colasanti, Neal Conrad, James J. Davis, Brian H. Davison, Matthew DeJongh, Scott Devoid, Emily Dietrich, Inna Dubchak, Janaka N. Edirisinghe, Gang Fang, José P. Faria, Paul M. Frybarger, Wolfgang Gerlach, Mark Gerstein, Annette Greiner, James Gurtowski, Holly L. Haun, Fei He, Rashmi Jain,

Marcin P. Joachimiak, Kevin P. Keegan, Shinnosuke Kondo, Vivek Kumar, Miriam L. Land, Folker Meyer, Marissa Mills, Pavel S. Novichkov, Taeyun Oh, Gary J. Olsen, Robert Olson, Bruce Parrello, Shiran Pasternak, Erik Pearson, Sarah S. Poon, Gavin A. Price, Srividya Ramakrishnan, Priya Ranjan, Pamela C. Ronald, Michael C. Schatz, Samuel M. D. Seaver, Maulik Shukla, Roman A. Sutormin, Mustafa H. Syed, James Thomason, Nathan L. Tintle, Daifeng Wang, Fangfang Xia, Hyunseung Yoo, Shinjae Yoo, and Dantong Yu. KBase: The United States Department of Energy Systems Biology Knowledgebase. *Nature Biotechnology*, 36(7):566–569, 2018.

[85] Jens Nielsen. Metabolic Engineering: Techniques for Analysis of Targets for Genetic Manipulations. *Biotechnology and Bioengineering*, 58:125–132, 1998.

[86] James E. Bailey. Toward a Science of Metabolic Engineering. *Science*, 252(5013):1668–1675, 1991.

[87] Douglas C. Cameron and I-Teh Tong. Cellular and metabolic engineering - An overview. *Applied Biochemistry and Biotechnology*, 38(1-2):105–140, 1993.

[88] Gregory Stephanopoulos. Metabolic engineering. *Current Opinion in Biotechnology*, 5:196–200, 1994.

[89] Jeong Wook Lee, Dokyun Na, Jong Myoung Park, Joungmin Lee, Sol Choi, and Sang Yup Lee. Systems metabolic engineering of microorganisms for natural and non-natural chemicals. *Nature Chemical Biology*, 8(6):536–546, 2012.

[90] Eric J. Steen, Yisheng Kang, Gregory Bokinsky, Zhihao Hu, Andreas Schirmer, Amy McClure, Stephen B. Del Cardayre, and Jay D. Keasling. Microbial production of fatty-acid-derived fuels and chemicals from plant biomass. *Nature*, 463(7280):559–562, 2010.

[91] Yu Kyung Jung, Tae Yong Kim, Si Jae Park, and Sang Yup Lee. Metabolic engineering of Escherichia coli for the production of polylactic acid and its copolymers. *Biotechnology and Bioengineering*, 105(1):161–71, 2010.

[92] Mark A. Eiteman, Sarah A. Lee, and Elliot Altman. A co-fermentation strategy to consume sugar mixtures effectively. *Journal of Biological Engineering*, 2:1–8, 2008.

[93] Kwang Ho Lee, Jin Hwan Park, Tae Yong Kim, Hyun Uk Kim, and Sang Yup Lee. Systems metabolic engineering of Escherichia coli for L-threonine production. *Molecular Systems Biology*, 3(149), 2007.

[94] Zhi Gang Qian, Xiao-Xia Xia, and Sang Yup Lee. Metabolic engineering of Escherichia coli for the production of putrescine: A four carbon diamine. *Biotechnology and Bioengineering*, 104(4):651–662, 2009.

[95] Clément Auriol, Gwénaëlle Bestel-Corre, Jean-Baptiste Claude, Philippe Soucaille, and Isabelle Meynial-Salles. Stress-induced evolution of Escherichia coli points to original concepts in respiratory cofactor selectivity. *Proceedings of the National Academy of Sciences of the United States of America*, 108(4):1278–1283, 2011.

[96] Josselin Noirel, Saw Yen Ow, Guido Sanguinetti, and Phillip C. Wright. Systems biology meets synthetic biology: A case study of the metabolic effects of synthetic rewiring. *Molecular BioSystems*, 5(10):1214–1223, 2009.

[97] Effendi Leonard, Parayil Kumaran Ajikumar, Kelly Thayer, Wen-Hai Xiao, Jeffrey D. Mo, Bruce Tidor, Gregory Stephanopoulos, and Kristala L. J. Prather. Combining metabolic and protein engineering of a terpenoid biosynthetic pathway for overproduction and selectivity control. *Proceedings of the National Academy of Sciences of the United States of America*, 107(31):13654–13659, 2010.

[98] Xueli Zhang, Kaemwich Jantama, Jonathan C. Moore, Laura R. Jarboe, Keelnatham T. Shanmugam, and Lonnie O. Ingram. Metabolic evolution of energy-conserving pathways for succinate production in Escherichia coli. *Proceedings of the National Academy of Sciences of the United States of America*, 106(48):20180–20185, 2009.

[99] Shota Atsumi, Tung Yun Wu, Iara M.P. Machado, Wei Chih Huang, Pao Yang Chen, Matteo Pellegrini, and James C. Liao. Evolution, genomic analysis, and reconstruction of isobutanol tolerance in Escherichia coli. *Molecular Systems Biology*, 6(449):1–11, 2010.

[100] Anthony P. Burgard, Priti Pharkya, and Costas D. Maranas. OptKnock: A Bilevel Programming Framework for Identifying Gene Knockout Strategies for Microbial Strain Optimization. *Biotechnology and Bioengineering*, 84(6):647–657, 2003.

[101] Guido Melzer, Manely Eslahpazir Esfandabadi, Ezequiel Franco-Lara, and Christoph Wittmann. Flux Design: In silico design of cell factories based on correlation of pathway fluxes to desired properties. *BMC Systems Biology*, 3:1–16, 2009.

[102] Ayoun Cho, Hongseok Yun, Jin Hwan Park, Sang Yup Lee, and Sunwon Park. Prediction of novel synthetic pathways for the production of desired chemicals. *BMC Systems Biology*, 4, 2010.

[103] Zachary A. King and Adam M. Feist. Optimizing Cofactor Specificity of Oxidoreductase Enzymes for the Generation of Microbial Production Strains—OptSwap. *Industrial Biotechnology*, 9(4):236–246, 2013.

[104] Priti Pharkya, Anthony P. Burgard, and Costas D. Maranas. OptStrain: A computational framework for redesign of microbial production systems. *Genome Research*, 14:2367–2376, 2004.

[105] Maurice Scheer, Andreas Grote, Antje Chang, Ida Schomburg, Cornelia Munaretto, Michael Rother, Carola Söhngen, Michael Stelzer, Juliane Thiele, and Dietmar Schomburg. BRENDA, the enzyme information system in 2011. *Nucleic Acids Research*, 39(Database):D670–D676, 2011.

[106] Minoru Kanehisa and Susumu Goto. KEGG: Kyoto encyclopedia of genes and genomes. *Nucleic Acids Research*, 28(1):27–30, 2000.

[107] Ron Caspi, Hartmut Foerster, Carol A. Fulcher, Rebecca Hopkinson, John Ingraham, Pallavi Kaipa, Markus Krummenacker, Suzanne Paley, John Pick, Seung Y. Rhee, Christophe Tissier, Peifen Zhang, and Peter D. Karp. MetaCyc: a multiorganism database of metabolic pathways and enzymes. *Nucleic Acids Research*, 34(Database):D511–D516, 2006.

[108] Priti Pharkya and Costas D. Maranas. An optimization framework for identifying reaction activation/inhibition or elimination candidates for overproduction in microbial systems. *Metabolic Engineering*, 8(1):1–13, 2006.

[109] Satoshi Ohno, Hiroshi Shimizu, and Chikara Furusawa. FastPros: Screening of reaction knockout strategies for metabolic engineering. *Bioinformatics*, 30(7):981–987, 2014.

[110] Habib Youssef, Sadiq M. Sait, and Hakim Adiche. Evolutionary algorithms, simulated annealing and tabu search: A comparative study. *Engineering Applications of Artificial Intelligence*, 14(2):167–181, 2001.

[111] Fred Glover. Future Paths for Integer Programming and Links to Artificial Intelligence. *Computers & Operations Research*, 13(5):533–549, 1986.

[112] Xin-She Yang. Review of Metaheuristics and Generalized Evolutionary Walk Algorithm. 3(2):77–84, 2011.

[113] Kenneth Sörensen. Metaheuristics-the metaphor exposed. *International Transactions in Operational Research*, 22(1):3–18, 2015.

[114] Christian Blum and Andrea Roli. Metaheuristics in combinatorial optimization. *ACM Computing Surveys*, 35(3):268–308, 2003.

[115] John H. Holland. *Adaptation in Natural and Artificial Systems: An Introdutory Analysis with Applications to Biology, Control, and Artifical Inteligence*. MIT Press, Cambridge, Massachussets, 1975.

[116] Charles Darwin. *On the Origin of Species by Means of Natural Selection or the Preservation of Favoured Races in the Struggle for Life*. John Murray, London, Uinted Kingdom of Great Britain and Ireland, 1859.

[117] Nils Aall Barricelli. Esempi Numerici di processi di evoluzione. *Methodos*, pages 45–68, 1954.

[118] Lawrence J. Fogel, Alvins J. Owens, and Michael J. Walsh. *Artificial Intelligence Through simulated Evolution*. Wiley, New York, 1966.

[119] David B. Fogel. The Advantages of Evolutionary Computation. *Proceeding Bio-Computing Emergent Computation, World Scientific Press*, (1995):1–11., 1997.

[120] Kiran Raosaheb Patil, Isabel Rocha, Jochen Förster, and Jens Nielsen. Evolutionary programming as a platform for in silico metabolic engineering. *BMC Bioinformatics*, 6:1–12, 2005.

[121] Vassil Guliashki, Hristo Toshev, and Chavdar Korsemov. Survey of Evolutionary Algorithms Used in Multiobjective Optimization. *Problems of Engineering Cybernetics and Robotics*, 60, 2015.

[122] Carlos M. Fonseca and Peter J. Fleming. Genetic Algorithms and Multiobjective Optimization: Formulation, Discussion and Generalization*. *Genetic Algorithms: Proceedings of the Fifth International Conference (S. Forrest, ed.), San Mateo, CA: Morgan Kaufman*, (July):416–423, 1993.

[123] Eckart Zitzler and Lothar Thiele. Multiobjective Optimization Using Evolutionary Algorithms - A Comparative Case Study. *Parallel Problem Solving from Nature - PPSN V Amsterdam, The Netherlands*, pages 292–301, 1998.

[124] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[125] David Hutchison, Juergen Branke, Slowinski Roman, and Kalyanmoy Deb. *Multiobjective Optimization: Interactive and Evolutionary Approaches*, volume 5252 LNCS. 2008.

[126] Wikipedia contributors. Pareto efficiency — Wikipedia, The Free Encyclopedia, 2019.

[127] Anaconda. Anaconda Software Distribution. Computer software. Vers. 2-2.4.0, 2016.

[128] JetBrains. PyCharm 2019.2.3. Build 192.6817.19, 2019.

[129] IBM. ILOG CPLEX Optimization Studio 12.8.0., 2017.

[130] Alberto Noronha, Jennifer Modamio, Yohan Jarosz, Elisabeth Guerard, Nicolas Sompairac, German Preciat, Anna Dröfn Daníelsdóttir, Max Krecke, Diane Merten, Hulda S. Haraldsdóttir, Almut Heinken, Laurent Heirendt, Stefanía Magnúsdóttir, Dmitry A.

Ravcheev, Swagatika Sahoo, Piotr Gawron, Lucia Friscioni, Beatriz Garcia, Mabel Prendergast, Alberto Puente, Mariana Rodrigues, Akansha Roy, Mouss Rouquaya, Luca Wiltgen, Alise Žagare, Elisabeth John, Maren Krueger, Inna Kuperstein, Andrei Zinovyev, Reinhard Schneider, Ronan M. T. Fleming, and Ines Thiele. The Virtual Metabolic Human database: Integrating human and gut microbiome metabolism with nutrition and disease. *Nucleic Acids Research*, 47(D1):D614–D624, 2019.

[131] Almut Heinken, Dmitry A. Ravcheev, Federico Baldini, Laurent Heirendt, Ronan M. T. Fleming, and Ines Thiele. Personalized modeling of the human gut microbiome reveals distinct bile acid deconjugation and biotransformation potential in healthy and IBD individuals. *bioRxiv*, page 229138, 2017.

[132] Adam M. Feist, Christopher S. Henry, Jennifer L. Reed, Markus Krummenacker, Andrew R. Joyce, Peter D. Karp, Linda J. Broadbelt, Vassily Hatzimanikatis, and Bernhard Ø. Palsson. A genome-scale metabolic reconstruction for Escherichia coli K-12 MG1655 that accounts for 1260 ORFs and thermodynamic information. *Molecular Systems Biology*, 3(121):1–18, 2007.

[133] Ana Paula Oliveira, Jens Nielsen, and Jochen Förster. Modeling Lactococcus lactis using a genome-scale flux model. *BMC Microbiology*, 5:1–15, 2005.

[134] Ibrahim E. El-Semman, Fredrik H. Karlsson, Saeed Shoaie, Intawat Nookaew, Taysir H. Soliman, and Jens Nielsen. Genome-scale metabolic reconstructions of Bifidobacterium adolescentis L2-32 and Faecalibacterium prausnitzii A2-165 and their interaction. *BMC Systems Biology*, 8(1):1–11, 2014.

[135] Philippe Goffin, Bert Van De Bunt, Marco Giovane, Johan H. J. Leveau, Sachie Höppener-Ogawa, Bas Teusink, and Jeroen Hugenholtz. Understanding the physiology of Lactobacillus plantarum at zero growth. *Molecular Systems Biology*, 6(413), 2010.

[136] Carla Risso, Jun Sun, Kai Zhuang, Radhakrishnan Mahadevan, Robert DeBoy, Wael Ismail, Susmita Shrivastava, Heather Huot, Sagar Kothari, Sean Daugherty, Olivia Bui, Christophe H. Schilling, Derek R. Lovley, and Barbara A. Methé. Genome-scale comparison and constraint-based metabolic reconstruction of the facultative anaerobic Fe(III)-reducer Rhodoferax ferrireducens. *BMC Genomics*, 10:447, 2009.

[137] Jared T. Broddrick, Benjamin E. Rubin, David G. Welkie, Niu Du, Nathan Mih, Spencer Diamond, Jenny J. Lee, Susan S. Golden, and Bernhard Ø. Palsson. Unique attributes of cyanobacterial metabolism revealed by improved genome-scale metabolic modeling and essential gene analysis. *Proceedings of the National Academy of Sciences of the United States of America*, 113(51):E8344–E8353, 2016.

[138] Joungmin Lee, Hongseok Yun, Adam M. Feist, Bernhard Ø. Palsson, and Sang Yup Lee. Genome-scale reconstruction and in silico analysis of the Clostridium acetobutylicum ATCC 824 metabolic network. *Applied Microbiology and Biotechnology*, 80(5):849–862, 2008.

[139] Nan Xu, Jie Liu, Lianzhong Ai, and Liming Liu. Reconstruction and analysis of the genome-scale metabolic model of Lactobacillus casei LC2W. *Gene*, 554(2):140–147, 2015.

[140] Jun Sun, Shelley A. Haveman, Olivia Bui, Tom R. Fahland, and Derek R. Lovley. Constraint-based modeling analysis of the metabolism of two Pelobacter species. *BMC Systems Biology*, 4(1):174, 2010.

[141] Jennifer Levering, Jared T. Broddrick, Christopher L. Dupont, Graham Peers, Karen Beeri, Joshua Mayers, Alessandra A. Gallina, Andrew E. Allen, Bernhard Ø. Palsson, and Karsten Zengler. Genome-scale model reveals metabolic basis of biomass partitioning in a model diatom. *PLoS ONE*, 11(5):1–22, 2016.

[142] Marco Fondi, Isabel Maida, Elena Perrin, Alessandra Mellera, Stefano Mocali, Ermenegilda Parrilli, Maria Luisa Tutino, Pietro Liò, and Renato Fani. Genome-scale metabolic reconstruction and constraint-based modelling of the Antarctic bacterium Pseudoalteromonas haloplanktis TAC125. *Environmental Microbiology*, 17(3):751–766, 2015.

[143] Monica L. Mo, Bernhard Ø. Palsson, and Markus J. Herrgård. Connecting extracellular metabolomic measurements to intracellular flux states in yeast. *BMC Systems Biology*, 3:1–17, 2009.

[144] Wei Zou, Maoda Zhou, Liming Liu, and Jian Chen. Reconstruction and analysis of the industrial strain Bacillus megaterium WSH002 genome-scale in silico metabolic model. *Journal of Biotechnology*, 164(4):503–509, 2013.

[145] Nadine Veith, Margrete Solheim, Koen W. A. van Grinsven, Brett G. Olivier, Jennifer Levering, Ruth Grosseholz, Jeroen Hugenholtz, Helge Holo, Ingolf Nes, Bas Teusink, and Ursula Kummer. Using a genome-scale metabolic model of Enterococcus faecalis V583 to assess amino acid uptake and its impact on central metabolism. *Applied and Environmental Microbiology*, 81(5):1622–1633, 2015.

[146] Radhakrishnan Mahadevan, Daniel R. Bond, John E. Butler, Abraham Esteve-Nuñez, Maddalena V. Coppi, Bernhard Ø. Palsson, Christophe H. Schilling, and Derek R. Lovley. Characterization of Metabolism in the Fe(III)-Reducing Organism - Geobacter sulfurreducens by Constraint-Based Modeling. *Applied and Environmental Microbiology*, 72(2):1558–1568, 2006.

[147] Rémi Peyraud, Kathrin Schneider, Patrick Kiefer, Stéphane Massou, Julia A. Vorholt, and Jean-Charles Portais. Genome-scale reconstruction and system level investigation of the metabolic network of Methylobacterium extorquens AM1. *BMC Systems Biology*, 5(1):189, 2011.

[148] Anu Raghunathan, Jennifer L. Reed, Sookil Shin, Bernhard Ø. Palsson, and Simon Daefler. Constraint-based analysis of metabolic capacity of Salmonella typhimurium during host-pathogen interaction. *BMC Systems Biology*, 3:1–16, 2009.

[149] Sven E. F. Borgos, Sergio Bordel, Håvard Sletta, Helga Ertesvåg, Øyvind Jakobsen, Per Bruheim, Trond E. Ellingsen, Jens Nielsen, and Svein Valla. Mapping global effects of the anti-sigma factor MucA in Pseudomonas fluorescens SBW25 through genome-scale metabolic modeling. *BMC Systems Biology*, 7:1–15, 2013.

[150] Grigoriy E. Pinchuk, Eric A. Hill, Oleg V. Geydebrekht, Jessica de Ingeniis, Xiaolin Zhang, Andrei Osterman, James H. Scott, Samantha B. Reed, Margaret F. Romine, Allan E. Konopka, Alexander S. Beliaev, Jim K. Fredrickson, and Jennifer L. Reed. Constraint-based model of Shewanella oneidensis MR-1 metabolism: A tool for data analysis and hypothesis generation. *PLoS Computational Biology*, 6(6):1–8, 2010.

[151] Seth B. Roberts, Christopher M. Gowen, J. Paul Brooks, and Stephen S. Fong. Genome-scale metabolic analysis of Clostridium thermocellum for bioethanol production. *BMC Systems Biology*, 4, 2010.

[152] Yohei Shinfuku, Natee Sorpitiporn, Masahiro Sono, Chikara Furusawa, Takashi Hirasawa, and Hiroshi Shimizu. Development and experimental verification of a genome-scale metabolic model for Corynebacterium glutamicum. *Microbial Cell Factories*, 8:1–15, 2009.

[153] Nathan E. Lewis, Kim K. Hixson, Tom M. Conrad, Joshua A. Lerman, Pep Charusanti, Ashoka D. Polpitiya, Joshua N. Adkins, Gunnar Schramm, Samuel O. Purvine, Daniel Lopez-Ferrer, Karl K. Weitz, Roland Eils, Rainer König, Richard D. Smith, and Bernhard Ø. Palsson. Omic data from evolved E. coli are consistent with computed optimal growth from genome-scale models. *Molecular Systems Biology*, 6(390):1–13, 2010.

[154] Olga Ponomarova, Natalia Gabrielli, Daniel C. Sévin, Michael Mülleder, Katharina Zirngibl, Katsiaryna Bulyha, Sergej Andrejev, Eleni Kafkia, Athanasios Typas, Uwe Sauer, Markus Ralser, and Kiran Raosaheb Patil. Yeast Creates a Niche for Symbiotic Lactic Acid Bacteria through Nitrogen Overflow. *Cell Systems*, 5(4):345–357, 2017.