NTNU
Norwegian University of Science and Technology

A Simulated Annealing Based Method for Generating Random Metabolic Networks

Siri Haftun Overn

Biotechnology (5 year)
Submission date: May 2017
Supervisor: Eivind Almaas, IBT

Norwegian University of Science and Technology
Department of Biotechnology and Food Science

Preface

The work presented in this master's thesis was carried out at the Norwegian University of Science and Technology (NTNU). This thesis is a part of the Systems Biology specialization of the five-year M.Sc. program in Biotechnology.

Siri H. Overn

Siri Haftun Overn

Trondheim, May 30, 2017

Acknowledgements

First and foremost, I want to thank my supervisor Professor Eivind Almaas at Department of Biotechnology for his invaluable assistance, patience, support, and encouragement over the past two years. His positivity and motivational words have been crucial in this process, as well as his strict voice each time I just pretended to understand things.

I would also like to thank all the members of the Network Biology group for making our weekly meetings worth to attend. In particular, I would like to express my gratitude to PhD candidate Snorre Sulheim at SINTEF Materials and Chemistry for his help with parallel computing. I had no chance to implement this alone. I also appreciate the assistance from previous group member, constructor of the reaction universe, and PhD candidate Ove Øyaas at ETH Zürich. He helped me finding the right metabolic models for my study. I am also grateful for his quick responses by email each time I thought I had messed things up.

Gratitude is also given to Senior Engineer Sandor Zeestraten and PhD candidate Oddgeir Lingaas Holmen at K.G. Jebsen Centre for their assistance with getting my software up and running on the Almaas Lab computers.

To Mikkel Berg Strømstad: Thank you very much for proofreading this thesis. To all of you who have contributed to five good years in Trondheim: Thank you!

S.H.O.

Abstract

A new method for randomization of metabolic networks was developed. The method was based on a Simulated Annealing algorithm, which is a well known technique for approximating optimal solutions to optimization problems. By modifying the original algorithm to include two temperature parameters instead of one, two different properties of the networks could be optimized at the same time. Compared to the original metabolic networks for microbes, previously generated random metabolic networks have had an extremely high fraction of blocked reactions. This was the motivation to develop a new method.

In this new method, both the total number of reactions and the fraction of blocked reactions in the networks were kept at their optimal levels. In addition to these requirements, the networks should also be viable, which means they should be able to produce biomass. Metabolic networks from three different microbes of varying sizes were chosen for randomization: *Escherichia coli, Mycobacterium tuberculosis*, and *Helicobacter pylori.* The content of these networks was randomized by replacing the metabolic reactions with new reactions from a reaction universe.

Extensive testing of the two temperature parameters established that the method was able to generate random networks with a lower number of blocked reactions while the total number of reactions remained stable. The disadvantage of this method is that it took a very long time to generate new random networks. Nevertheless, it is possible to save huge amounts of time by making some improvements in the algorithm. Suggestions for such improvements are given, but not implemented here.

Although the fraction of blocked reactions was lower than in the previous networks generated, the fraction of essential reactions was virtually unchanged. Visualization of the random networks pointed out that the majority of them had formed multiple clusters of reactions, which explained the unexpected high fraction of essential reactions.

The random metabolic networks were also tested for growth in different growth media. These results indicated that the random networks were only able to produce biomass in the growth medium they were generated in, and otherwise not. This supports that metabolic reactions and pathways that are not in use will be lost in the randomization process.

Sammendrag

En ny metode ble utviklet for randomisering av metabolske nettverk. Denne metoden var basert seg på en standard Simulated Annealing-algoritme, som er en kjent teknikk for å finne tilnærmede optimale løsninger for optimaliseringsproblemer. Ved å modifisere den originale Simulated Annealing-algoritmen slik at den inneholdt to temperaturparametre i stedet for én, kunne flere av egenskapene til nettverkene optimaliseres samtidig. De tilfeldige metabolske nettverkene som er blitt generert tidligere har hatt en veldig høy andel blokkerte reaksjoner sammenlignet med de originale metabolske nettverkene. Dette var motivasjonen for å utvikle en helt ny metode til dette formålet.

I den nye metoden ble både det totale antallet reaksjoner og andelen blokkerte reaksjoner holdt på et optimalt nivå. I tillegg til dette var det også et krav om at de nye nettverkene skulle være levedyktige, som betyr at de skulle være i stand til å produsere biomasse. Metabolske nettverk for tre ulike mikrober av varierende størrelse ble valgt ut: *Escherichia coli, Mycobacterium tuberculosis* og *Helicobacter pylori.* Reaksjonsinnholdet i disse nettverken ble randomisert ved å erstatte de metabolske reaksjonene med nye reaksjoner tatt fra et reaksjonsunivers.

Omfattende testing av de to temperaturparametrene slo fast at metoden var i stand til å generere tilfeldige nettverk med et lavere antall blokkerte reaksjoner, samtidig som det totale antallet reaksjoner ble holdt seg stabilt. Ulempen med denne metoden var derimot at det tok veldig lang tid å generere nye tilfeldige nettverk. Det er imidlertid mulig å spare mye tid ved å gjøre noen forbedringer i algoritmen. Forslag til slike forbedringer er gitt, men ikke implementert her.

Selv om andelen blokkerte reaksjoner var lavere enn i tidligere genererte nettverk, forble andelen essensielle reaksjoner tilnærmet uendret. Visualiseringen av nettverkene viste at de fleste tilfeldige nettverkene som var blitt generert hadde dannet flere reaksjonsklynger, som forklarer den uventede høye andelen essensielle reaksjoner.

De tilfeldige metabolske nettverkene ble også testet i ulike næringsmiljø. Disse resultatene viste at de tilfeldige nettverkene kun var i stand til å produsere biomass i de vekstmediene de var generert i, og ikke ellers. Dette indikerte at metabolske spor og reaksjoner som ikke var aktive i randomiseringsprosessen, forsvant.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Systems biology emerged as an own field for about 15 years ago. Thanks to this, it has become possible to model the properties of biological systems [1]. An example of such a system is a metabolic network, which is illustrated at the front of this thesis. Metabolic networks include the metabolic reactions and pathways that are catalyzed by enzymes encoded in the organism's genome. When such networks are reconstructed into a mathematical model, they are called genome-scale metabolic models [2].

Here, an introduction to systems biology and metabolic networks is given together with some applications for genome-scale metabolic models. Due to the purpose of this thesis, a brief introduction to optimization theory will also be provided.

1.1 Systems Biology

Being able to predict a cellular function from a given genotype is a fundamental goal in biology. Due to the development of powerful molecular biology tools in the latter half of the 20th century, the reductionist approaches dominated the practice of biology and the genotype-phenotype relationship was explained from individual components [1, 3, 4]. Reductionism is the idea of breaking problems down to its individual parts and understand them in detail [5, 6]. These approaches provided detailed understanding of biological components, but missed out the systemic interactions between biological and environmental components that underlie phenotypes [1].

At the beginning of the 21st century, a paradigm shift in biology was unfolding [2, 5, 7]. The development of high throughput technologies forced biologists to stop focusing on individual cellular components and rather begin to view cells as systems [2, 8, 9, 10]. The first sequenced genome in 1995 and the birth of functional genomics made large quantities of high quality data available [9]. Microarrays allowed high throughput screening of cellular components, which revealed the components' status at any given time. Protein chips and yeast two-hybrid screens helped to determine how these components interacted with each other. Different types of biochemical networks emerged from these interactions, e.g. protein-protein interaction networks, metabolic networks, and signalling networks [11]. As a consequence of new technologies and the collection of comprehensive data sets, systems biology emerged as a new field [2, 8].

Systems biology spans many areas of modern science, ranging from biology and chemistry to mathematics and computing. This field is using a holistic approach instead of reductionist, which considers biological systems as a whole rather than the sum of its components [5]. One of the aims of systems biology is modelling and analyzing cellular functions as a system and understanding the interactions between biological components [3, 5, 9, 12, 13]. The development of computational methods was important for understanding these interactions in biochemical networks [8, 12].

1.2 Metabolic Engineering

Metabolic engineering is defined as the directed improvement of cellular properties through the modification of specific biochemical reactions or introduction of new ones [14]. The vision of defining metabolic engineering as an own discipline was first suggested by Bailey in 1991 [15]. Since then, the strategy has been used in many areas, production of e.g. amino acids, antibiotics, and vitamins. The development of molecular biological techniques for bringing together DNA material from different sources was crucial for the emergence metabolic engineering [14]. Some of the applications of metabolic engineering are improvement of the yield and productivity of native products synthesized by microorganisms [7, 16] and production of new products [14].

Cellular functions rely on multiple gene products [9]. One of the important aspects of metabolic engineering is its emphasis of the interactions between biochemical reactions instead of individual enzymatic reactions [10, 14]. The complete set of metabolic reactions, metabolic pathways, and their regulatory interactions is called a cell's metabolic network. Here, nodes represent the metabolites and the links between them illustrate their interactions. In metabolic engineering, the most significant contribution has been the study of movement of matter through metabolic networks, also called metabolic fluxes. Metabolic fluxes represent the reaction rates in metabolic pathways and carry valuable information about a cell's metabolism. They describe the activities of metabolic networks and the metabolic state of a cell [10, 14, 17]. Because of this, metabolic flux was stated as the most fundamental measure of cell physiology already in the 1990s [18].

1.3 Antibiotic Targets

Horizontal gene transfer is a frequent process in prokaryotes, where genetic material is moved from one organism to another. This leads to new reactions being introduced, which over time will cause metabolic networks to differ in their reaction content. Horizontal gene transfer is therefore considered as an important factor of the evolution of metabolism [19, 20, 21].

Many antibiotics target the metabolism of pathogens and are therefore called antimetabolic drugs [22]. Resistance to antimetabolic drugs is developed through *de novo* mutation or horizontal gene transfer. In the latter case, the acquisition of genes from other organisms may enable the pathogens to produce enzymes that destroy the drug, proteins that transport the drug out of the cell, or alternative metabolic pathways that bypass the target of the antimetabolic drug [19, 23].

Antibiotic resistance has emerged as a serious threat to human health over the past few decades [24]. Enormous investments are needed to develop new and better antibiotic drugs [19, 25]. An ideal enzymatic drug should inhibit enzymes that are absolutely necessary for the organism to live, so-called essential enzymes. However, horizontal gene transfer is able to introduce new reactions and enzymes that bypass the essential ones, which makes them non-essential. Enzymatic drugs are therefore vulnerable to antibiotic resistance, unless it is possible to find reactions and enzymes that are essential in absolutely all metabolic networks. To find these, extremely many metabolic networks need to be tested for essential reactions in different growth media [19].

1.4 Genome-Scale Modelling

The availability of complete genome sequences and the ability to determine when a cell expresses particular genes provide an opportunity to develop metabolic models on a genomic scale [2, 7, 9, 10]. These network reconstructions contain all of the known metabolic reactions in an organism together with the genes that encode each enzyme [26]. The first genome-scale metabolic network was reconstructed in 1999 [27]. Since then, genome-scale metabolic models have been constructed for multiple organisms [10, 17, 28]. The genome-scale metabolic models have shown some promise in connecting genotype with phenotype and predict phenotypic changes as a results of genetic modification [7, 9, 10, 29]. Metabolic phenotypes can be predicted by observing how the cell is affected by external changes, i.e. changes to its environment [2, 5, 7].

To understand metabolic networks and be able to study the metabolic capability in a particular condition, modeling and simulation play an important role [10, 29]. A number of approaches have been developed in order to predict how metabolic fluxes are distributed, which is critical information in metabolic engineering [30]. Analysis of genome-scale metabolic models has allowed researchers to understand the metabolic status in a cell and improve the production of desired products [10, 17]. Some models have been used for the identification of metabolic engineering targets for increased production of amino acids [31, 32] and identification of drug targets in pathogens [10, 33].

The growth of computational approaches and the increased availability of high quality metabolic reconstructions have made the *in silico* analysis of metabolism to an invaluable part of systems biology [34].

1.5 Optimization Techniques

Since the industrial revolution, the size and complexity of organizations have increased a lot. Along with all its blessing, the increased specialization has created new problems that are still current issues. When the complexity increases, it becomes more and more difficult to allocate resources to the various activities. Problems like this and the attempt to solve them led to the emergence of a new field called Operations Research (OR) [35, 36].

Although OR emerged around the late 1700s, good methods for solving such problems did not exist until World War II. The British and U.S. military management asked a large number of scientist, the worlds first OR team, to apply a scientific approach to allocate scarce resources to various military operations in an effective manner. Their effective methods of how to manage convoy and antisubmarine operations played a major role in winning the Battle of the North Atlantic [35].

When the war ended, an industrial boom followed. There was great interest in the OR methods, especially when people realized that there were many similarities between the problems caused by the increased complexity in organizations and the problems the OR team solved during the war. Before the end of the 1950s, many of the current standard tools of OR, such as the well known Simplex method, was developed [35].

The development of electronic digital computers in the 1980s was another huge step for OR. Today, all computers are able to solve OR problems, even small laptops. Most people are more familiar with the term *optimization theory*, which has mostly replaced the use of OR over the years [35, 36].

Today, optimization theory is a large area of applied mathematics and has found a widespread use in economics, business, and engineering. Optimization problems exist within any field, and efforts are continuously being made to find ways to solve new problems or to streamline already existing methods [35].

1.6 Thesis Objective

The main focus of this project is to develop a **new method for generating random viable metabolic networks**. Random metabolic networks are generated by replacing and randomizing their reaction content. The requirement that they must be viable means they should be able to produce biomass in a prescribed chemical environment, thus emulating a normal metabolic network.

A key question is how it is biologically relevant to change the reaction content in metabolic networks and making random ones. The answer is to investigate the new phenotypes of such random networks. Changing the reaction content is a way of simulating the evolution of metabolism, which is a central part of metabolic engineering. Varying the reaction content simulates the loss and acquisition of enzyme-coding genes, while the organism's ability to synthesize biomass is preserved. By analyzing a large number of random networks it is also possible to study which reactions are always essential in that organism's metabolic network. Reactions that are always essential will most likely not be bypassed by new reactions, and will therefore be a good target for antimetabolic drugs.

Previously, random viable metabolic networks have been generated by a method based on a Markov Chain Monte Carlo (MCMC) algorithm that was developed by the Andreas Wagner Laboratory at the University of Zürich [19]. Later, it was used by Ove Øyaas in his master thesis at NTNU [37]. He discovered that the networks had a very large fraction of blocked reactions compared to the original networks. The percentage of blocked reactions is usually around 30 % in already existing networks, but routinely reached 70 % in the networks generated by the Wagner method. This means that the vast majority of the reactions in the networks were not connected, and thus, could not carry flux.

Another relevant question that should be asked is why the Wagner method is not adapted to lower the number of blocked reactions instead of developing a completely new method. The answer to this is that it is not entirely in line with metabolic evolution that the number of reactions is kept at the same level all the time, which is the case for the Wagner method. A new method that allows some fluctuations in the number of reactions would therefore be much more realistic.

The method in this project is based on a procedure developed in the field of physics called Simulated Annealing. Simulated Annealing accepts major fluctuations but also guarantees an optimal solution if implemented correctly. It was first described by Kirkpatrick *et al.* [38] in 1983, and has been used to solve many optimization problems since.

Consequently, this thesis provides a new method based on Simulated Annealing for generating random viable metabolic networks with a controllable fraction of blocked reactions. To test this method, random metabolic networks have to be generated to see how the number of blocked reactions affects the other properties of the networks.

Chapter 2

Theory and Literature Review

In this chapter, theory and research relevant for this thesis is presented. The concepts of linear programming will be explained, focusing on problem definition and the simplex method. Another optimization technique relevant for this thesis is the Simulated Annealing algorithm, which will be presented in detail. Due to later use, a well known optimization problem from the field of physics called the Ising model will also be introduced here. When these optimization techniques have been explained, genome-scale metabolic reconstructions will be described together with a method that uses linear programming to obtain optimal solutions for such metabolic models. Finally, robustness in metabolic networks and an introduction to random viable metabolic networks will be given. All of these topics are relevant for this thesis, either for developing the new method or for analyzing the resulting metabolic networks.

2.1 Linear Programming

Linear programming (LP) is a method used to solve optimization problems where the requirements can be formulated as linear relationships. The most common application type of LP is the allocation of scarce resources among competing activities in an optimal way. Definition of LP problems and solving these with use of the simplex method will be presented in this section, all based on Hillier and Lieberman [35].

2.1.1 Problem Definition

An LP problem is defined by determining the appropriate objectives and the restrictions on what can be done. To make the problem convenient for analysis, the problem should be represented as a mathematical model with linear functions. One of the two standard ways to express an LP problem is:

$$\text{maximize} \quad z = \sum_{j=1}^{n} c_j x_j \tag{2.1}$$

$$\text{subject to} \quad \sum_{j=1}^{n} a_{ij} x_j = b_i, \quad i = 1,\dots,m \tag{2.2}$$

$$\text{and} \quad x_j \geq 0, \quad j = 1,\dots,n \tag{2.3}$$

where n is the number of variables and m is the number of constraints. Equation 2.1 is the function that specifies the problem's objective, which in this case is to maximize the objective value z. c_j is the model's objective coefficients for the variable x_j. The constraints in the model are given in Equation 2.2, often defined as the problem's functional constraints. In the i^{th} constraint, a_{ij} is the coefficient value for the variable x_j. All constraints also involve a right hand side which is denoted b_i. All coefficients and constants in an LP model are known data. Equation 2.3 defines the allowed values for the decision variable x_j, often referred to as the non-negativity constraints.

LP problems can also be expressed in a matrix form. The general programming problem presented above will look like this in matrix notation:

$$\text{maximize} \quad \mathrm{Z} = \mathbf{c}^{\mathrm{T}}\mathbf{x} \tag{2.4}$$

$$\text{subject to} \quad \mathbf{Ax} \leq \mathbf{b} \tag{2.5}$$

$$\text{and} \quad \mathbf{x} \geq 0 \tag{2.6}$$

Here, Equation 2.4 is the model's objective function to be maximized. $\mathbf{c}^{\mathrm{T}}$ is the transposed $n \times 1$ vector of the objective coefficients corresponding to the values of c_j in Equation 2.1. $\mathbf{x}$ is a $n \times 1$ vector of decision variables. $\mathbf{A}$ in Equation 2.5 is the matrix of coefficients and $\mathbf{b}$ is a $m \times 1$ vector of the right hand side constants.

The functionality constraints and non-negativity constraints in an LP model define the prob-

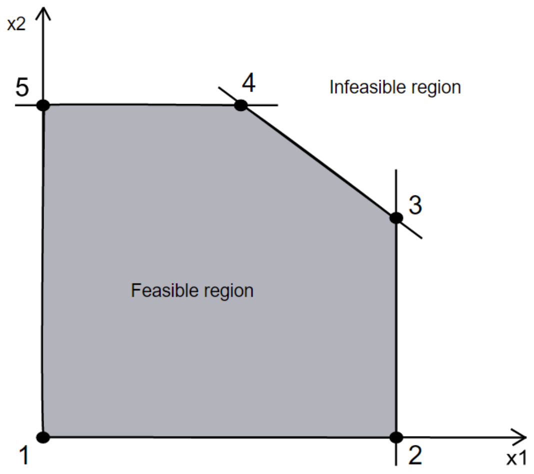


Figure 2.1: The feasible region of an LP problem with the two decision variables x_1 and x_2. The feasible region is defined by the model's constraints that formulate the edges surrounding the gray area. The black dots correspond to the extreme points for the LP problem, which are potential optimal solutions to the problem. The enumeration of vertices correspond to a suggested order for solving the LP problem, which is explained in Section 2.1.4.

lem's feasible region, which is illustrated in Figure 2.1. The feasible region of a problem has as many dimensions as there are decision variables in the model.

Any set of values for the decision variables corresponds to a solution to the LP problem, even though the values are not allowed considering the problem's constraints. It is thus necessary to define different types of solutions. If a solution violates at least one of the constraints, that solution is said to be infeasible. For feasible solutions, all constraints in the model are satisfied. An optimal solution is the feasible solution that has the most favorable value of the objective function, i.e. the largest value possible for maximization problems and the smallest value possible for minimization problems.

2.1.2 Solutions to LP Problems

LP problems can have zero, one, or an infinite number of optimal solutions. The optimal solution for an LP problem is always obtained at the boundary of the feasible region, which is beneficial when the solving the problem. This means that if the problem has exactly one optimal solution, the solution must be in one of the corners of the feasible region. Corner points are often referred to as extreme points. If the problem has multiple optimal solutions, all of them

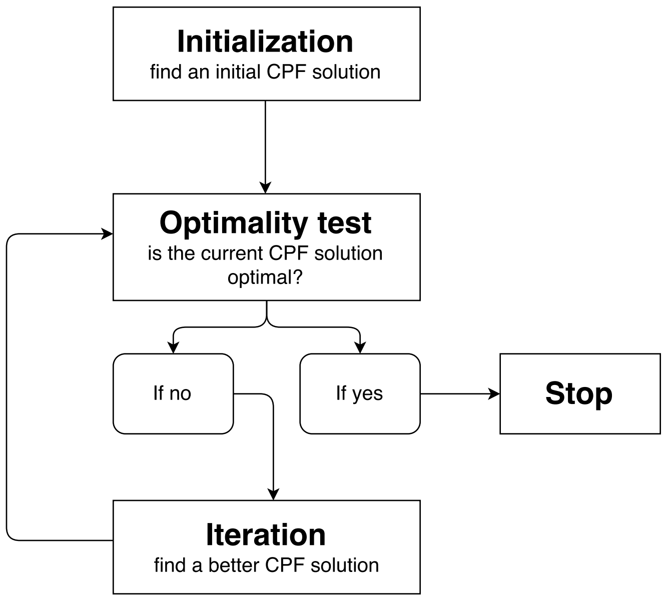

Initialization
find an initial CPF solution
Optimality test
is the current CPF solution optimal?
If no
If yes
Stop
Iteration
find a better CPF solution

Figure 2.2: The concept of the Simplex method presented in a flow chart. The optimality test is repeated for the best CPF solution in each iteration. The figure is adapted from Hillier and Lieberman [35].

lie along the edge between two extreme points. If the objective is unbounded or there does not exist any feasible solutions, there will be no optimal solutions for the problem. An unbounded objective means that the feasible region is not closed, implying that the objective value Z can have infinitely high or low values.

2.1.3 Solving LP Problems

A consequence of that the optimal solutions of LP problems always lie in one of the extreme points is that only the corners have to be investigated when searching for the best solution. Solutions found in the extreme points of a feasible region are referred to as corner-point feasible (CPF) solutions. One of the most common procedures for solving LP problems is the simplex method. This method repeats a series of steps until the optimal solution is obtained. Each of these series is called an iteration, which is illustrated in the flow chart in Figure 2.2. The procedure for the simplex method is as follows:

1. *Initialization.* In the initialization process, a random CPF solution must be chosen. When possible, the origin is picked as initial CPF solution. If the origin is infeasible for the problem, there exists special procedures to find another initial CPF solution. The simplex al-

gorithm considers the CPF solution's neighbors when moving from one possible solution to the next.

2. *Optimality test.* The rate of improvement in the objective must be computed to determine the direction when moving between different CPF solutions. A positive rate of improvement in Z implies that the CPF solution in the other end of that edge is better than the current one. Similarly, a negative rate of improvement means that the corresponding CPF solution is worse.

3. *Iterate*: The algorithm stops if no better CPF solution is found among the adjacent ones, and hence the current CPF is the optimal solution for the LP problem. If the optimality test finds a better CPF solution, the algorithm continues to this solution. Repeat from step 2.

2.1.4 Graphical Example of the Simplex Method

Even though the simplex method is an algebraic procedure, the concepts are geometric. When the method is explained, it is often easier to illustrate it graphically. In Figure 2.1, the simplex method would start by picking the origin as the initial CPF solution, and then examine which one of the associated edges that gives the largest rate of improvement of Z. If the objective is to maximize Z and the objective function is $\mathrm{Z} = 4x_1 + 2x_2$, the edge that lies along the x_1 axis would give the largest rate of improvement in Z. The CPF solution marked 2 will be the new current solution, and the procedure will continue counterclockwise around the gray area until the optimal CPF solution is found and the algorithm stops.

2.2 Simulated Annealing

Simulated annealing (SA) is an optimization technique that belongs to a class called Local Search Algorithms (LSA) [39, 40, 41]. SA has several positive features: Its implementation is quite easy compared to other optimization algorithms, and statistically it can promise to deliver an optimal solution [40].

This section will explain the underlying concepts of LSA, give insight to how SA is able to solve optimization problems, and introduce an example of a well known problem from physics that can be solved using SA.

2.2.1 Local Search Algorithms

As its name suggests, LSA is a class of methods that use various search schemes to find an optimal solution. The search is guided by a neighborhood function, which is an essential part of these algorithms. The neighborhood N of solution i, $N(i)$, is defined as a set of solutions that are in some sense close to i. LSA is an iterative algorithm that explores the neighborhood $N(i)$ for a better solution than the current one, i. If a better solution is found, solution i is replaced by this one. If a better solution cannot be found in $N(i)$, the procedure stops. Often, the search process is terminated in a local optimum due to this neighborhood function. Local optima are not the problem's optimal solution, but LSA is not allowed to continue because none of the adjacent solutions is better [42, 43]. A local optimum is illustrated in Figure 2.3.

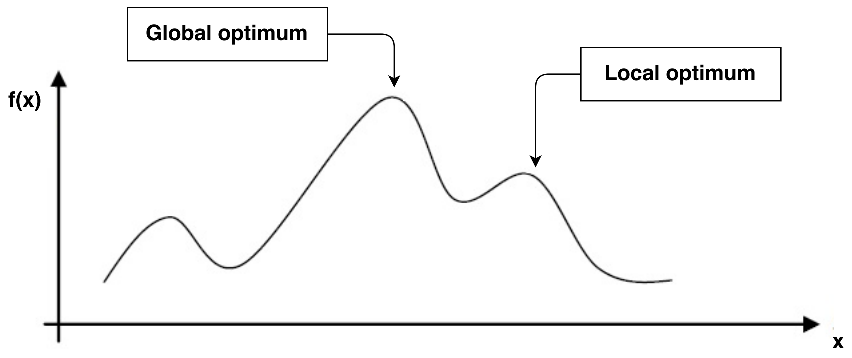


Figure 2.3: An illustration of the global and local optimum for a maximization problem. LSAs will often terminate in such local optima because there are no better solutions in their neighborhood. The figure is adapted from Burke, Newall and Weare [44].

.

2.2.2 Basic SA and the Metropolis Algorithm

The concepts of SA were introduced by Kirkpatrick *et al.* [38] in the early 1980's . The method was inspired by an analogy between physical annealing of solids and solving large optimization problems [41, 42]. The physical annealing process includes increasing the temperature at which the solid melts and slowly decreasing the temperature until the particles arrange themselves in an ordered low-energy structure [38, 41]. If the temperature is lowered too fast, the solid will freeze in a non-optimal structure where the particles are still arranged randomly and the corresponding energy is not minimal [42]. This annealing process of melted solids is simulated by a procedure known as the Metropolis algorithm [41, 45]:

1. Find an initial solution i with corresponding energy E_i. Set the temperature parameter T to a high value to simulate the melting process.

2. Generate a new state j from i, which is in the neighborhood of solution i. This is done by random perturbations, for example by displacement of one particle. Compute the energy E_j corresponding to solution j.

3. Compute the energy difference between the two states, $\mathrm{E}_j - \mathrm{E}_i$. State j is accepted if the difference is less than or equal to zero. If the difference is greater than zero, state j is accepted with a probability given by:

$$\mathrm{P}(\text{accept } j) = \exp\left(\frac{\mathrm{E}_j - \mathrm{E}_i}{k_B \mathrm{T}}\right) \tag{2.7}$$

4. After N iterations of step 2 and 3, lower the temperature and return to step 2.

Equation 2.7 is known as the Metropolis criterion, which is a probability proportional to the energy difference and the current temperature [41, 45]. k_B is a physical constant named the Boltzmann constant, which is related to the kinetic energy in particles. If the number of perturbations at each temperature is equal to the number of particles in the system (N), thermal equilibrium is reached [42].

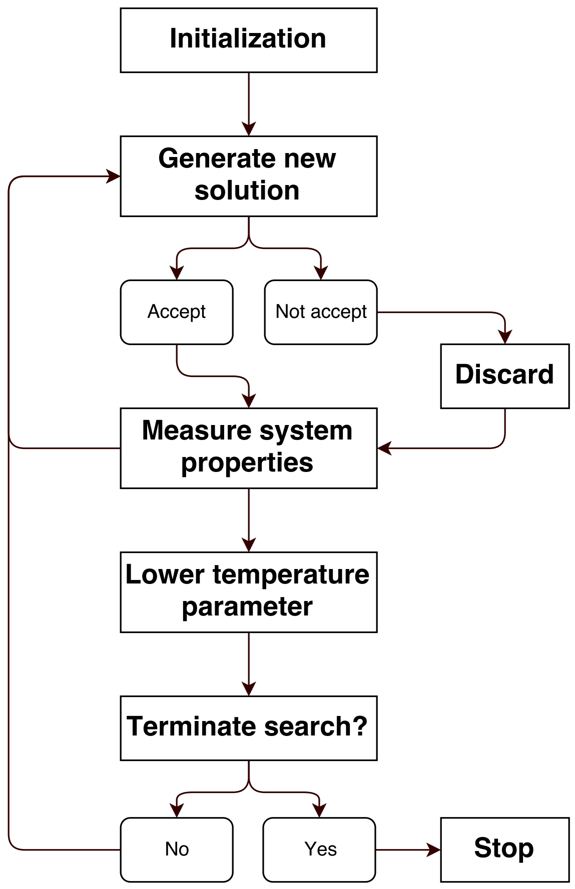


Figure 2.4: An overview of the SA algorithm. This is an iterative method that explores the whole solution space. A predefined number of iterations is repeated before the temperature parameter is lowered. When the temperature parameter reaches a desired value, the search terminates, and the optimal solution is found.

2.2.3 SA as an Optimization Method

By the following assumptions, the Metropolis algorithm can be used to solve general optimization problems [42]:

1. The physical states correspond to solutions in the optimization problem.

2. The energy E of a state corresponds to the objective value Z of a solution.

Figure 2.4 illustrates how the SA algorithm solves optimization problems. The same acceptance rule is used in SA as in the Metropolis algorithm. Instead of the Boltzmann constant and the actual temperature T that is used in in Equation 2.7, an artificial temperature parameter is in-

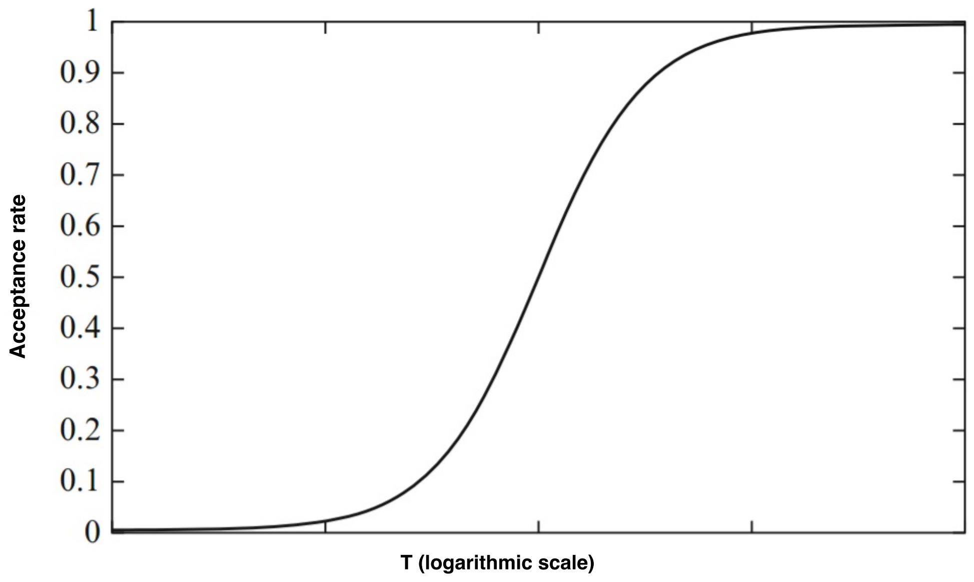
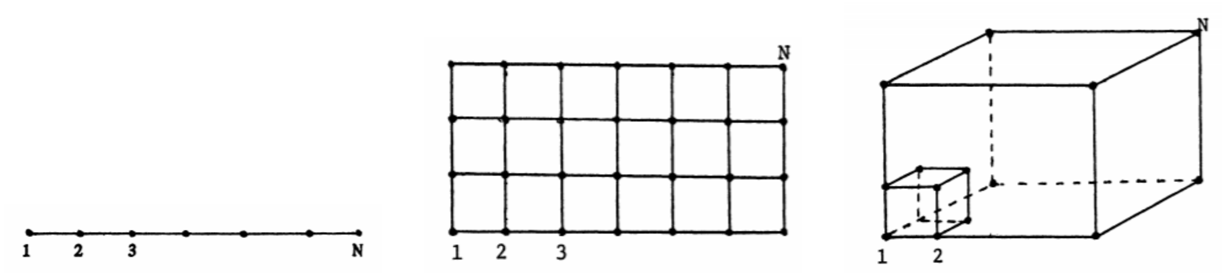


Figure 2.5: The acceptance rate of SA as a function of the temperature parameter T. For low Ts, almost none of the new solutions will be accepted. When T is high, approximately 100 % of the new solutions are accepted. The figure is adapted from Aarts, Korst and Michiels [42].

troduced [42, 46]. The probability of retaining a worse solution than the previous one then becomes:

$$\mathrm{P(accept\ new\ solution)} = \exp\left(-\frac{\Delta Z}{T}\right) \tag{2.8}$$

where ΔZ is the difference in objective value between the two solutions. When the control parameter T is high, solutions far from the optimum are explored and deteriorations have a notable probability of being retained. The acceptance rule will become more stringent as the parameter T is lowered, which effectively reduces the probability of retaining worse solutions. When the control parameter approaches zero, only solutions that are better than the current one will be accepted [41, 42]. The acceptance rate as a function of T is shown in Figure 2.5.



(a) one-dimensional (b) two-dimensional (c) three-dimensional

Figure 2.6: Examples of the Ising model in one, two, and three dimensions. a) One-dimensional Ising model. The sites 1 to N are arranged in a chain. Each spin σ has two neighbors: one at the left side and one at the right side. By definition σ_1 and σ_N are neighbors. b) Two-dimensional Ising model. The sites are arranged in a lattice where each spin has four neighbors. This dimension will be explained in more detail later. c) Three-dimensional Ising model. This will not be explained any further here because of its complexity, and is viewed just as an example of an Ising model with higher dimensions than two. The figures are obtained from Cipra [49].

2.2.4 The Ising Model

The Ising model was introduced by Ising [47] in 1925 and is a mathematical model for ferromagnetism. The Ising model can have a variety of dimensions, as illustrated in Figure 2.6. The idea of this model is a lattice of "spin" variables, σ_k, that can take the values +1 or -1 only, where k refers to a site in the lattice. The values +1 and -1 are referred to as a site's configuration. The Ising model is a well known optimization problem from the field of physics that can be solved by using SA [48].

One-Dimensional Ising Model

The one-dimensional Ising model is the easiest to explain because each spin has only two neighbors. As shown in Figure 2.6, all spins are arranged in a chain.

The energy corresponding to configuration σ can be expressed as:

$$E = -\sum_{<i,j>} J_{ij}\sigma_i\sigma_j - \mu\sum_j H\sigma_j \tag{2.9}$$

where i and j are adjacent sites and J_{ij} is the interaction between them. If σ_i and σ_j have opposite configurations, the interaction energy is $+J$. If they have the same configuration, the interaction energy is $-J$. H is the external magnetic field and μ is the magnetic moment. For

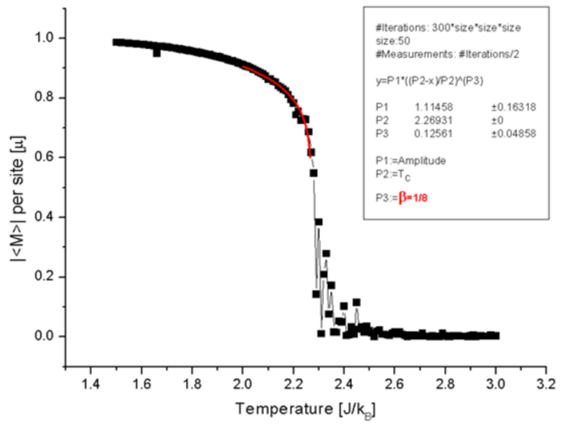


Figure 2.7: The average magnetization in the Ising model as a function of the temperature. When the temperature is high, the spins are randomly aligned resulting in an average magnetization of zero. When the temperature is lowered, the average magnetization approaches one. The figure is obtained from Witthauer and Dieterle [50].

simplicity, the external magnetic field is often set to zero thus eliminating the last sum.

A typical quantity of interest when working with Ising models is the average magnetization in the system, given by:

$$M = \frac{1}{N}\sum_{i=1}\sigma_i \tag{2.10}$$

where N is the total number of spins in the system.

The optimal solution can be found by using SA to minimize the total energy in the system. After the chain of randomly aligned spins have been generated, the initial temperature have been set to a high value, and the initial starting energy and magnetization have been calculated, the simulation can start. One spin at a time is flipped. If the new energy is lower than the previous, the new solution will always be accepted. If the new energy is higher, the corresponding solution will be accepted with a probability given in Equation 2.8. As the temperature is lowered, the energy in the system will also become lower. When finished, all spins have the same configuration.

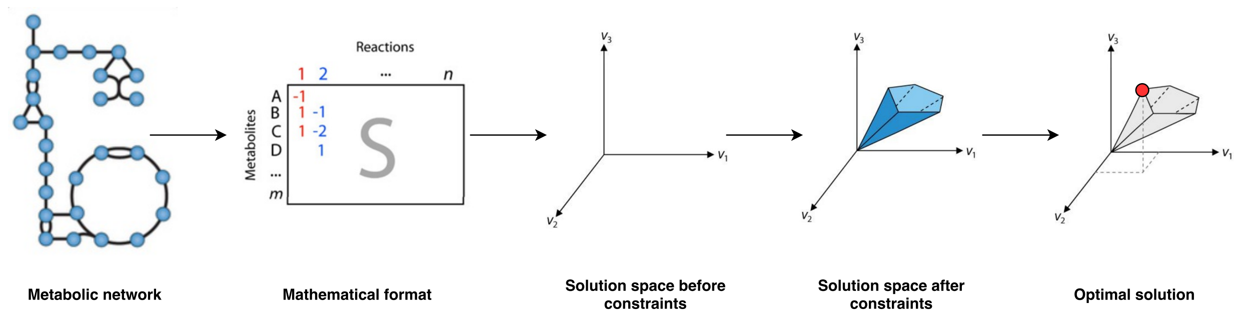


Figure 2.8: An overview of constraint-based modelling. In the reconstruction process, the metabolic network is converted to a mathematical format known as the stoichiometric matrix. This conversion is necessary in order to make computations on the network. The flow of metabolites is controlled by internal and external constraints. These constraints "shrink" the solution space, previously referred to as the feasible region, to a more biological relevant size. LP is used to find an optimal solution based on this solution space and the objective function. This figure is adapted from Bordbar *et al.* [1] and Orth, Thiele and Palsson [26].

2.3 Constraint-Based Reconstruction and Analysis

The ability to sequence whole genomes has made it possible to formulate metabolic models at genome scale. Constraint-based reconstruction and analysis (COBRA) is the reconstruction of metabolic networks from annotated genomes and experimental literature as well as analysis of their metabolic functions [1]. This section includes an explanation of the reconstruction process, the applications of genome-scale metabolic networks, how the reconstructions are converted into a mathematical format for computation, and optimization of biological phenotypes using flux-balance analysis (FBA). The reconstruction, mathematical representation, and the concept of FBA are summarized in Figure 2.8.

2.3.1 Genome-Scale Metabolic Reconstructions

Metabolic networks differ from other cellular networks in that their interaction topology is well established. Most reactions, how they interact stoichiometrically, the enzymes that catalyze them, and the genes that encode the enzymes are well known [1, 12, 26]. This is far beyond any other cellular network and makes it possible to create metabolic models representing nearly entire microbial genomes [12, 29].

A genome-scale metabolic reconstruction contains all the metabolic reactions known to take place in an organism. Metabolic pathways represented as a network is illustrated in Figure 2.9.

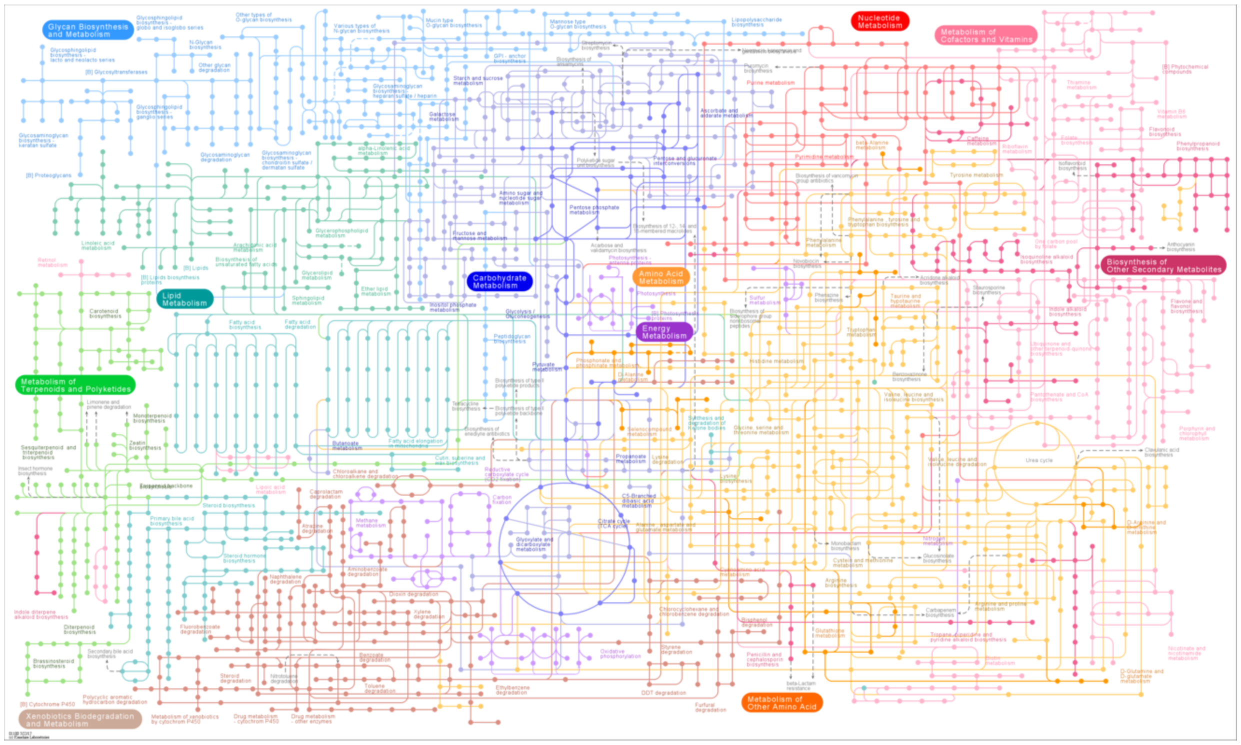

Glycan Biosynthesis and Metabolism
Nucleotide Metabolism
Metabolism of Cofactors and Vitamins
Carbohydrate Metabolism
Lipid Metabolism
Amino Acid Metabolism
Energy Metabolism
Biosynthesis of Other Secondary Metabolites
Metabolism of Terpenoids and Polyketides
Xenobiotics Biodegradation and Metabolism
Metabolism of Other Amino Acid

Figure 2.9: An overview of metabolic pathways found in the KEGG PATHWAYS illustrated as a metabolic network. The figure is obtained from the KEGG API [52].

The number of network reconstructions is rapidly increasing and the range of their usage is broadening [28, 51]. The following subsections will take a closer look at the reconstruction process and the applications of genome-scale metabolic models.

The Reconstruction Process

Genome-scale metabolic reconstructions represent structured information of all the biological transformations taking place in a specific organism. By converting genome-scale metabolic reconstructions into mathematical models, the metabolic capabilities can be investigated using computational methods [51, 53]. The reconstruction process is an iterative procedure that relies on both automatic and manual collection of organism-specific information [53]. Thiele and Palsson [53] have designed a detailed protocol for generating a high quality genome-scale metabolic reconstruction. This procedure comprises four stages:

1. *Creating a draft reconstruction.* The metabolic genes are automatically retrieved from the genome annotation of the target organism [51, 53]. Biochemical reaction databases are

used to find the metabolic reactions catalyzed by the enzymes encoded in the genome. The draft reconstruction is a list of all metabolic reactions found here. Because of these automatic procedures for collecting information, some of the metabolic functions may be falsely included or missing [53].

2. *Manual reconstruction refinement.* The draft reconstruction needs to be re-evaluated and improved. This correction process is done manually and is therefore very time-consuming [51]. Due to potential errors related to genome annotation and the organism unspecific biochemical databases used in stage one, the presence of each annotated gene and its function should be supported by experimental data or organism-specific literature [53]. By manual curation, network gaps are filled and information that is mistakenly included or misplaced is removed [51]. In addition to the metabolic reactions, the organism's biomass reaction is connected to the reconstruction. This reaction is needed for growth simulations and accounts for all the biomass constituents, which is derived from experimental data, and how they contribute to the overall cellular biomass [53].

3. *Conversion to a mathematical model.* The network reconstruction must be converted to a mathematical format before computations of the network properties can be done [54, 55]. This conversion makes a true genome-scale metabolic model [51]. Computational tools can analyze network properties and evaluate which functions a reconstructed network can carry out and perform under the physiochemical constraints placed on the cell [51]. This approach has led to the constraint-based reconstruction and analysis (COBRA) framework [56].

4. *Network evaluation.* The last stage in the process is the final debugging stage where the network is verified and validated. The mathematical model made in the third step is tested for its ability to synthesize metabolic precursors, which may reveal areas of disagreement between the model's behavior and experimental observations. These gaps are filled by repeating steps two and three until model predictions are similar to the phenotypic [53].

Applications of Genome-Scale Metabolic Models

In 1999, the first genome-scale metabolic reconstruction was published [27]. Since then, this field has expanded rapidly and genome-scale metabolic reconstructions have shown a growing influence on biomedical and biological research. The focus is therefore shifting from the development of its methods to the development of its applications. The application areas for genome-scale metabolic models are [3]:

1. *Contextualization of high-throughput data.* Genome-scale metabolic models can be used as a framework on which other data types can be overlaid to organize and contextualize them. For example, gene microarray data overlaid on genome-scale metabolic reconstructions can be used to determine condition-dependent cell phenotypes.

2. *Guidance of metabolic engineering.* Traditional metabolic engineering has been performed on a small scale based on literature-derived metabolic pathway maps or intuitive knowledge. The use of genome-scale metabolic models makes it possible to perform system-level analyses to determine optimal engineering strategies, for example for production of cellular compounds.

3. *Directing hypothesis-driven discovery.* Genome-scale metabolic models represent collections of existing hypotheses. Taken together as a broad context they enable systematic identification of new hypotheses that can be tested and resolved.

4. *Interrogation of multi-species relationships.* Comparative network-level analysis either compares metabolic reconstructions of highly related species or use models of interacting species to predict common phenotypes.

5. *Network property discovery.* One of the most direct contributions of metabolic models to the understanding of metabolism has been its enabling to study otherwise inaccessible network properties using computational methods. Examples of network properties are: loops, optimal pathway usage, and metabolite connectivity.

2.3.2 The Stoichiometric Matrix

As already mentioned, a mathematical representation is crucial for analyzing the metabolic network using computational methods. The metabolic network is automatically translated into a stoichiometric matrix, often referred to as an S-matrix, in order to apply FBA [7, 12, 26]. Each row in the matrix corresponds to a metabolite and each column corresponds to a reaction. The entries in each column corresponds to the stoichiometric coefficients of the metabolites in the given reaction. A positive coefficient means that the metabolite is produced in that reaction and a negative coefficient implies a consumed metabolite. If a metabolite does not participate in the reaction, the coefficient is zero [26, 51].

A general S-matrix has the dimension $m \times n$:

$$\mathbf{S} = \begin{pmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,n} \\ c_{2,1} & c_{2,2} & \cdots & c_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m,1} & c_{m,2} & \cdots & c_{m,n} \end{pmatrix} \tag{2.11}$$

where m is the number of metabolites, n is the number of reactions, and c_{ij} is the stoichiometric coefficient of metabolite i in reaction j [7, 51].

2.3.3 Steady State Mass Balance

The concentration of each of the m metabolites in the system is expressed in a concentration vector $\mathbf{x}$. Assuming mass balance at steady state ($\mathrm{d}\mathbf{x}/\mathrm{d}t = 0$), the total amount of the m metabolites being produced is equal to the amount of metabolites being consumed [7, 12, 26]. The system of mass balance equations is given by:

$$\mathbf{Sv} = \mathbf{0} \tag{2.12}$$

where $\mathbf{S}$ is the S-matrix in Equation 2.11 and the flux vector $\mathbf{v}$ corresponds to the movement of metabolites through the n reactions [1]. This equation is fundamental to constraint-based modeling [12].

2.3.4 Flux-Balance Analysis

FBA was the first constraint-based method for biological predictions. It is a widely used mathematical approach for harnessing the knowledge encoded in biochemical networks, in particular the genome-scale metabolic network reconstructions [1, 7, 26]. Successful FBA includes prediction of optimal metabolic states, among other network capabilities [12, 57]. By calculating the flow of metabolites through metabolic networks it is possible to predict the growth rate of an organism [3, 7, 26, 28].

Constraint-based modeling with FBA allows steady-state analysis without large fitted parameter sets, as opposed to dynamic models that often require detailed kinetic parameters, which are often inaccessible [7, 8, 12, 29, 58]. This is possible due to the assumption that networks modeled at steady state is able to yield valuable information [3]. Predictions of FBA are in good agreement with experimental results [57, 59].

Optimization of Phenotype

The phenotype associated with the problem to be solved has to be defined in the form of a biological objective. An objective function include the variables, or metabolic reactions, that are being maximized or minimized [1]. Mathematically, the objective function is used to define how much each reaction in the model contributes to the phenotype [26]. To identify optimal solutions, the objective functions are defined to solve a system of linear equations representing mass balance constraints [12]. In the case of predicting growth rate, the biological objective is biomass production. A biomass reaction, which is an artificial reaction that simulates metabolites consumed during biomass production, is added to the S-matrix in Equation 2.11.

The objective function and the mass balance constraints from Equation 2.12 define the following LP problem:

$$\text{maximize} \quad Z = \mathbf{c}^{\mathrm{T}}\mathbf{v} \tag{2.13}$$

$$\text{subject to} \quad \mathbf{S}\mathbf{v} = \mathbf{0} \tag{2.14}$$

$$\text{and} \quad v_i^{lb} \le v_i \le v_i^{ub} \tag{2.15}$$

where Z is the objective function and $\mathbf{c}^{\mathrm{T}}$ is the transposed vector of coefficients indicating how much each reaction contributes to Z. v_i is the flux of metabolite i, and v_i^{lb} and v_i^{ub} are the corresponding lower and upper bounds respectively [12, 26]. **S** and **v** are already defined in Equation 2.12. The upper and lower bounds in Equation 2.15 and the mass balance in Equation 2.14 define the problem's constraints, similar to the general programming problem in Equation 2.5-2.6. The LP problem above can be solved as described in Section 2.1.3, yielding a flux distribution that optimizes the system for its objective. The flux distribution is the set of all calculated flux values for all reactions [1]. Typically, the objective is to maximize the growth rate. This gives an optimal set of metabolic flux values that achieves the highest possible flux through the biomass reaction [3, 12, 26].

In any realistic metabolic model, there are more reactions than compounds ($n > m$). Because of this, there are a multitude of solutions that satisfy the LP problem, i.e. multiple intracellular flux distributions that underlie the same optimal value [12, 26, 60]. As most optimization methods, FBA will return only one of these optimal solutions within the feasible region [26]. Opposed to picking just one optimal solution, an extension of FBA called flux variability analysis (FVA) is able to explore the entire feasible region [3, 60].

2.4 Robustness of Metabolic Networks

Generally, robustness is a measure of the complexity of a network. In biology, network robustness is important for the studying of mutations. The network-based view in biology that has emerged over the past few decades has altered the way mutations and other genetic defects are studied. The damage of a protein or a gene often spreads through the network, leading to loss of function in one or several functional modules [61].

Robustness in metabolic networks is often associated with the concept of essential reactions and how easy it is to bypass defects and mutations [61, 62].

2.4.1 Essentiality and Superessentiality

A metabolic reaction is essential if removal of this reaction renders the network unable to synthesize all biomass molecules that are necessary for growth and survival in a given environment.

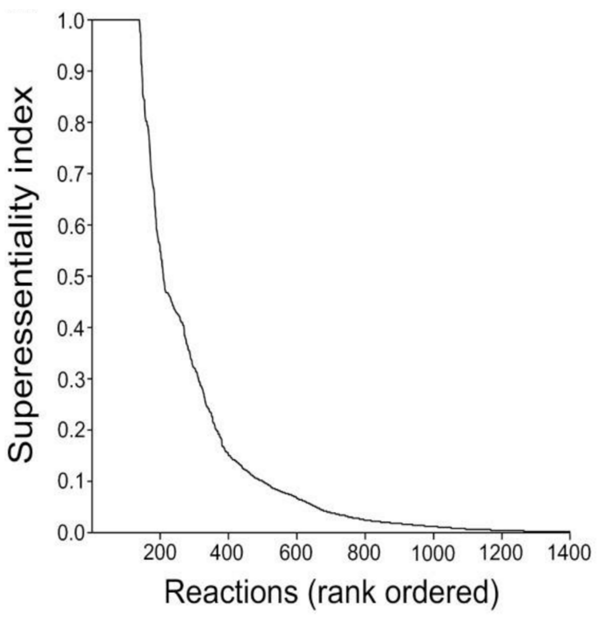


Figure 2.10: Rank plot of I_{SE} of 1,400 reactions in 500 random metabolic networks, where all reactions were essential in at least one of them. The plateau to the left corresponds to the 2.3% absolutely superessential reactions ($I_{SE} = 1$). The figure is obtained from Barve, Rodrigues, and Wagner [19].

Reactions that can be bypassed of other reactions or metabolic pathways in the network are considered non-essential [19, 61].

Essentiality is an organism- and environment-specific property. This means that a reaction that is essential in one environment or in one organism not necessary is essential in another environment of in another organism. When comparing essential reactions in different environments and multiple metabolic networks, it may be established whether they are essential in none, all, or just a few of them. If a biochemical reaction is essential in all random metabolic networks with a given phenotype, it is said to be superessential [19, 59, 61]. A measure for the superessentiality of a reaction is the superessentiality index (I_{SE}). This index is a number between zero and one, where a value of zero means that the reaction is non-essential in all tested networks, and a value of one means that it is essential in absolutely all. Barve, Rodrigues and Wagner [19] calculated the superessentiality indices for over 1,400 reactions, illustrated in Figure 2.10. The horizontal plateau at the left shows the 139 essential reactions with $I_{SE} = 1$, also called absolutely superessential reactions. These reactions can never be bypassed. The lowest I_{SE} in the plot was 0.02, which indicates that the corresponding reactions were essential in only

1 of 500 networks. This plot points out that most of the essential reactions can be bypassed, which makes metabolic networks quite robust to changes [19, 59].

2.5 Random Viable Metabolic Networks

An organism's metabolism evolves through deletion of reactions caused by loss of function mutations and addition of new reactions caused by horizontal gene transfer. Over time, this deletion and addition of reactions lead to metabolic networks that differ in their genotypes. Because the metabolic phenotype is very robust to genotypic changes, the organisms may still be able to synthesize biomass in the same chemical environment [19, 20, 21].

Random metabolic networks are made using algorithms based on principles of metabolic evolution. Previously, they have been generated using a method developed by the Andreas Wagner Lab [19], hereinafter referred to as the Wagner method. These random metabolic networks were generated by replacing reactions with alternatives taken from a collection of metabolic reactions called a reactions universe [19, 37]. Both the reaction universe and the Wagner method will be explained in this section.

2.5.1 The Reaction Universe

Randomization of reaction content is possible due to the construction of a metabolic reaction universe, which is the set of all reactions that are known to take place in a metabolic system [19]. The FBA method has made it possible to compute an organism's metabolic phenotype from its metabolic genotype [63]. An organism's metabolic genotype is all the reactions that are catalyzed by enzymes encoded in the organism's genome. The reaction universe is thus also referred to as a metabolic genotype space, because it comprises all possible metabolisms [64].

The reaction universe constructed by Øyaas in 2015 [37] include 13,849 metabolic reactions from the MNXref namespace available at MetNetX.org [65].

2.5.2 The Wagner Method

Random metabolic networks have previously been generated using a method based on Markov Chain Monte Carlo (MCMC) [66]. The Wagner method performs a random walk in a genotype network, which is a networks that consists of all metabolic genotypes that encodes the same phenotype [59, 63]. The genotype network is organized so that each node represents a metabolic genotype and the edges represent genotypic changes. Two nodes are connected if they differ from each other by only one reaction [59]. What the method actually does is removing and adding a metabolic reactions in each "walk", with the only requirement that the networks must still be able to produce biomass. Reactions are taken from the reaction universe [63].

Chapter 3

Software and Methods

This chapter presents the software tools and methods used to generate and present the results in this thesis. The modifications and adaptations of already existing procedures will also be described.

3.1 Software

Here, the computer programs and software tools used in this project is presented.

3.1.1 Python

The computer programs used to generate results were written in the Python programming language (version 2.7) [67]. An example of a Python script is included in Appendix A, where the SA algorithm is implemented.

3.1.2 COBRApy

COBRApy [68] is a Python module for COBRA methods, which are widely used in genome-scale modeling. In this project, both versions 0.4, 0.5, and 0.6 were used.

3.1.3 Gurobi

The Gurobi Optimizer [69] was used to solve the optimization problems in this project. Gurobi is a commercial mathematical programming solver that supports Python through the module Gurobipy. COBRApy includes an interface to Gurobi.

3.1.4 libSBML

All models in this project were stored in the Systems Biology Markup Language (SBML) file format. Working with SBML files in this study was facilitated by the libSBML library. With libSBML installed, SBML files could be read and written by built-in COBRApy functions [70].

3.1.5 Joblib and Multiprocessing

The Joblib package [71] is a set of tools that provide pipelining in Python. Joblib can be used for writing parallel loops, which makes it possible to perform multiple computations on different CPUs at the same time. Mutliprocessing [72] is another Python package that supports this.

3.1.6 Cytoscape

Visualization of metabolic networks were performed using Cytoscape [73]. This is a platform for visualizing complex networks that supports multiple file formats, such as SBML.

3.2 Computing Resources

All simulations in this study were run on computers in the Almaas Lab set up with Ubuntu 16.04 LTS operating systems (OS). Two different platforms with this OS were accessed:

- Dell Precision Optiplex T7500 with 12x Intel Xeon X5690 @ 3.47 GHz. 130 GB RAM.
- Three machines, each consisting of PoerEdge R730xd with 2x Intel Xeon E5-2680 v4 @ 2.4 GHz containing 56 threads with 448 GB RAM.

3.3 FBA

FBA with Gurobi set as solver was used to predict growth rates, find essential reactions, and to compute the number of blocked reactions in the models.

3.3.1 Growth Rate

The prediction of a model's growth rate was performed using FBA with maximization of the biomass reaction as objective function. This is a measurement of whether the model could produce biomass with the current reaction content or not. In this study, networks were considered as viable if they had a growth rate above 20 % of the growth rate in the original network.

3.3.2 Blocked Reactions

Blocked reactions are those reactions in the network that cannot carry flux. They were identified according to this procedure:

1. The model was imported into COBRApy and optimized.
2. The reactions with nonzero flux were identified. Reactions that carry flux can not be blocked, and were excluded from further analysis.
3. The remaining reactions, which had zero flux in the optimized model, were maximized and minimized. If both the maximum and minimum flux were below $\pm 10^{-8}$, the corresponding reaction were considered as blocked.

Due to numerical uncertainty, the $\pm 10^{-8}$ value is standard for computations like this. Without this uncertainty, the value would have been zero for both the lower and upper bound [68].

3.3.3 Essential Reactions

Essential reactions were identified through single reaction deletion. The idea behind this method is to study what happens if a certain reaction in the network is not allowed to carry flux, i.e. upper and lower bound set to zero. By studying how this affects the model as a whole, some of the reactions' characteristics can be analyzed.

Single reaction deletion was performed according to the following steps:

1. The model of interest was imported into COBRApy.
2. The wild-type growth rate was calculated.
3. Reactions with zero flux were definitely not essential, and were excluded from further analysis.
4. The remaining reactions were removed in an iterative manner. In each iteration, the new growth rate was computed. If the growth rate after deletion was less than 10^{-3} [68], the corresponding reaction was classified as essential. Each reaction was added again before next one was removed.

3.4 Metabolic Models and the Reaction Universe

The genome-scale metabolic reconstructions used in this project are presented in Table 3.1. They were downloaded from MetaNetX.org, which is an online platform "for accessing, analyzing and manipulating genome-scale metabolic networks and biochemical pathways" [65].

The reaction universe was made by Øyaas in 2015 [37]. The universe includes 13,849 balanced biochemical reactions taken from the same website as the models listed in Table 3.1. Thus, both the reaction universe and reconstructions are using MNXref, which is a common namespace for the metabolites, reactions and cellular compartments [74]. This standardization in nomenclature makes it possible to move reactions smoothly between the reaction universe and the metabolic models. An example of metabolic reactions using the MNX id is presented in Appendix B.

Table 3.1: Three high quality genome-scale metabolic reconstructions and their corresponding number of reactions, metabolites, and genes. The year of publication is also included.

Model	Organism name	Reactions	Metabolites	Genes	Year of publ.
iIT341	*Helicobacter pylori*	556	414	340	2005 [75]
iNJ661	*Mycobacterium tuberculosis*	1028	825	662	2007 [76]
iAF1260	*Escherichia coli*	2374	1668	1261	2007 [77]

3.5 Adaptation of the Simulated Annealing Algorithm

Random viable metabolic networks were generated from the already existing metabolic models presented in Table 3.1 using a modification of the SA algorithm. The traditional SA method is explained in Section 2.2. Unlike this original method, which has one temperature parameter, the modified algorithm has two such parameters. This means that each new solution had to go through two approval steps in order to be accepted. Both approval steps were adapted from the Metropolis criterion presented in Equation 2.7.

This section will explain how the two approval steps work.

3.5.1 First Step: Total Number of Reactions.

In the first approval step, the recent change in reaction content is approved based on the new number of reactions in the model. The probability of accepting the recent change is given by the following equation:

$$P_r = \exp\left(-\frac{\Delta R}{T_r}\right) \tag{3.1}$$

where ΔR is the difference between the optimal number of reactions and the current number of reactions. Like the temperature parameter in the traditional algorithm, T_r is high when the simulation starts. This leads to a high probability of accepting non-optimal solutions. As T_r is lowered, the probability of accepting a number of reactions that are far from the optimum also decreases.

3.5.2 Second Step: Number of Blocked Reactions.

The second step of approval is based on the number of blocked reactions. The probability of accepting a solution that contains a higher number of blocked reactions than the optimal number is:

$$P_b = \exp\left(-\frac{\Delta B}{T_b}\right) \tag{3.2}$$

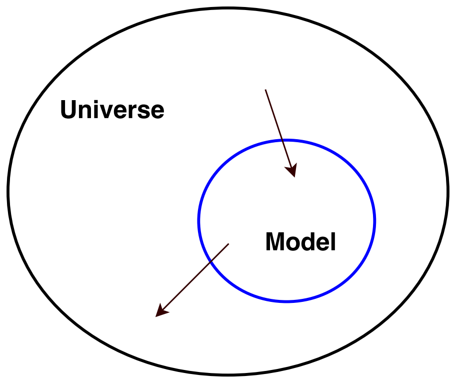


Figure 3.1: The concept of randomizing the reaction content in the metabolic models. As the figure illustrates, metabolic reactions are replaced by reactions from the universe.

where ΔB is the difference between the number of blocked reactions in the current solution and the optimal number of blocked reactions. The optimal number of blocked reactions is the same as the number of blocked reaction in the original model, which must be computed before the generation process starts. Unlike the temperature parameter in Equation 3.1, T_b is kept constant at a low value throughout the process.

3.6 Randomization of Metabolic Networks

This section will explain how the random viable metabolic reactions are generated using SA. As shown in Figure 3.1, the reaction content is randomized by adding and withdrawing reactions from the reaction universe. The modified implementation of SA in Python can be found in Appendix A, and the whole randomization process is shown in Figure 3.2. The procedure is as follows:

1. The reaction universe and a reaction model was imported into COBRApy.
2. Initial parameters were set. The predefined parameters are presented in Table 3.2
3. A random intracellular reaction was either added or removed from the model, both with a probability of 50 %.
    3.1. Adding a reaction: a randomly chosen reaction was moved from the reaction universe to the reaction model.

   3.2. Removing a reaction: a randomly chosen intracellular reaction was removed from the reaction model and added to the universe. The deletion of a reaction from the model could only be done if the model was still able to produce biomass (see Section 3.3.1).

4. The change in reaction content in the model was accepted by the following rules:

   4.1. If the number of reactions in the model is higher than the optimal number of reactions, a deletion will always be accepted.

   4.2. If the number of reactions in the model is lower than the optimal number of reactions, an addition will always be accepted.

   4.3. Otherwise, the addition/deletion is accepted by a probability given in equation 3.1

5. The number of blocked reactions in the model were calculated (see Section 3.3.2). If the number of blocked reactions was below the optimal number of blocked reactions, the current model was always accepted. If the number was higher than the optimal number, the new model was accepted by ha probability given in Equation 3.2.

6. T_1 in Equation 3.1 was lowered by 10 %.

7. Step 3-6 was repeated until $\mathrm{T_r}$ reached zero.

Table 3.2: The predefined parameters for generating random viable metabolic networks. The reason why these values were chosen is discussed in Appendix C.

Parameter	Value
$\mathrm{T_r}$	50
$\mathrm{T_b}$	0.5
Stop	1.0
Iterations	num of reactions in original network
Optimal num of reactions	num of reactions in original network
Optimal num of blocked reactions	num of blocked reactions in original network

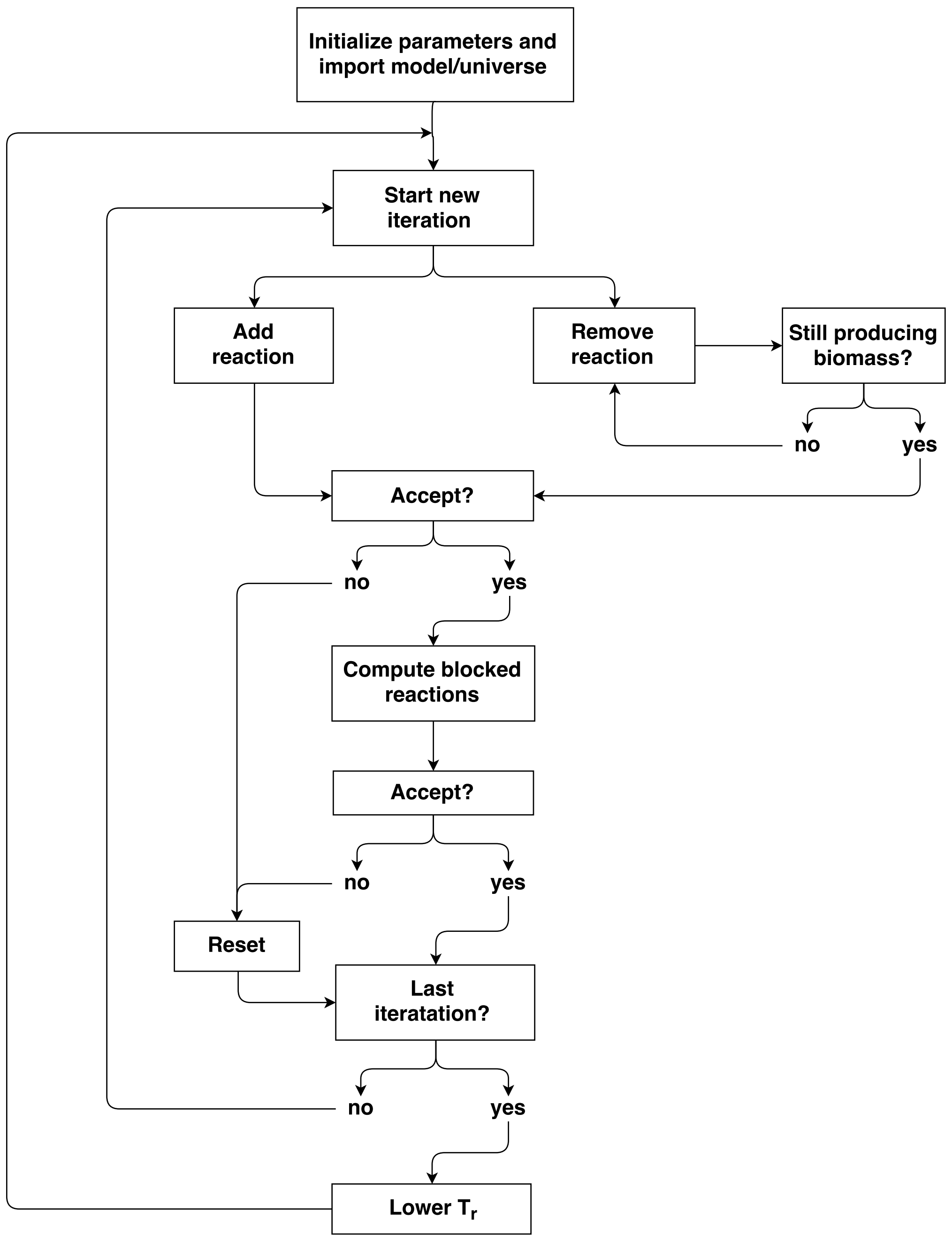


Figure 3.2: Flowchart illustrating the complete method for generating random viable metabolic networks. The top box illustrates the initialization process and the import of the model and universe into COBRApy. The bottom box illustrates the lowering of T_r. Everything in between those two illustrates one iteration of the method. The algorithm stops when T_r reaches its stop criterion.

3.7 Construction of Minimal Growth Media

Different growth media were constructed for two of the models iIT341 *H. pylori* and iNJ661 *M. tuberculosis*. The following procedure were used for construction of a given medium:

1. The reaction model was imported to COBRApy.

2. The uptake reaction for the carbon source of interest were identified for construction of anaerobic media. For aerobic, the uptake reaction for oxygen was also identified.

3. While the boundary reactions for uptake of the carbon source and oxygen were allowed to carry flux, all the other uptake reactions were switched on and off in an iterative manner. If the growth rate dropped below 2.0, the corresponding reaction was saved for later use. If the growth rate was still higher than 2.0, the reaction was turned off permanently.

4. The constructed growth medium included all the stored uptake reactions together with the carbon source and oxygen.

Chapter 4

Results and Discussion

In this chapter, obtained results are presented and discussed. The chapter is divided into the five following main parts, where the first three is testing and validation of the developed method:

- *Testing and validation of traditional SA.* The traditional SA algorithm, as explained in Section 2.2, was implemented and tested on both a 2D Ising model and the addition/deletion process of reactions. Here, only one temperature parameter was used.

- *Testing and validation of the second temperature parameter.* The second temperature parameter, T_b, was used to control the fraction of blocked reactions in the networks. Random networks were generated for multiple values of T_b.

- *Testing and validation of the program.* The whole program was tested to ensure that both the total number of reactions and the number of blocked reactions were kept at their desired levels. Some of the disadvantages of the method along with some suggestions on how these can be improved in the future are also presented here.

- *Analysis of the random viable metabolic networks.* The reaction content of three existing genome-scale metabolic models was randomized using the adapted SA algorithm. The requirements for the generating random metabolic networks were that they should:

  - have approximately the same size as the original networks.
  - have approximately the same number of blocked reactions as the original networks.

  - produce the precursors necessary for biomass production.

- *Software challenges.* During the production phase, some unforeseen problems with COBRApy were revealed. In this section, the troubleshooting process and some suggested solutions will be explained.

4.1 Testing and Validation of Traditional SA

The method used for generating random viable metabolic networks in this thesis was based on the SA algorithm presented in Section 2.2. The purpose of testing the SA algorithm was to verify that the method was understood and implemented correctly before the second temperature parameter was introduced. First, the method was used to solve a 2D Ising model. After this problem was solved, it was tested whether the method could be used to keep the number of reactions in a network at the desired level.

4.1.1 Solving the 2D Ising Model

The Ising model, explained in Section 2.2.4, is a well known optimization problem that can be solved with SA. In this example, a two dimensional version of the problem was used. In this version of the problem, the spins are arranged in a lattice, as illustrated in Figure 2.6. For convenience, the external magnetic field was turned off. The total energy in the system, as expressed in Equation 2.9 can thus be simplified to:

$$E = -\sum_{<i,j>} J_{ij}\sigma_i\sigma_j \tag{4.1}$$

where the magnetic part is eliminated. The objective in this optimization is to minimize the E in Equation 4.1.

In the implementation process 1 and -1 were used to denote the direction of 100 atomic spins, which were organized in a 10×10 lattice. Initially, these spins were randomly aligned, meaning around half of the atomic spins had the value 1 and the other half the value -1. This caused the average magnetization in the system, M from Equation 2.10, to be approximately zero. As long as the temperature was high, worse solutions had a high probability of being ac-

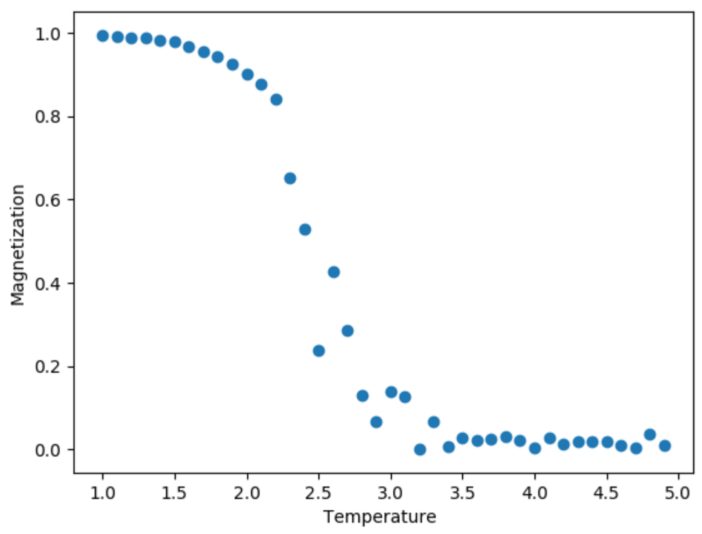


Figure 4.1: A 2D Ising model was optimized with SA. The optimization is illustrated with the average magnetization in the system as a function of the decreasing temperature.

cepted. As the temperature parameter was lowered, the probability of accepting worse solutions also decreased and neighboring atomic spins were aligned. This decreased the overall energy in the system. At the end, all atomic spins were aligned, meaning they all had the value 1 or -1. This corresponds to a average magnetization of 1.

The average magnetization in Figure 4.1 compared to the figure obtained from literature (see Figure 2.7) shows that they are almost identical. This indicates that the SA algorithm was implemented correctly.

4.1.2 Optimizing the Number of Reactions Using a Number Generator

The next step was to see if the same algorithm was able to optimize the number of reactions in a network. A number generator was used to simulate the addition and deletion of reactions, as shown in Figure 4.2. There were two reasons for this:

1. By avoiding usage of COBRApy and the actual reaction models, the method itself was tested instead of COBRApy specific functions.

2. Only one reaction can be added or removed in each iteration. By setting the number generator to return only $+1$ or -1, the process could be simulated in a very realistic way.

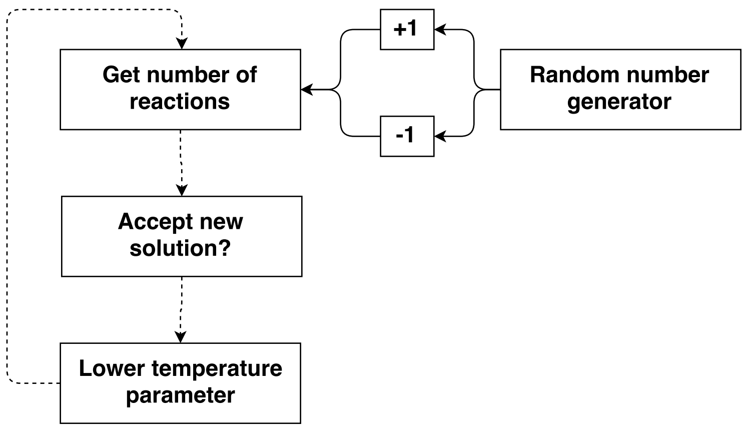


Figure 4.2: This figure shows one iteration of the simulation of the deletion/addition process of reactions using a number generator. The generator returns +1 or -1 by a probability of 50%.

The optimal number of reactions was set to 500, which is shown by the blue horizontal line in Figure 4.3. New solutions were always accepted if the total number of reactions approached this line from either above or below. Otherwise the new number of reactions was accepted by a probability given in Equation 3.1. The average number of reactions for each temperature parameter value was stored and plotted in Figure 4.3. As expected, the number of reactions had a high fluctuation around the optimum for high parameter values and approached the optimum as the temperate parameter was lowered. By comparing the acceptance rate from this simulation (Figure 4.3) to the theoretical acceptance rate from Figure 2.5, it clearly shows that this is in line with the behavior of a traditional SA algorithm.

In addition to validating the algorithm, this simulation gave valuable information about initialization of the parameters. Referring to the reaction numbers in Figure 4.3, the optimal value was reached long before the simulation was terminated. If efficiency is taken into account, the algorithm spent an unnecessary amount of time doing changes that were not accepted. By setting a stop criterion earlier, a lot of time could be saved.

4.1.3 Optimizing the Number of Reactions Using a Reaction Model

After the method was tested for the reaction number using a number generator, real networks were introduced. The difference between this test and the previous one was that biomass production was tested each time a reaction was removed. If the network could not produce biomass

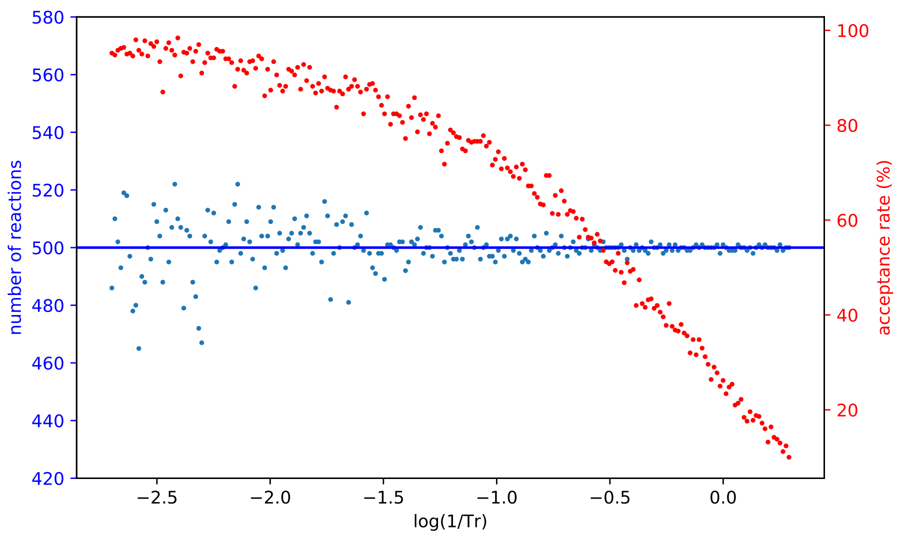


Figure 4.3: The average number of reactions (blue) and the acceptance rate (red) shown for each value of the T_r parameter. The number generator was used for the addition/removal. The horizontal line illustrates the optimal value, which was set to 500. T_r was lowered by 5% each 500^{th} iteration, from its start value (500) to the stop criterion (0.5). The number of reactions is shown as the average of all the 500 measurements for each value of T_r. The acceptance rate is shown as the fraction of accepted changes per total number of changes for each value of T_r.

after removal of a reaction, the method would try to remove another one. Due to this extra step, it was expected to see that the acceptance rate was lower than in Figure 4.3.

The model chosen for this test was the iIT341 *H. pylori* model. Its metabolic network consists of 557 reactions, which is illustrated with the blue horizontal line in Figure 4.4. Not surprisingly, more of the total changes were rejected due to the additional requirement for viability. However, the method was still able to optimize the network with respect to the number of metabolic reactions.

4.1.4 Summary

- The implemented SA method was able to optimize the 2D Ising model. This was concluded when the average magnetization measured in this test was compared to the one obtained from literature.

- The same algorithm were able to simulate the addition/deletion process of reactions us-

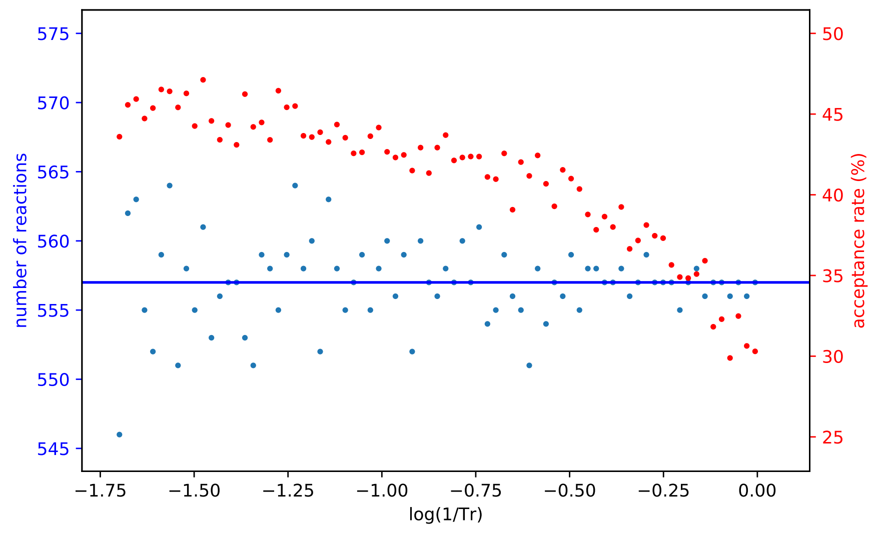


Figure 4.4: The addition/deletion of reactions using the iIT341 *H. pylori* model. The average number of reactions (blue) and acceptance rate (red) are shown for each value of the parameter T_r. The total number of reactions in iIT341 *H. pylori* is 557, illustrated with the blue horizontal line. T_r was set to 50 and was lowered with 5% each 500th iteration until the parameter reached 0.5.

ing both a number generator and an actual metabolic network. This was concluded by comparing the acceptance rates for the two tests with the one obtained from literature.

- Based on these three observations, it can be concluded that the SA algorithm with one temperature parameter (T_r) works properly.

4.2 Testing and Validation of T_b

The next step in the validation process was to test the second temperature parameter, T_b. The purpose of this test was to ensure that the number of blocked reactions was kept at approximately the same level as in the original models after randomization.

The test was performed by randomizing the reaction content in the iNJ661 *M. tuberculosis* model for different values for T_b. That means, the whole program was executed once for each of the T_b values chosen. It was expected that both the acceptance rate and the number of blocked reactions would decrease for low values of T_b, whereas high values would cause almost no restriction on the number of blocked reactions at all.

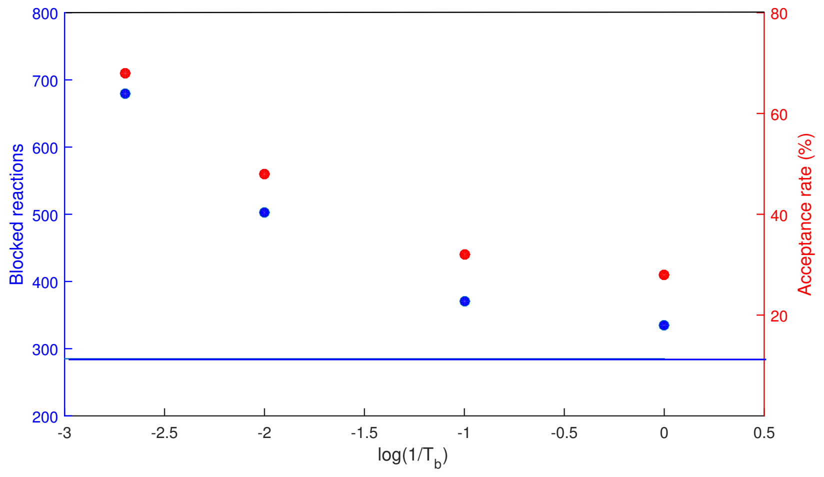


Figure 4.5: The number of blocked reactions (blue) and the acceptance rate (red) for generation of random metabolic models from the iNJ661 *M. tuberculosis* model for four different values of T_b: 500, 100, 10 and 1. The blue horizontal line illustrates the optimal number of blocked reactions, which is 285. T_r was set to 50 and was lowered by 10% each 1028th iteration until 1.0 was reached.

The results from this test are shown in Figure 4.5. The optimal number of blocked reactions for the iNJ661 *M. tuberculosis* model is 285, which corresponds to 28% of the reaction content. When T_b was set to 500, which means that the number of blocked reactions was virtually unrestricted, as many as 66% of the reactions in the random metabolic network were blocked. As expected, the number of blocked reactions decreased for lower values of T_b. For $T_b = 1$, the proportion of blocked reactions was 32%. These results show that the modified version of SA succeeds in controlling the number of blocked reactions in the randomized networks, as opposed to the Wagner method.

The acceptance rate for the four executions are also shown in Figure 4.5. To be able to plot the acceptance rate for the T_b parameter, only the reaction changes that were accepted from the first approval step were considered. The acceptance rate in the figure represent the average acceptance rate for the whole generation process of a new network.

4.2.1 Summary

- The T_b parameter can control the fraction of blocked reactions in a metabolic network.

4.3 Testing and Validation of the Program

After both temperature parameters had been studied, the performance of the whole program was tested. Testing of T_r and T_b have already shown that optimal values are reached after randomization of reaction content for both the total number of reactions and the fraction of blocked reactions. The purpose of testing the whole program was to confirm if this also applied when both of these factors were studied at the same time. In this section, attention is also paid to the fluctuation in reaction number and the computation time.

4.3.1 Optimal Solutions

For testing the number of reactions and number of blocked reactions after randomization, 100 random metabolic networks should be generated for the three original networks. Nevertheless, for the iAF1260 *E.coli* model only 10 networks were generated. For more info about this, see Section 4.5 about software challenges.

Figure 4.6 shows the results from this test. Here, both the optimal values and the average values for the random metabolic networks are represented. The dark blue and the dark red bars illustrate the total number of reactions and the number of blocked reactions, respectively. The same applies to the light blue and the light red bars, just that these illustrate the average numbers for the random networks.

It appears that the program works optimally when it comes to generating networks of the same size as the original networks, that is, networks with the same number of reactions. This can be seen clearly in Figure 4.6 where the blue bars show approximately the same values for both the original networks and the random ones. The number of blocked reactions proved to be approximately the same in the random networks as in the original ones for the models iIT341 *H. pylori* and iNJ661 *M. tuberculosis*. The number of blocked reactions in the random metabolic networks generated from the iAF1260 *E. coli* model was however much lower than in the original network. This was not expected, but it should again be noted that only 10 networks were generated from this model compared to 100 from the other two. Nevertheless, this concludes that this method succeeded in producing networks with a much lower number of blocked reactions than the Wagner method did.

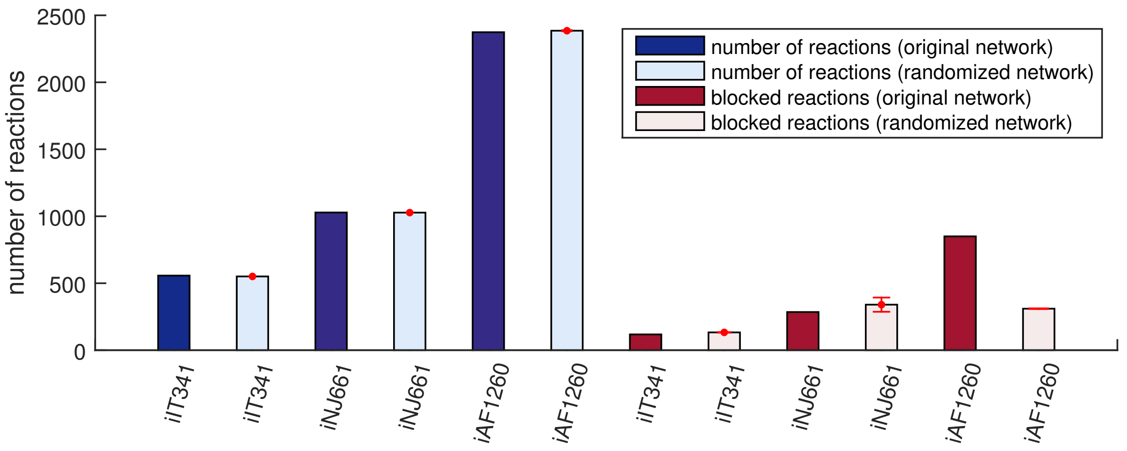


Figure 4.6: The total number of reactions (blue) and number of blocked reactions (red) in the random networks compared to the original ones. The numbers for the random networks from the models iIT341 *H. pylori* and iNJ661 *M. tuberculosis* show the average of 100 measurements. The values for randomization of iAF1260 *E. coli* are average values of 10 measurements.

4.3.2 Fluctuation in the Number of Reactions

To study the fluctuations in the total number of reactions and in the number of blocked reactions, every change in the reaction content was measured for five random network generations from the iIT341 *H. pylori* model. The averages of both the number of reactions and the number of blocked reactions were calculated for each value of T_r. The result is shown in the two subplots in Figure 4.7, where the fluctuations in the total number of reactions is shown in the upper one and the fluctuation in the number of blocked reactions is shown in the lower one. The plots in Figure 4.7 reveal two very important features of this method, where one being a strength and the other one a limitation.

The Limitation

One of the limitations of the program can be seen in the upper subplot in Figure 4.7. Although the number of reactions should be able to rise high above the optimal value, that was not the case. The number of reactions in the network was either around the optimal value or below it for all five executions. This may be a weakness in terms of randomizing the reaction content. If a higher number of reactions were allowed, it would be more likely that the added reactions could bypass some of the essential reactions, which would allow a larger proportion of the reaction

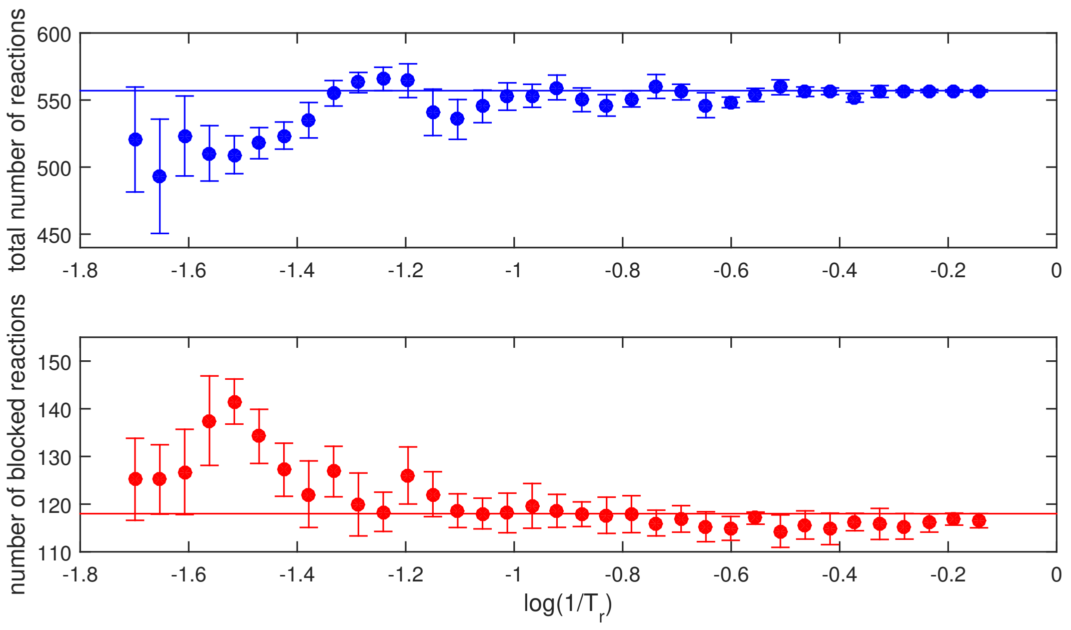


Figure 4.7: The fluctuations of total number of reactions and number of blocked reactions as functions of the decreasing parameter T_r. These numbers were measured for 5 executions of the program. T_r was set to 50 and was lowered by 10% until it reached 2. For each value of T_r, the average number of reactions and blocked reactions are shown. The error bars illustrate the fluctuation for each value of T_r. The horizontal lines show the optimal values for iIT341. The error bars indicate one standard deviation.

content to be replaced. This trend was not seen when reactions were added and removed without the requirement of blocked reactions, as shown in Figure 4.4. Then it was equally likely that the number of reactions was above the optimal value as below it. Based on this, it is assumed that there is the requirement for a low number of blocked reactions that restricts the total number of reactions. This makes sense, because a low total number of reactions gives a low number of blocked reactions.

Up to this point, the optimal number of blocked reactions has been set as the same as the number of blocked reactions in the original model. For the iIT341 *H. pylori* model, this number is 118, which corresponds to 21% of the total reaction content. An attempt to allow greater fluctuations was done by setting the optimal number of blocked reactions equal to 21% of the total number of reactions. Then the optimal number of blocked reactions would be updated each time a reaction was added or subtracted. Unfortunately, this proved not to be the case. Perhaps this is idea can be further investigated in the future. Another explanation may also be that the T_r was lowered too quickly in the first part of the executions. In the future it may be interesting to look at other ways of lowering this parameter.

The Advantage

The clear advantage is that although the number of blocked reactions rose above the optimum value, it was able to sink again. The whole purpose of this project was to generate a method that could generate networks with a low percentage of blocked reactions, and it certainly works. This can be seen clearly in both Figure 4.7 and Figure 4.6 where the the number of blocked reactions was approximately either at or below the optimal values.

4.3.3 Computation Time

In terms of efficiency, one of the major challenges of this method is its computation time. This section presents measures that have been implemented and tested as well as ideas for what can be done better in the future.

Reset to Previous Model

If the change in reaction content is not accepted, the new solution will be replaced by the previous one. In the first version of the program, this was done by taking copies of the entire network and the universe in each iteration. If the change was rejected, the previous network would be set as the current one. The copying of such large models was extremely time consuming; approximately 70-80% of the total running time was spent here.

The solution of this challenge was to remember the reaction that was added or removed in each iteration, and return to the previous version of the network by resetting the relevant reaction only.

Adding and Deletion of Reactions

Although adding and removing reactions are not the most time-consuming part of the program, it should still be streamlined due to the high number of iterations. As seen in Figure 4.5,

Table 4.1: Total time and time spent per hit given in seconds for the different functions in the program. The time usage was measured for the mid-sized iNJ661 *M. tuberculosis* model, and is the average of three executions of the program.

Total time spent (s)	Time spent per hit (s)	% Time	Functions/function contents
29	29.0	0.03	Import reaction model and reaction universe
378	0.02	0.48	Add reaction to network
4344	0.24	5.59	Remove reaction from network
73260	1.98	93.9	Compute the number of blocked reactions
1	1.10	0.0	Save random networks
78012			Total

fewer and fewer changes are accepted as T_r is lowered. The result is that much time is spent on changes that will not be accepted.

The solution was to accept or reject new changes before they were made by calculating ΔR in Equation 3.1 beforehand. This caused only changes that were already accepted to be made. This is especially time-saving when T_r is low, because very few changes will be accepted at this point anyway.

Parallel Programming

After the two improvements mentioned above, the most time-consuming function was the calculation of blocked reactions. The time spent for each of the functions is presented in Table 4.1. The time usage is not directly transferable to the other models because of the difference in size, but it is a good indicator of where the bottleneck in the program is. For models that are either smaller or larger it will take shorter or longer time to calculate the number of blocked reactions, respectively.

The idea was to run this function in parallels to make the computation of blocked reactions more effective. Both Joblib and the Multiprocessing package for Python were tested.

- Parallel programming with Joblib was tested on 12 processors. It was expected to see that the program would run significantly faster, but it turned out that the relevant function became five times slower instead.

- The Multiprocessing package was tested because of the non-expecting results after parallel processing with Joblib. The computation of blocked reactions was spread out on 8 processors, but this only made the function run 10% faster. It is a waste of resources to use multiple processors for such a small improvement in speed, and the multiprocessing step was removed from the program.

A possible explanation for these results is that it takes a lot of time to divide all the reactions into subsets and process the information that comes back from the processors compared to computing the blocked reactions as before. For this reason, the idea of parallel programming was discarded.

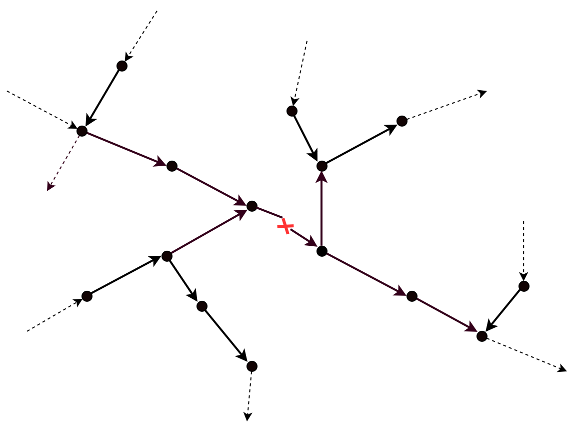
Alternative Ways of Computing Blocked Reactions

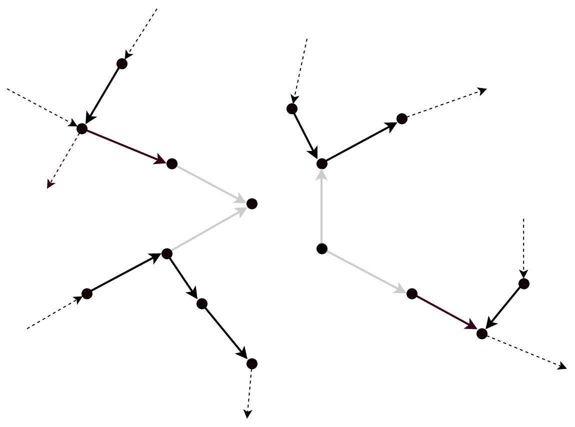
In the current version of the program, the number of blocked reactions is computed by analyzing all reactions in the model in each iteration, as explained in Section 3.3.2. As already mentioned, this is very time-consuming. This function could have been implemented in an alternative way that will be outlined here:

- Adding a reaction can never increase the number of blocked reactions by more than 1. That is, if the added reaction itself is blocked. If the added reaction is blocked, no other reactions needs to be analyzed. If the reaction is not blocked, it must be investigated whether it may have made some of the blocked reactions unblocked.

- Removing a reaction can never make blocked reactions unblocked. The reactions that are directly connected to the reaction being removed need to be investigated. The idea is checking these adjacent reactions in a stepwise manner as shown in Figure 4.8. The first step is looking at the reactions that were directly connected to the removed reaction and see if they are still able to carry flux. Those that cannot are illustrated with gray color, which in this case were all of them. The next step is looking at their directly connected reactions to see if any of these are still able to carry flux. This procedure is continued until no more adjacent reactions is blocked.

Another method for computation of the number of blocked reactions has been developed by Vlassis *et al.* [78]. This procedure is completely different from the one used in this thesis. The idea is to maximize the number of feasible fluxes in the system, which can be explained


(a) Removal of reaction


(b) Blocked reactions in step one

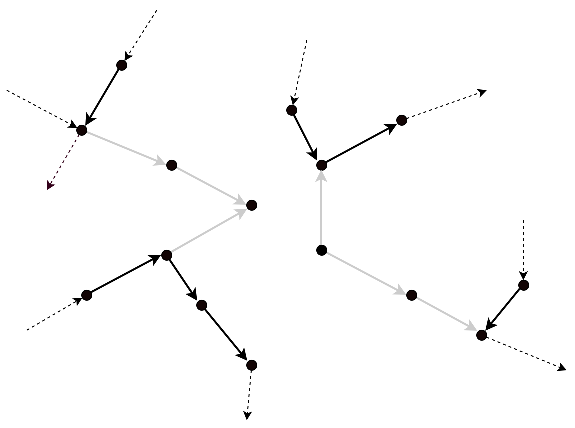

(c) Blocked reactions in step two

Figure 4.8: The alternative way of computing blocked reactions when a reaction is removed. Here, the reaction with the X is removed from the network. Then, the reactions illustrated by gray arrows are not able to carry flux, and will be considered as blocked. The dashed arrows are connected to the rest of the network.

as "pushing" flux through as many reactions as possible. In this way, the number of non-zero entries in the flux vector **v** will be maximized (see Section 2.3.4 about FBA for more information about the flux vector). The number of zero entries in **v** corresponds the number of blocked reactions. A single iteration of this method will find all the blocked and unblocked reactions in the network in a much more effective way than the method implemented in this thesis.

The two alternative procedures described above are suggestions for how the program can be more effective in the future and were not implemented in this thesis.

4.3.4 Summary

- It has been proven that the implemented method can generate networks of the same size as the original networks.

- The method was also able to keep the fraction of blocked reactions at or below the set limit.

- The fluctuations in the number of reactions were smaller as expected. Alternative ways to lower T_r should be investigate in the future.

- The most time consuming part of the program was the computations of blocked reactions. Two faster alternatives have been outlined.

Table 4.2: Overview of the generated metabolic networks. From each of the original models, three different types of randomized networks were generated. They differ from each other in the fractions of blocked reactions that were set as optimum in the initialization process, shown as the percentage on top of each column. * For info about the low number of generated networks from the iAF1260 *E. coli* model, see Section 4.5 about software challenges.

	Original	65%	200%
iIT341 *H. pylori*	100	100	100
iNJ661 *M. tuberculosis*	100	100	100
iAF1260 *E. coli*	10*	6*	11*

4.4 Analysis of Random Viable Metabolic Networks

Random viable metabolic networks were generated from the three existing models iIT341 *H. pylori*, iNJ661 *M. tuberculosis*, and iAF1260 *E. coli*. These networks were produced by changing their reaction content, as explained in Section 3.5. Three requirements were formulated for the generation of new networks:

- The first requirement was that the networks should consist of approximately as many reactions as the original network.
- The second requirement was that the networks should be able to produce biomass so that they could be considered viable.
- The third requirement was to keep the proportion of blocked reactions at a certain level.

The third requirement was not formulated in the Wagner method, so the number of blocked reactions in the networks generated by this method was much higher than expected. Therefore, extra attention was paid to the number of blocked reactions in this thesis.

An overview of the random viable metabolic networks that were generated is given in Table 4.2. Note that for each of the three models, random networks were generated with three different levels of blocked reactions. For the lowest level, the upper limit of blocked reactions was set as the same as in the original models, which are 21 %, 28 %, and 34 % for the three models iIT341, iNJ661 and iAF1260, respectively. For the highest level, the number was thus set to 200 % of the total reaction number to ensure that it did not put any restrictions on the randomization. For the middle level, the upper limit of blocked reactions was set to 65 %. For simplicity, these variations in the networks will be separated from each other in the remaining text with these abbreviations:

- *LB* - low level of blocked reactions. Optimal value was set to the same as the proportion of blocked reactions in the original models.
- *MB* - medium level of blocked reactions. Optimal value was set to 65 % of the total number of reactions.

- *HB* - high level of blocked reactions. Optimal value was set to 200 % to ensure no restriction on the randomization process.

The purpose of generating networks for different levels of blocked reactions was to study how the networks' properties changed as a function of blocked reactions and how it affected the randomization.

This section presents some of the properties of the random viable metabolic networks that were generated. Many of the same properties were measured in Øyaas' master thesis [37], where random metabolic networks were generated using the Wagner method. The average randomization degree, fraction of blocked reactions, and fraction of essential reactions were calculated for all the random metabolic networks generated. These results are plotted in Figure 4.9. Then, the random metabolic networks were tested for biomass production in different growth media.

4.4.1 Evaluation of the Randomization of Reaction Content

The randomization degree was determined as the fraction of metabolic reactions in the random network that did not participate in the original model.

The average degree of randomization is illustrated with blue triangles in the Figure 4.9. The degree was highest in the *HB*-networks, which virtually had no restriction on the number of blocked reactions. The lowest degree of randomization was measured in the networks with the lowest proportion of blocked reactions. For these networks, fewer reactions could be replaced compared to the *HB*- and *MB*-networks. The observation that the randomization degree correlated with the proportion of blocked reactions was therefore expected. From optimization theory, it is also known that the solution space of a problem can never be increased by adding more constraints [35]. By imposing the restriction on blocked reactions, the randomization degree could therefore never be higher than in the unrestricted networks.

The size of the models also appeared to be an important factor of the degree of randomization. The large iAF1260 *E. coli* model had a higher average randomization degree than the small iIT341 *H. pylori* model, and the mid-sized iNJ661 *M. tuberculosis* model's randomization degree was placed between those two. Due to small models' few reactions, they have fewer alternative ways of producing the biomass precursors needed for growth than larger models, imply-

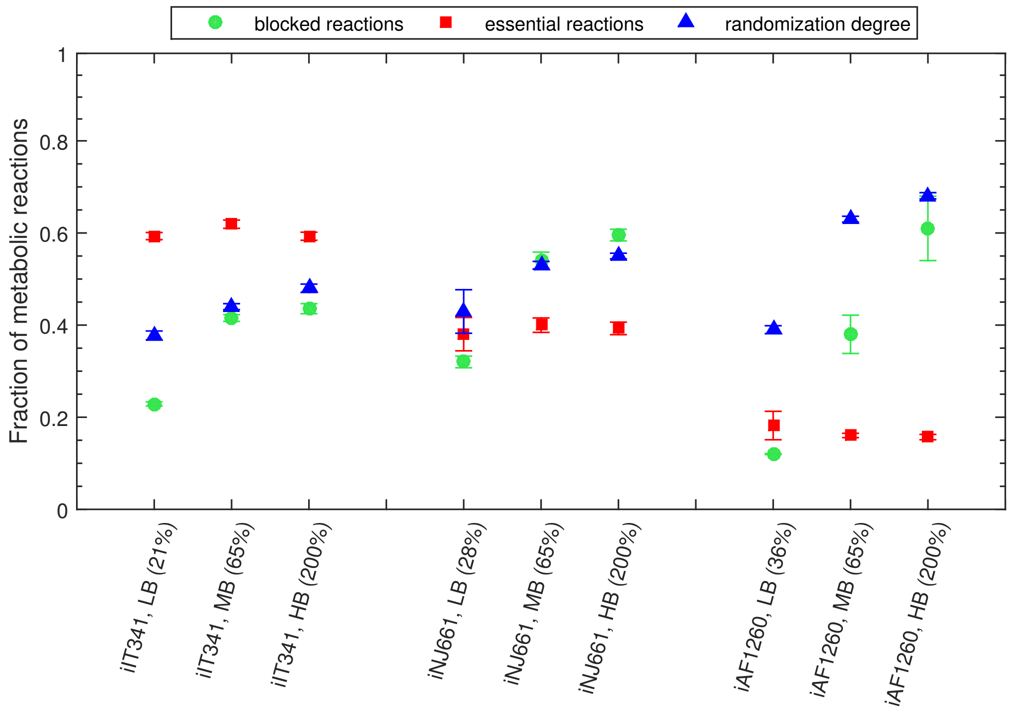


Figure 4.9: Average fraction of metabolic reactions in the three different categories; blocked reactions, essential reactions, and reactions not found in the original networks. The models from which randomized networks were generated are indicated along the horizontal axis. The number of networks in each category is shown in Table 4.2. Measurements were done for three different levels of blocked reactions to study how the proportion of blocked reactions affect the other properties. The error bars indicate one standard deviation.

ing larger share of essential reactions. The randomization process used to generate these networks repeatedly removed nonessential reactions from the networks and added new ones. For networks with few reactions, only a small fraction of reaction was actually possible to remove. Therefore, a higher randomization degree is difficult to obtain. This trend is further enhanced by the fact that the number of reactions rarely rises above optimal value during the generation process, which causes essential reactions to be even more difficult to bypass. This limitation is shown in the upper subplot in Figure 4.7. If the method can be improved so that higher numbers of reactions are allowed, the randomization degree will probably increase for all three models.

4.4.2 Evaluation of Blocked Reactions

The average fraction of blocked reactions is illustrated with green circles in Figure 4.9. The number of blocked reactions is interesting to study in this project, both in terms of the generated networks' structure and the developed method's functionality.

Random metabolic networks generated from the iIT341 *H. pylori* model had an overall low fraction of blocked reactions. For the *LB*-networks, the average fraction was 0.23, which was approximately the same as in the original model (0.21). For the *MB*-networks and the *HB*-networks, the average fractions were 0.42 and 0.45, respectively.

Random networks generated from the iNJ661 *M. tuberculosis* model had higher shares of blocked reactions than those generated from iIT341 *H. pylori*. The random metabolic *LB*-networks had a fraction of 0.33, which was a little bit higher than that of the original model (0.28). For the two others, the average fractions of blocked reactions were 0.55 for the *MB*-networks and 0.61 for the *HB*-networks.

Networks generated from iAF1260 *E. coli* had the greatest spread in blocked reactions, raging from an average fraction of 0.13 to 0.62. The random metabolic *LB*-networks generated from this model were also the only ones that had fewer blocked reactions than the original metabolic network has.

Networks containing large fractions of blocked reactions have relatively few reactions that are able to carry flux. A consequence of this is that the networks are sparsely connected and hence little robust for intracellular and extracellular changes. On the other hand, networks with low fraction of blocked reactions have many reactions that can carry flux, which leads to robust and densely connected networks that can easily adapt to changes. It is obvious that the latter topological structure is preferred for metabolic networks. All generated random metabolic *LB*-networks had approximately as many blocked reactions as in the original model, or fewer.

Compared to the random networks generated using the Wagner method, the *LB*-networks were definitely more similar to the original models with respect to the fraction of blocked reactions. This is shown in Table 4.3. The *HB*-networks are those that were most similar to the previous networks generated in Øyaas' master thesis. This was expected because both of them were generated with no restrictions on the number of blocked reactions.

One last thing that may be worth noting is the low numbers of blocked reactions in the *LB*-networks generated from the iAF1260 *E. coli* model compared to the others. The implemented method does not try to force the number of blocked reactions to the exact same value as in the original networks, as is done for the total number of reactions. All numbers of blocked reactions

Table 4.3: The fraction of blocked reactions in the original models together with the average fractions for random metabolic networks generated by Wagner's method and SA. For the networks generated by SA, the average value for the *LB*-networks is shown.

	Original	Wagner's method	SA method
iIT341	0.21	0.50	0.22
iNJ661	0.28	0.55	0.33
iAF1260	0.36	0.75	0.13

are considered as optimal as long as they are equal to or below the set limit. A low fraction of blocked reactions is therefore not wrong according to the method, just a bit surprising. An obvious weakness of these results is that very few random metabolic networks were generated from the iAF1260 *E. coli* model due to unforeseen challenges with COBRApy. Therefore, more networks should be generated in order to say something quite sure about these results. If this is done and the results remain the same, it might be interesting to see if this happen in other metabolic networks of similar size as well.

4.4.3 Evaluation of Essential Reactions

The average fraction of essential reactions in the generated metabolic networks is illustrated with red squares in Figure 4.9. On average, the fraction of essential reactions is higher in small metabolic networks than in large ones. This is already discussed in the paragraph about randomization degree.

When it comes to the *LB*-networks generated using this method compared to those generated with the Wagner method, the fraction of reactions considered as blocked versus essential is worth noting. For the networks generated in Øyaas' thesis, almost all reactions were considered either as blocked or essential. This was also observed for the *HB*-networks in this study. Many reactions were neither considered as blocked nor essential in the *LB*-networks. An overview of these reactions is viewed in Table 4.4.

These results were very surprising. As the number of blocked reactions was lowered, it was expected that there would be fewer essential reactions because the network would contain more alternative pathways for biomass production. Due to these results, the networks were visualized to study if there were any obvious differences between the original networks and the random

Table 4.4: The average fraction of reactions that are neither essential nor blocked in the random metabolic networks generated.

	LB-networks	*MB*-networks	*HB*-networks
iIT341	0.17	0.02	0.0
iNJ661	0.26	0.05	0.0
iAF1260	0.66	0.48	0.23

ones. The original networks and one of the *LB*-networks from each of the three models were imported into Cytoscape and visualized. These six network can be found in Appendix D. These network structures explain why the number of essential reactions was higher than expected. In many of the random metabolic networks, multiple clusters of reactions formed. This explains why so many of the reactions could not bypass the essential ones. For future network generation, the method should be modified to withstand formation of additional sub-networks.

4.4.4 Growth in Different Environments

Adding and removing metabolic reactions is a way of simulating metabolic evolution in organisms. In nature organisms pay no effort in keeping reactions that are not used. Over time, these will disappear.

To see if this happens when generating random metabolic networks, different growth media were constructed for the iIT341 *H. pylori* model. In the randomization process, only intracellular reactions were replaced, which means that all exchange reactions were intact in the random metabolic networks. By testing the networks' growth rate in different growth media, it would be clarified whether unused metabolic pathways were preserved in the randomization process or not. Figure 4.10 shows the concept of changing growth media for a metabolic networks. The arrows represent the boundary reactions that transport metabolites from the extracellular space and into the cytoplasm. New media were constructed by turning desired uptake reactions on or off. The different growth media used in this study are presented in Table 4.5. The original metabolic network of the iIT341 *H. pylori* model was able to produce the biomass precursors required for growth in all of them.

The average growth rates for the *LB*-networks, *MB*-networks, and *HB*-networks in different growth media are shown in Figure 4.11. The highest growth rate for all the three types of net-

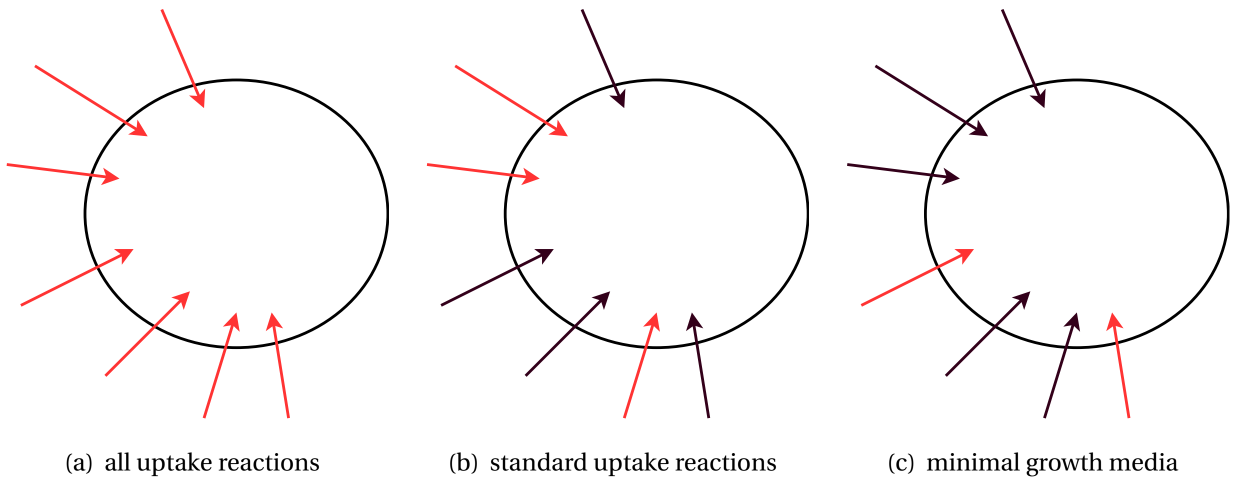


Figure 4.10: The concept of different growth media. Red arrows represent uptake reactions turned on, black arrows represent uptake reactions turned off, and the circle represents the intracellular space of metabolic reactions.

Table 4.5: The seven different growth media tested for networks generated from the iIT341 *H. pylori* model. Medium 1 consists of all uptake reactions, medium 2 is the uptake reactions that are standard for the iIT341 model, and medium 3-7 are minimal growth media constructed for different carbon sources. The standard uptake reactions for iIT341 are listed in Appendix B.

	type of medium
medium 1	all uptake reactions
medium 2	standard uptake reactions
medium 3	minimal glucose aerobic
medium 4	minimal glucose anaerobic
medium 5	minimal succinate aerobic
medium 6	minimal pyruvate aerobic
medium 7	minimal acetate aerobic

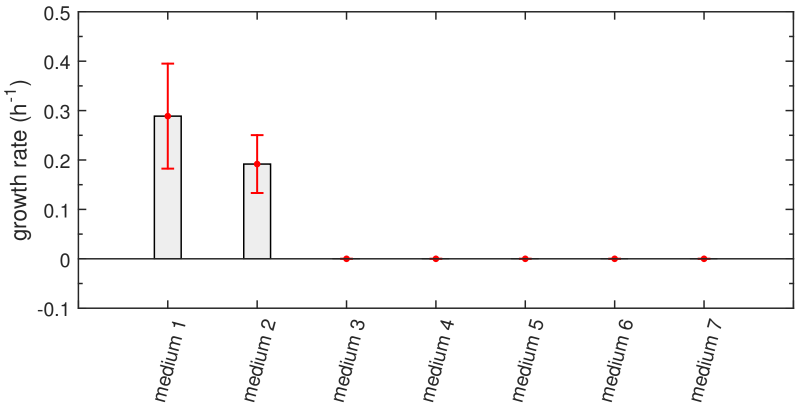
works were measured when all uptake reactions were turned on. For iIT341 *H. pylori* there are a total of 77 uptake reactions, which are the boundary reactions that move metabolites from the extracellular space and into the organism. The second highest growth rate was measured when the only the standard uptake reactions for the iIT341 *H. pylori* were turned on. This is the composition of uptake reactions used when the random metabolic networks were generated. These standard uptake reactions for iIT341 *H. pylori* are listed in Appendix B. No growth was measured for the five minimal growth media.

That the highest growth rate was measured when all the uptake reactions were turned on was not surprising. Here, all extracellular metabolites were made available for the metabolic networks, which enabled them to generate all biomass precursors that the networks still had the metabolic pathways to produce. That the second highest growth rate was measured for the standard composition of uptake reactions was also expected because these pathways had to be preserved in the generation process due to the requirement of biomass production. That no growth rate was measured for any of the minimal growth media means that the metabolic pathways that synthesize biomass precursors from these compounds were lost in the randomization process.

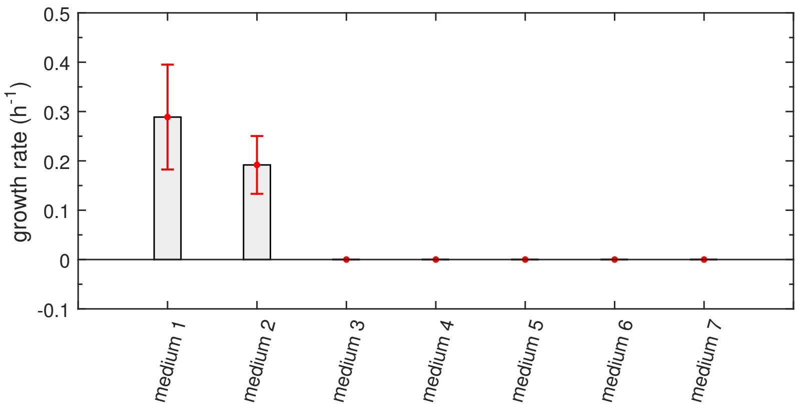
For later studies, this means that networks should be generated for a variety of uptake reactions turned on to be able to study other pathways in the networks.

4.4.5 Summary

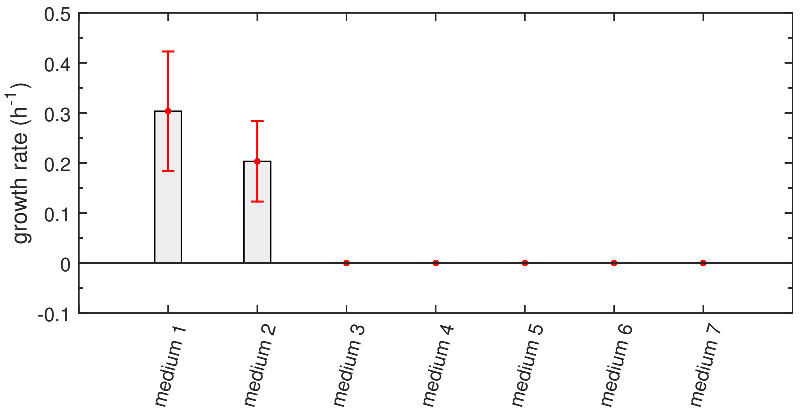
- The randomization degree was lower in small networks than in large ones because a smaller fraction of the reaction content can be replaced.
- The fraction of essential reactions was much higher than expected. When the random networks were visualized, it could be concluded that the formation of multiple clusters caused this surprising results.
- Networks were only able to produce biomass in the same growth media they were generated in. Unused reactions were lost in the randomization process.



(a) *LB*-networks generated from the iIT341 *H. pylori* model.



(b) *MB*-networks generated from the iIT341 *H. pylori* model.



(c) *HB*-networks generated from the iIT341 *H. pylori* model.

Figure 4.11: The growth rates of the random metabolic networks generated from the iIT341 *H. pylori* model in different growth media. The error bars indicate one standard deviation. The growth media are listed in Table 4.5.

4.5 Software challenges

During the production phase, some unforeseen problems with COBRApy were revealed. Those problems were first discovered during simulation of the random networks on the Almaas Lab machines in March 2017. The testing and validation of the method were done on a private computer, and the same problems were never seen here.

4.5.1 Troubleshooting

Initially, the program code was erased and rewritten, to be sure there were no small mistakes in implementation that triggered the problems. All reaction models were then downloaded again from MetaNetX.org. Regarding the reaction universe, Øyaas was contacted to ensure that it was the correct version of the universe that had been received (Ove Øyaas, personal communication, April 2017). All the procedures and functions in the program were debugged on a private computer to see that they acted as they should, but the same error messages were not possible to evoke. This computer had an older version of COBRApy installed. This indicated that there was nothing wrong with neither the implementation, the models, nor the reaction universe. Observations from the debugging step indicated that there was nothing wrong with the interaction between the implemented program and the COBRApy specific functions.

After a closer look at the error messages received during the network generation, it appeared that they came from other solvers than Gurobi. Gurobi was therefore reinstalled on the Almaas Lab machines, but the problem was still not resolved. At this point, the COBRApy developer team was contacted. The developer team had stopped working on the solver interfaces in COBRApy 0.5.0, which is the COBRApy version that was used in this study. A new version that would probably fix the problems would soon be released (contributor Moritz Beber, personal communication, May 2017). The new version of COBRApy (version 0.6.0) was released later in May. After installing and testing the new version, it could be concluded that it was the COBRApy version 0.5.0 that had caused the problems.

4.5.2 Limitations

- Due to problems with the COBRApy solvers, there were not generated as many networks as intended from the iAF1260 *E. coli* model. The problems affected these networks the most because iAF1260 is the largest model. Therefore, these networks took the longest time to generate. The likelihood that the execution was aborted due to error messages was thus much higher here. Due to the small number of networks, the results may be considered as inconclusive.

- Minimal growth media were also constructed for the iNJ661 *M. tuberculosis* model, as for the iIT341 *H. pylori* model in Section 4.4.4. The biomass production in the original *M. tuberculosis* model was tested in all growth media and it was observed that it had a positive growth rate in all of them. However, problems occurred when all the random networks should be tested for growth on the Almaas Lab machines. None of the networks, not even the original *M. tuberculosis* network, were able to produce biomass in any of the growth media. This test was therefore postponed until the new version of COBRApy was released. Unfortunately, many of the COBRApy specific functions were renamed so that the original program code could not be used.

4.5.3 Summary

- Challenges related to COBRApy delayed the generation and testing of random metabolic networks.

- After the new version of COBRApy was installed, the problems disappeared.

- There was nothing wrong with the implemented method, the metabolic networks, or the reaction universe.

Chapter 5

Conclusions

A method for generating random viable metabolic networks was developed. The method was based on the SA algorithm and was adapted to contain two temperature parameters instead of one. The verification of the implemented method was done by performing three tests: One for the first temperature parameter, one for the second temperate parameter, and one for the completed program. Random viable metabolic networks were generated for different levels of blocked reactions. The purpose was to study how the fraction of blocked reactions affected the networks' properties. It was particularly interesting to investigate whether a higher fraction of non-blocked reactions led to a lower fraction of the essential reactions. I that case, it would indicate that the reactions added from the reaction universe were able to bypass some of the essential reactions.

By testing the first temperature parameter, T_r, it was confirmed that the SA algorithm was implemented correctly. Results were compared to algorithm specific characteristics from literature, for instance the average magnetization of an Ising model and the acceptance rate for SA. It was also concluded that SA was able to optimize the number of reactions in an actual metabolic network by randomizing a iIT341 *H. pylori* network.

Testing of the second temperature parameter, T_b, revealed that the method succeeded in controlling the number of blocked reactions in the networks. This test was performed by generating random networks from the iNJ661 *M. tuberculosis* model for different values of T_b.

When the program was completed, it was extensively tested before production of the random networks started. The measurements of fluctuations in reaction number indicated that T_r

may have been lowered in a non-optimal way. Investigation of the computation time revealed that one calculation caused over 90 % of the time usage. Alternative ways to perform this calculation have been suggested.

When the random metabolic networks were generated, a limitation of the COBRApy software was revealed. However, most of the networks were generated and tested. Surprisingly, the fraction of essential reactions were not lower than in previous network generations. Therefore, the networks were visualized to study their structure. It appeared that many of the random metabolic networks had formed multiple reaction clusters, which explains the non-expected number of essential reactions.

Finally, the random networks generated from the iIT341 *H. pylori* model were tested for growth in different growth media. As expected, the networks could not produce biomass in other growth media than the one they had been generated for. For future investigation of metabolic pathways or growth media, different uptake reactions should be turned on.

A possible extension of the work done here would be a better implementation of the method. Other cooling schedules for the T_r parameter should be considered together with another algorithm for the calculation of blocked reactions (see Section 4.3.3 for suggestions). The method should also withstand formation of several clusters. Otherwise, the idea behind the method is good, as the fraction of blocked reactions was reduced significantly.

Bibliography

[1] Aarash Bordbar, Jonathan M Monk, Zachary A King, and Bernhard O Palsson. Constraint-based models predict metabolic and associated cellular functions. *Nature Reviews Genetics*, 15(2):107–120, 2014.

[2] Bernhard Ø Palsson. *Systems Biology - Properties of Reconstructed Networks*. Cambridge University Press, 2006.

[3] Matthew A Oberhardt, Bernhard Ø Palsson, and Jason A Papin. Applications of genome-scale metabolic reconstructions. *Molecular Systems Biology*, 5(1), 2009.

[4] Rama S Singh. Darwin to dna, molecules to morphology: the end of classical population genetics and the road ahead. *Genome*, 46(6):938–942, 2003.

[5] Eberhard O. Voit. *A First Course in Systems Biology*. Garland Science, 1st edition, 2013.

[6] Fulvio Mazzocchi. Complexity in biology. *EMBO Reports*, 9(1):10–14, 2008.

[7] Kiran Raosaheb Patil, Mats Åkesson, and Jens Nielsen. Use of genome-scale microbial models for metabolic engineering. *Current Opinion in Biotechnology*, 15(1):64–69, 2004.

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

[9] Bernhard Ø Palsson. *Systems biology*. Cambridge University Press, 2015.

[10] Jong Myoung Park, Tae Yong Kim, and Sang Yup Lee. Constraints-based genome-scale metabolic simulation for systems metabolic engineering. *Biotechnology Advances*, 27(6):979–988, 2009.

[11] Albert-Laszlo Barabasi and Zoltan N Oltvai. Network biology: Understanding the cell's functional organization. *Nature Reviews Genetics*, 5(2):101–113, 2004.

[12] Robert Schuetz, Lars Kuepfer, and Uwe Sauer. Systematic evaluation of objective functions for predicting intracellular fluxes in escherichia coli. *Molecular Systems Biology*, 3(1):119, 2007.

[13] R Aebersold. Molecular systems biology: a new journal for a new biology? *Molecular Systems Biology*, 1(1), 2005.

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

[15] James E Bailey. Toward a science of metabolic engineering. *Science*, 252(5013):1668–1676, 1991.

[16] Masato Ikeda and Ryoichi Katsumata. Metabolic engineering to produce tyrosine or phenylalanine in a tryptophan-producing corynebacterium glutamicum strain. *Applied and Environmental Microbiology*, 58(3):781–785, 1992.

[17] Hyun Uk Kim, Tae Yong Kim, and Sang Yup Lee. Metabolic flux analysis and metabolic engineering of microorganisms. *Molecular BioSystems*, 4(2):113–120, 2008.

[18] George Stephanopoulos, Aristos A Aristidou, and Jens Nielsen. *Metabolic engineering: principles and methodologies.* Academic Press, 1998.

[19] Aditya Barve, João Frederico Matias Rodrigues, and Andreas Wagner. Superessential reactions in metabolic networks. *Proceedings of the National Academy of Sciences*, 109(18):E1121–E1130, 2012.

[20] Carlton Gyles and Patrick Boerlin. Horizontally transferred genetic elements and their role in pathogenesis of bacterial disease. *Veterinary Pathology*, 51(2):328–340, 2014.

[21] Jeremy S Edwards and Bernhard O Palsson. Robustness analysis of the escherichia coli metabolic network. *Biotechnology Progress*, 16(6):927–939, 2000.

[22] Andanappa K Gadad, Chanabasappa S Mahajanshetti, Sudarshan Nimbalkar, and Anandkumar Raichurkar. Synthesis and antibacterial activity of some 5-guanylhydrazone/thiocyanato-6-arylimidazo [2, 1-b]-1, 3, 4-thiadiazole-2-sulfonamide derivatives. *European Journal of Medicinal Chemistry*, 35(9):853–857, 2000.

[23] Fred C Tenover. Mechanisms of antimicrobial resistance in bacteria. *The American Journal of Medicine*, 119(6):S3–S10, 2006.

[24] WHO et al. Antimicrobial resistance: global report on surveillance 2014. geneva, switzerland: Who; 2014, 2014.

[25] David J Payne, Michael N Gwynn, David J Holmes, and David L Pompliano. Drugs for bad bugs: confronting the challenges of antibacterial discovery. *Nature Reviews Drug Discovery*, 6(1):29–40, 2007.

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

[27] Jeremy S Edwards and Bernhard O Palsson. Systems properties of the haemophilus influenzaerd metabolic genotype. *Journal of Biological Chemistry*, 274(25):17410–17416, 1999.

[28] Adam M Feist and Bernhard Ø Palsson. The growing scope of applications of genome-scale metabolic reconstructions using escherichia coli. *Nature biotechnology*, 26(6):659–667, 2008.

[29] 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.

[30] Weng H Chan, Mohd S Mohamad, Safaai Deris, and Rosli M Illias. A review of computational approaches for in silico metabolic engineering for microbial fuel production. *Current Bioinformatics*, 8(2):253–258, 2013.

[31] Priti Pharkya, Anthony P Burgard, and Costas D Maranas. Exploring the overproduction of amino acids using the bilevel optimization framework optknock. *Biotechnology and bioengineering*, 84(7):887–899, 2003.

[32] Jin Hwan Park, Kwang Ho Lee, Tae Yong Kim, and Sang Yup Lee. Metabolic engineering of escherichia coli for the production of l-valine based on transcriptome analysis and in silico gene knockout simulation. *Proceedings of the National Academy of Sciences*, 104(19):7797–7802, 2007.

[33] Iwei Yeh, Theodor Hanekamp, Sophia Tsoka, Peter D Karp, and Russ B Altman. Computational analysis of plasmodium falciparum metabolism: organizing genomic information to facilitate drug discovery. *Genome Research*, 14(5):917–924, 2004.

[34] Christopher S Henry, Matthew DeJongh, Aaron A Best, Paul M Frybarger, Ben Linsay, and Rick L Stevens. High-throughput generation, optimization and analysis of genome-scale metabolic models. *Nature Biotechnology*, 28(9):977–982, 2010.

[35] Frederick S Hillier and Gerald J Lieberman. *Introduction to operations research*. McGraw-Hill Education, 10th edition, 2012.

[36] J Lundgren, M Rönnqvist, and P Värbrand. *Optimization*. Studentlitteratur, 2010.

[37] Ove Øyås. Identification of antimicrobial drug targets from robustness properties of metabolic networks. Master's thesis, NTNU, 2015.

[38] Scott Kirkpatrick, C Daniel Gelatt, Mario P Vecchi, et al. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

[39] E.H.L. Aarts and J.K. Lenstra. *Local Search in Combinatorial Optimization*. Princeton University Press, 2003.

[40] Lester Ingber. Simulated annealing: Practice versus theory. *Mathematical and Computer Modelling*, 18(11):29–57, 1993.

[41] Dario Floreano and Claudio Mattiussi. *Bio-inspired Artificial Intelligence: theories, methods, and technologies*. MIT press, 2008.

[42] Emile Aarts, Jan Korst, and Wil Michiels. Simulated annealing. In *Search Methodologies*, pages 265–285. Springer, 2014.

[43] Nico Ulder, Emile Aarts, Hans-Jürgen Bandelt, Peter van Laarhoven, and Erwin Pesch. Genetic local search algorithms for the traveling salesman problem. *Parallel Problem Solving from Nature*, pages 109–116, 1991.

[44] E Burke, James Newall, and R Weare. A memetic algorithm for university exam timetabling. *Practice and Theory of Automated Timetabling*, pages 241–250, 1996.

[45] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

[46] Emile HL Aarts and Jan HM Korst. Simulated annealing. *ISSUES*, 1:16, 1988.

[47] Ernst Ising. Beitrag zur theorie des ferromagnetismus. *Zeitschrift für Physik A Hadrons and Nuclei*, 31(1):253–258, 1925.

[48] Barry M McCoy and Tai Tsun Wu. *The two-dimensional Ising model*. Courier Corporation, 2014.

[49] Barry A Cipra. An introduction to the ising model. *American Mathematical Monthly*, 94(10):937–959, 1987.

[50] Lilian Witthauer and Manuel Dieterle. The phase transition of the 2d-ising model. *Summer Term*, 2007.

[51] Adam M Feist, Markus J Herrgård, Ines Thiele, Jennie L Reed, and Bernhard Ø Palsson. Reconstruction of biochemical networks in microorganisms. *Nature Reviews Microbiology*, 7(2):129–143, 2009.

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

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

[54] Irina Borodina and Jens Nielsen. From genomes to in silico cells via metabolic networks. *Current Opinion in Biotechnology*, 16(3):350–355, 2005.

[55] Richard A Notebaart, Frank HJ Van Enckevort, Christof Francke, Roland J Siezen, and Bas Teusink. Accelerating the reconstruction of genome-scale metabolic networks. *BMC bioinformatics*, 7(1):296, 2006.

[56] Jennifer L Reed, Iman Famili, Ines Thiele, and Bernhard O Palsson. Towards multidimensional genome annotation. *Nature Reviews Genetics*, 7(2):130–141, 2006.

[57] Jeremy S Edwards, Rafael U Ibarra, and Bernhard O Palsson. In silico predictions of escherichia coli metabolic capabilities are consistent with experimental data. *Nature Biotechnology*, 19(2):125–130, 2001.

[58] James E Bailey. Complex biology with no parameters. *Nature Biotechnology*, 19(6):503–504, 2001.

[59] Areejit Samal, João F. Matias Rodrigues, Jürgen Jost, Olivier C. Martin, and Andreas Wagner. Genotype networks in metabolic reaction spaces. *BMC Systems Biology*, 4(1):30, 2010.

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

[61] Adilson E Motter, Natali Gulbahce, Eivind Almaas, and Albert-László Barabási. Predicting synthetic rescues in metabolic networks. *Molecular Systems Biology*, 4(1):168, 2008.

[62] Patrick F Suthers, Madhukar S Dasika, Vinay Satish Kumar, Gennady Denisov, John I Glass, and Costas D Maranas. A genome-scale metabolic reconstruction of mycoplasma genitalium, i ps189. *PLoS Comput Biol*, 5(2):e1000285, 2009.

[63] Joao F Matias Rodrigues and Andreas Wagner. Evolutionary plasticity and innovations in complex metabolic reaction networks. *PLoS Comput Biol*, 5(12):e1000613, 2009.

[64] Andreas Wagner, Vardan Andriasyan, and Aditya Barve. The organization of metabolic genotype space facilitates adaptive evolution in nitrogen metabolism. *Journal of Molecular Biochemistry*, 3(1), 2014.

[65] Mathias Ganter, Thomas Bernard, Sébastien Moretti, Joerg Stelling, and Marco Pagni. Metanetx.org: a website and repository for accessing, analysing and manipulating metabolic networks. *Bioinformatics*, page btt036, 2013.

[66] Christian P Robert. *Monte carlo methods*. Wiley Online Library, 2004.

[67] The Python Software Foundation. Python 2.7, 2010.

[68] Ali Ebrahim, Joshua A. Lerman, Bernhard O. Palsson, and Daniel R. Hyduke. Cobrapy: Constraints-based reconstruction and analysis for python. *BMC Systems Biology*, 7(1):74, 2013.

[69] Gurobi optimizer reference manual, 2016.

[70] Benjamin J Bornstein, Sarah M Keating, Akiya Jouraku, and Michael Hucka. Libsbml: an api library for sbml. *Bioinformatics*, 24(6):880–881, 2008.

[71] G Varoquaux and O Grisel. Joblib: running python function as pipeline jobs. *packages. python. org/joblib*, 2009.

[72] Norman Matloff and Francis Hsu. Tutorial on threads programming with python. *University of California*, 2007.

[73] Melissa S Cline, Michael Smoot, Ethan Cerami, Allan Kuchinsky, Nerius Landys, Chris Workman, Rowan Christmas, Iliana Avila-Campilo, Michael Creech, Benjamin Gross, et al. Integration of biological networks and gene expression data using cytoscape. *Nature Protocols*, 2(10):2366–2382, 2007.

[74] Sébastien Moretti, Olivier Martin, T Van Du Tran, Alan Bridge, Anne Morgat, and Marco Pagni. Metanetx/mnxref–reconciliation of metabolites and biochemical reactions to bring together genome-scale metabolic networks. *Nucleic Acids Research*, 44(D1):D523–D526, 2016.

[75] Ines Thiele, Thuy D Vo, Nathan D Price, and Bernhard Ø Palsson. Expanded metabolic reconstruction of helicobacter pylori (iit341 gsm/gpr): an in silico genome-scale characteri-

zation of single-and double-deletion mutants. *Journal of Bacteriology*, 187(16):5818–5830, 2005.

[76] Neema Jamshidi and Bernhard Ø Palsson. Investigating the metabolic capabilities of mycobacterium tuberculosis h37rv using the in silico strain inj 661 and proposing alternative drug targets. *BMC Systems Biology*, 1(1):26, 2007.

[77] 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-ccale metabolic reconstruction for escherichia coli k-12 mg1655 that accounts for 1260 orfs and thermodynamic information. *Molecular Systems Biology*, 3(1):121, 2007.

[78] Nikos Vlassis, Maria Pires Pacheco, and Thomas Sauter. Fast reconstruction of compact context-specific metabolic network models. *PLoS Comput Biol*, 10(1):e1003424, 2014.

Appendix A

Simulated Annealing Code

The following Python script is the SA method that were used in this project.

- Python specific keywords are written in magenta.
- Comments are written in green.
- Words or phrases with underscores are variable names.
- Words or phrases written in uppercase and lowercase letters are function names.

```python
__author__ = 'Siri'
import math
import random
import cobra

T_r = 50 # set start temperature for reactions
T_b = 0.5 # set temperature for blocked reactions
optimal_reactions = reactionsInOriginalModel()
optimal_blocked = blockedInOriginalModel()
num_iterations = optimal_reactions

while T_r > 0.5:
    for i in range(num_iterations):
        last_num_reactions = new_num_reactions
        if random.random() > 0.5: # 50% probability for adding/removing reaction
```

```
            new_num_reactions += 1
            delta_r = math.fabs(optimal_reactions - new_num_reactions)
            probability_r = math.exp(-delta_r/T_r) # Metropolis criterion
            if new_num_reactions < optimal_reactions:
                new_num_reactions = addReaction()
            else:
                if random.random() < probability_r:
                    new_num_reactions = addReaction()
        else:
            new_num_reactions -= 1
            delta_r = math.fabs(optimal_reactions-new_num_reactions)
            probability_r = math.exp(-delta_r/T_r) # Metropolis criterion
            if new_num_reactions > optimal_reactions:
                new_num_reactions = removeReaction()
            else:
                if random.random() < probability_r:
                    new_num_reactions = removeReaction()
        last_blocked = new_blocked
        new_blocked = numberOfBlockedReactions()
        if new_blocked > optimal_blocked and new_blocked > last_blocked:
            delta_b = math.fabs(optimal_blocked-new_blocked)
            probability_b = math.exp(-delta_b/T_b) # Metropolis criterion
            if random.random() > probability_b: # changes not accepted
                if last_blocked < new_blocked:
                    new_num_reactions = resetModelRemoveReaction()
                    new_blocked = last_blocked
                if last_blocked > new_blocked:
                    new_num_reactions = resetModelAddReaction()
                    new_blocked = last_blocked
    T_r = T_r * 0.9
```

Appendix B

Growth Medium for iIT341

Table B.1 shows the standard uptake reactions for iIT341. The following is included in the table; name of metabolites, BiGG reaction ids, lower bound values, upper bound values, MNX reaction ids, and compartment specific reaction ids. MNX reaction id is the generic id system for reactions in models downloaded from MetaNetX. This id refer to the reactions only, and are not implying in which compartment the metabolites are located. On the other hand, the compartment specific ids specify where in the system the reactions and their metabolites are located. When different growth media were constructed, these ids were used to identify the reactions of interest. All reactions in Table B.1 transport metabolites from the extracellular region compartment to the model boundary compartment. BiGG reaction ids are most common in genome-scale modelling, and are therefore included as well [74].

Table B.1: The uptake reactions that are standard for the iIT341 *H. pylori* model. The name of the metabolites being transported, the reactions' BiGG id, lower and upper bounds, MNX id, and compartment specific id are included. It is the latter id system that are used when identifying reactions in growth media construction.

metabolite	BiGG id	l. b.	u. b.	MNX reaction id	comp. specific id
D-alanine	EX_ala_D(e)	-10	1000	MNXR1425	RC96712E3
L-alanine	EX_ala_L(e)	-10	1000	MNXR639	R555722AC
L-arginine	EX_arg_L(e)	-10	1000	MNXR788	RD6F7BF7B
Fe^{2+}	EX_fe2(e)	-1000	1000	MNXR2212	R15DF7064
Fe^{3+}	EX_fe3(e)	-1000	1000	MNXR2228	R8FDD65E8
H^{+}	EX_h(e)	-1000	1000	None	R47177312
L-histidine	EX_his_L(e)	-10	1000	MNXR2975	R7BB6BC6B
L-isoleucine	EX_ile_L(e)	-10	1000	MNXR3097	REEFDC248
L-leucine	EX_leu_L(e)	-10	1000	MNXR3225	R82521F85
L-methionine	EX_met_L(e)	-10	1000	MNXR3506	RA7DC31A9
Ammonia	EX_nh4(e)	-1000	1000	MNXR3814	R65404FB9
Oxygen	EX_o2(e)	-12	0	MNXR3931	RA65CB13F
Protoheme	EX_pheme(e)	-10	1000	MNXR78986	R7EAE41AC
Phosphate	EX_pi(e)	-1000	1000	MNXR4294	R596FDB0E
Pimelate	EX_pime(e)	-10	1000	MNXR4288	RFBB325D
Sulfate	EX_so4(e)	-1000	1000	MNXR4873	R4B5145FC
Thiamin	EX_thm(e)	-10	1000	MNXR5053	RE47D2287
L-valine	EX_val_L(e)	-10	1000	MNXR5325	RF382895C
S-adenosyl-L-homocysteine	sink_ahcys_c	-1000	1000	MNXR580	R1DBDE290
S-adenosyl-4-methylthio-2-oxobutanoate	sink_amob	-1000	1000	MNXR82312	R6D6AB2A7

Appendix C

Program Parameters

The parameters in the SA algorithm are crucial for the method's performance. The reasons for the parameter values used in this project are presented in the following sections.

C.1 Temperature Parameters

The temperature parameters are important for accepting new solutions. In this project, the temperature parameter that controlled the total number of reactions (T_r) was lowered and the temperature parameter that controlled the number of blocked reactions (T_b) was kept at a constant level.

C.1.1 Parameter T_r

To allow large fluctuations in the number of reactions, the initial value of T_r needs to be high. Nevertheless, it should not be too high, as shown in Figure D.6. For T_r values above 40, the probability does not increase much. Therefore, it is unnecessary to set the temperature parameter much higher than this. Based on this figure, T_r was set to 50 in the initialization process. This corresponds to a probability of 98 % to accept a worse solution.

This predictability when it comes to the relationship between T_r and the corresponding probability is caused by the fact that the number of reaction can only differ by 1 in each iteration. The equation used to plot the graph in Figure D.6 is:

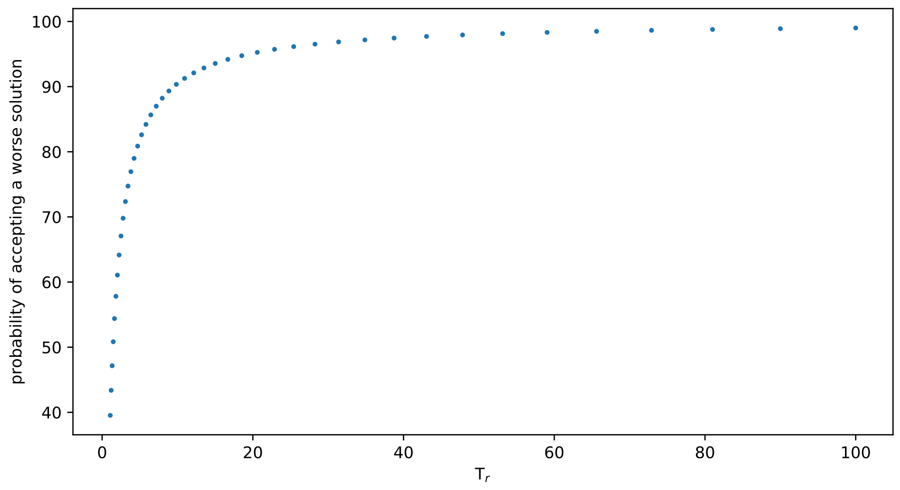


Figure C.1: The probability of accepting worse solutions as a function of T_r, given in Equation C.1. From $T_r \approx 40$ to $T_r = 100$ there is no big difference in the probability.

$$P(\text{accept new solution}) = \exp\left(-\frac{1}{T_r}\right) \tag{C.1}$$

where ΔR from Equation 3.1 is replaced with 1.

C.1.2 Parameter T_b

For keeping the proportion of blocked reaction to approximately the same level as in the original networks, the value of T_b had to be low. After testing with different values it became apparent that T_b had to be below 5. It is not possible to make a plot like that in figure D.6 because the number of blocked reactions does not differ with the same number in each iteration.

After testing random metabolic generation for all T_b values between 0.5 and 5, it became clear that 0.5 was necessary to keep the percentage of blocked reactions low. See also Figure 4.5 in Section 4.2 for a illustration of how the proportion of blocked reactions differ as a function of T_b.

C.1.3 Cooling Schedule

The cooling schedule is a critical part of the SA algorithm. If the temperature parameter is lowered too fast, the result often gets trapped in a local optimum. If the temperature parameter is lowered excessively slow, the computation time will increase drastically.

It may be easier to understand the following reasoning by thinking about the program as two separate parts, each part having different requirements for optimality and efficiency. The first part is called the randomization part and the second is named the optimization part.

- In the randomization part, the requirement is that T_r should be high so that as many as possible of the reactions will be replaced. Here, T_r can be lowered more rapidly than in the optimization part. The solutions will never get trapped in local optima in this part because T_r still has a high value. An unnecessary amount of time will also be spent in this part if T_r is lowered too slow.

- In the optimization part, T_r should be low so that the reactions can approach the optimal value. To ensure that global optimum is reached, T_r needs to be lowered slowly. Therefore, most of the time should be spent in the optimization part.

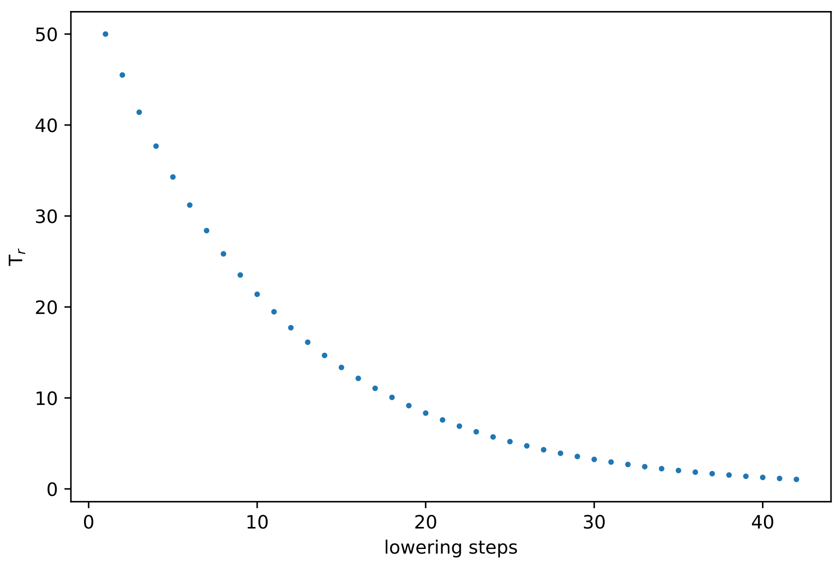
Based on these requirements, it was not appropriate to lower the parameter with the same number in each part. Instead, the parameter was lowered by 10 % each time, which fulfilled both requirements mentioned. This procedure ensures that T_r is lowered more slowly when approaching its stop value, as shown in Figure C.2(a). For comparison, Figure C.2(b) shows how the cooling schedule would look like if T_r was lowered by the same number in each iteration. Based on the requirements mentioned above, this would be a non-optimal way to perform this schedule.

C.1.4 Stop Criterion

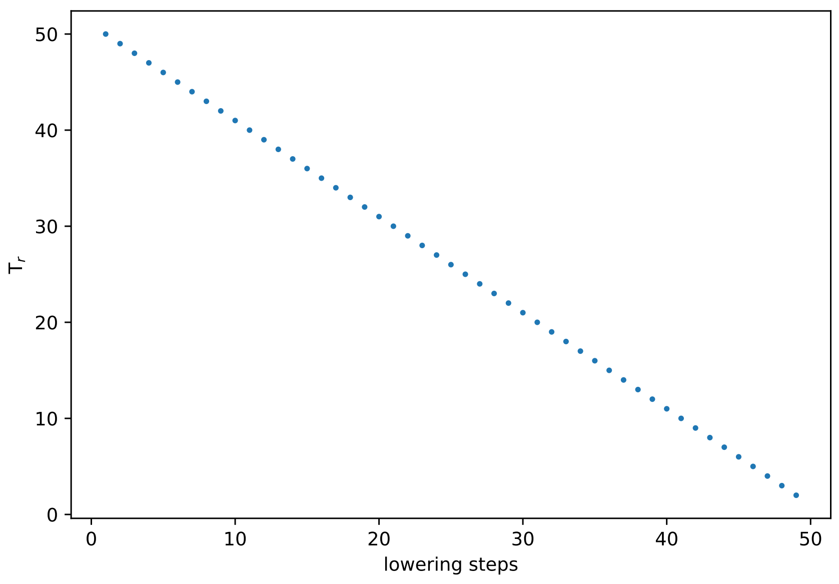
The stop criterion decides for which value of T_r the simulation should be stopped. The results in Figure 4.4 established that the optimal number of reactions was reached around $T_r = 1$.

C.2 Number of Iterations

From the description of the Metropolis algorithm, which is explained in Section 2.2.2, it was stated that thermal equilibrium was reached for each temperature when the number of iterations was the same as the number of particles in the solid. Based on this, the number of iterations in this project was set as the same number as reactions in the metabolic networks.



(a) T_r lowered by 10 %



(b) T_r lowered by 1

Figure C.2: Two different approaches for lowering T_r. The x axis shows the number of lowering steps and the y axis shows the value of T_r. In C.2(b), T_r was lowered by 1. In C.2(a), T_r was lowered by 10 %.

Appendix D

Metabolic Networks

This appendix includes visualization of both the original metabolic networks and one of the random metabolic networks from each of the three models iIT341 *H. pylori*, iNJ661 *M. tuberculosis*, and iAF1260 *E. coli*. The networks are visualized as a bipartite graph using Cytoscape. Blue nodes illustrate metabolites and pink nodes illustrate reactions. Note that reaction clusters tend to form for the random networks, which explains the large fraction of essential reactions in the networks. The separate reactions arranged at the bottom of the random network figures are mainly inactive uptake reactions.

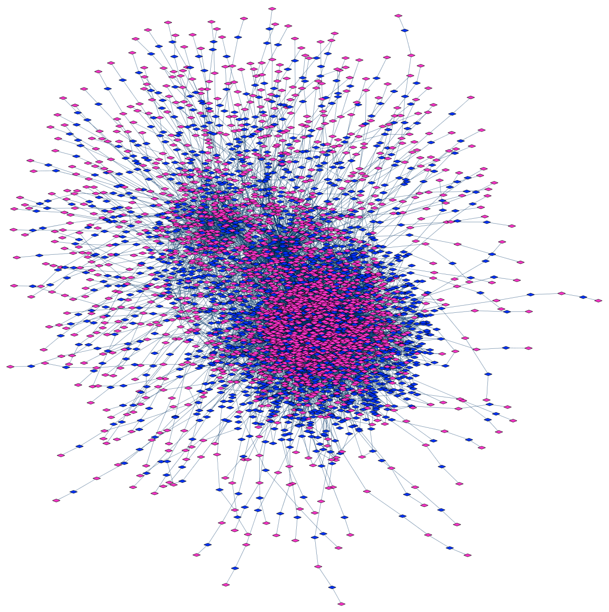


Figure D.1: Visualization of the original metabolic network of the iAF1260 *E. coli* model.

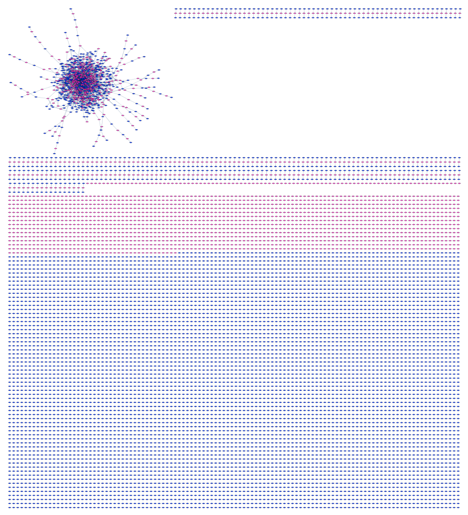


Figure D.2: Visualization of the random metabolic network of the iAF1260 *E. coli* model.

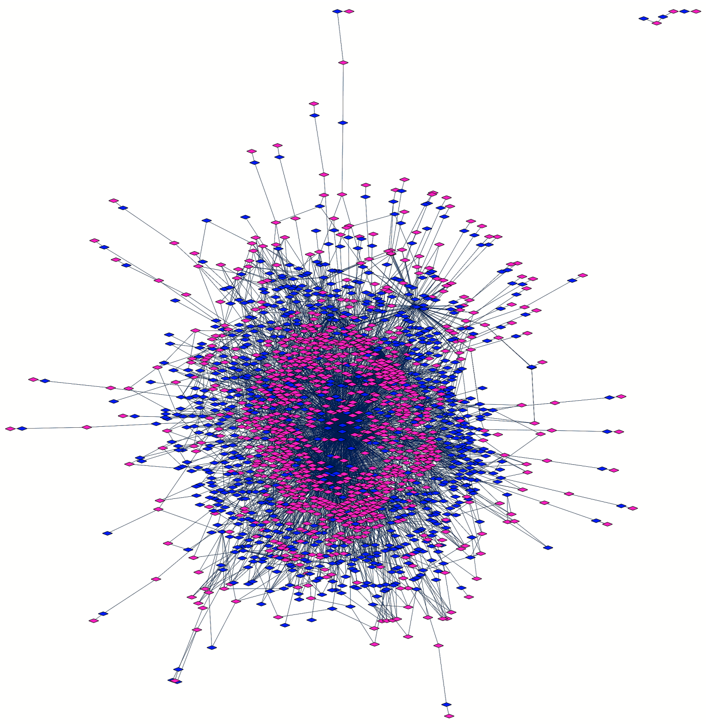


Figure D.3: Visualization of the original metabolic network of the iNJ661 *M. tuberculosis* model.

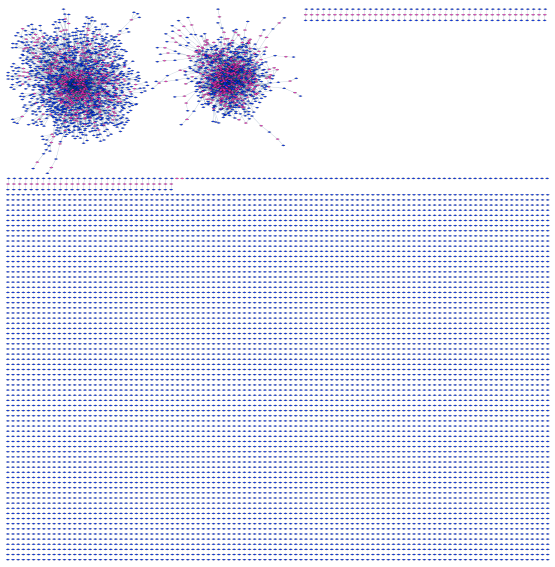


Figure D.4: Visualization of the random metabolic network of the iNJ661 *M. tuberculosis* model.

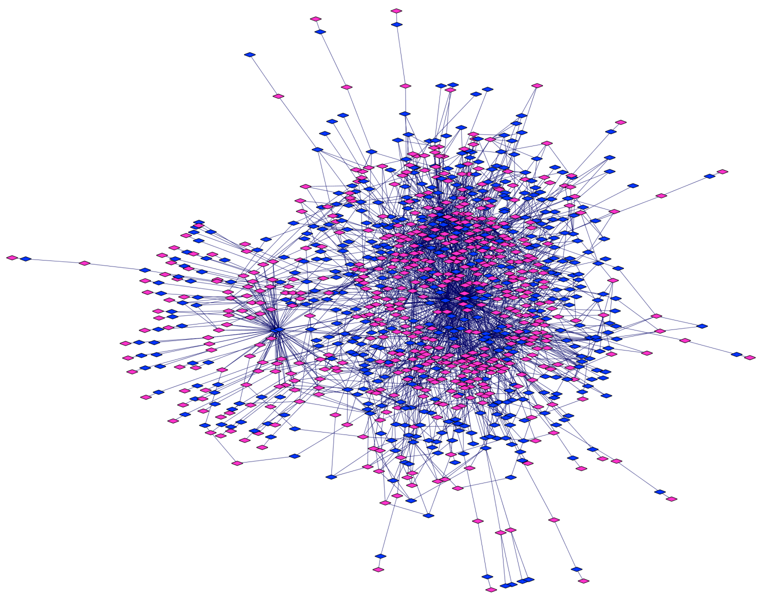


Figure D.5: Visualization of the original metabolic network of the iIT341 *H. pylori* model.

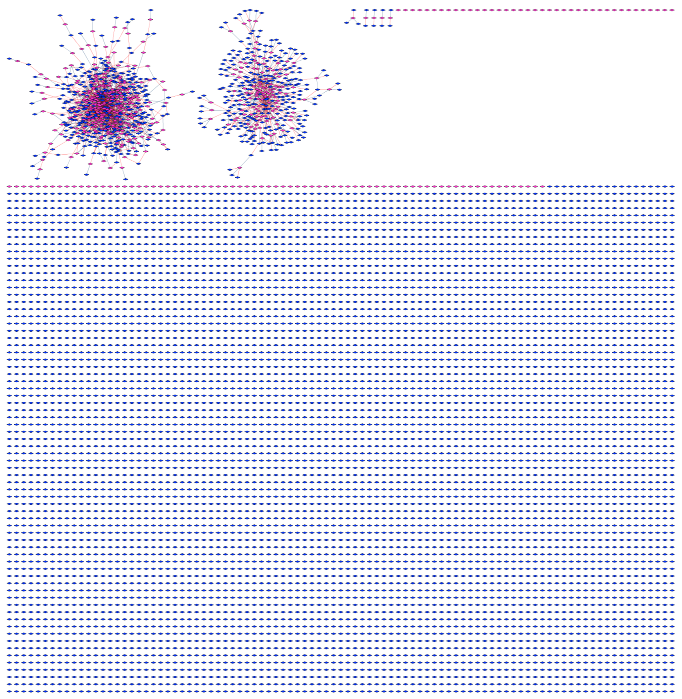


Figure D.6: Visualization of the random metabolic network of the iIT341 *H. pylori* model.