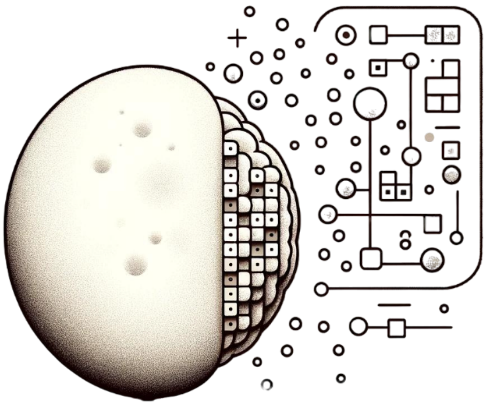

Improving Genome Scale Metabolic Models using Gene Regulatory Networks

Utilizing transcription data and transcription factor interactions for improved modeling capabilities

Master's thesis in Biotechnology

Liam Davidsson

DEPARTMENT OF LIFE SCIENCES

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2024
www.chalmers.se

Master's thesis 2024

Improving Genome Scale Metabolic Models using Gene Regulatory Networks

Utilizing transcription data and transcription factor interactions for improved modeling capabilities

LIAM DAVIDSSON

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Life Sciences
Chalmers University of Technology
Gothenburg, Sweden 2024

Improving Genome Scale Metabolic Models using Gene Regulatory Networks
Utilizing transcription data and transcription factor interactions for improved modeling capabilities
LIAM DAVIDSSON

Supervisor: Daniel Brunnsåker, Department of Computer Science and Engineering
Examiner: Marija Cvijovic, Department of Mathematical Sciences

Master's Thesis 2024
Department of LIFE Sciences
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Yeast cell being analyzed and digitized. Created with the assistance of DALL · E

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2024

Improving Genome Scale Metabolic Models using Gene Regulatory Networks
Utilizing transcription data and transcription factor interactions for improved modeling capabilities
LIAM DAVIDSSON
Department of LIFE Sciences
Chalmers University of Technology

Abstract

This thesis aims to enhance Genome Scale Metabolic Models (GEMs) by integrating Gene Regulatory Networks (GRNs) to improve modeling capabilities for the yeast *Saccharomyces cerevisiae*. The project involved constructing a pipeline to incorporate gene expression data into GEMs, resulting in a constrained metabolic model with a more diverse and characteristic flux distribution. By utilizing transcription data and transcription factor interactions, the integration aims to provide a more comprehensive understanding of cellular processes. The research demonstrates that incorporating GRNs can enhance the predictive accuracy of GEMs, despite the challenges associated with data complexity and integration methodologies. Potential future work includes upgrading the GRN to a dynamic Bayesian network and exploring the effects of network size on model outcomes.

Keywords:
Genome Scale Metabolic Models, GEM, Gene Regulatory Network, GRN, Co-Regulation, Gene Expression, Machine Learning, Decision Trees, Random Forest.

Acknowledgements

This thesis would not have been possible without the invaluable support and guidance of my supervisor, Daniel Brunnsåker. His willingness to offer me the opportunity to engage in a more data science-intensive project without a propper data science background is highly appreciated. I am deeply grateful for his mentorship and insights throughout the process. I would also like to extend my heartfelt thanks to my friends from the lunchroom, whose camaraderie and support have made the past six months immensely enjoyable. Your companionship has been a source of great motivation and joy. I'm always open to a fika anytime <3. Lastly I'd like to express my sincere appreciation to my examiner, Marija Cvijovic, for taking the time out of her busy schedule to see that my last project as a Chalmers student is duly scrutinized.

Liam Davidsson, Gothenburg, June 2024

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

AUC	Area Under the Curve
CPD	Conditional Probability Distribution
CV	Cross-Validation
DAG	Directed Acyclic Graph
DNA	DeoxyriboNucleic Acid
FBA	Flux Balance Analysis
FN	False Negative
FP	False Positive
FPR	False Positive Rate
GEM	Genome-Scale Metabolic Model
GI	Gini Impurity
GO	Gene Ontology
GRN	Gene Regulatory Network
HT	High Throughput
LOO	Leave-One-Out
MC	Manually Curated
pFBA	Parsimonious Flux Balance Analysis
RF	Random Forest
RNA	RiboNucleic Acid
ROC	Receiver Operating Characteristic
RNG	Random Number Generator
SS_{res}	Sum-of-Squares, residual
SS_{tot}	Sum-of-Squares, total
TF	Transcription Factor
TG	Target Gene
TN	True Negative
TP	True Positive
TPR	True Positive Rate
YPD	Yeast Extract–Peptone–Dextrose

Contents

1
Introduction

1.1 Background

The yeast *Saccharomyces cerevisiae* has for many thousands of years been used as a tool, primarily for fermentation or leavening [1]. More recently the rise of synthetic biology has allowed for introduction of heterologous genes which allows for production of products not normally produced by yeast. A vital tool for the continued development of yeast as a tool has been mathematical models of the inner workings of yeasts, which allows for simulations and predictions regarding behaviour under various conditions. A commonly used type of model is a Genome-Scale Metabolic Model (GEM) where metabolites and reactions in the cell are ordered and annotated with corresponding genes, if applicable [2]. This allows for simulating the rate of flow of metabolites through the different reactions, in this context called the flux, which is then useful for predicting growth rate or product formation given a set of constraints. These models can then further be improved by adding on additional constraints to the solution space, such as enzyme constraints or thermodynamic feasibility. In addition to the development of GEMs, current research is also focusing on the integration of regulatory networks with these models. Gene Regulatory Networks (GRNs) represent the interactions among genes regulating the activation of specific cell functionalities, such as reception of (chemical) signals or reaction to environmental changes [3]. The integration of GRNs with GEMs allows for a more comprehensive understanding of cellular processes, by allowing the regulation of metabolic genes to affect the metabolic model.

Current research

A lot of work has been, and currently is being put into improving many aspects of the models used today. For GEMs there are continuously new models being published and old ones are being updated to include yet more of the processes taking place in the cells [2]. This project used the Yeast 9 GEM, which is one of these recently published models [4]. Recent developments regarding GRNs include a stronger focus on improving signal-to-noise ratios in the data used to infer the network [5], the use of multiple models to complement each other [6] and a larger focus on single cell multi-omics data which better captures the state of a cell, and allows for better models of early cell development at the cost of increased noise and lack of true biological replicates [7].

1.2 Aim

The aim of this project was to construct a model of the yeast *Saccharomyces cerevisiae* which integrates a GRN based on expression data and a GEM for increased fidelity. A quick list of the main parts of the project are shown below.

1. Construction of a valid GEM for yeast growing in standard media. The base model being the Yeast 9 model [4].
2. Construct a co-regulatory network to mimic the effects of gene regulation in yeast. The network is based off of co-expression links in gene expression data found through machine learning and can if desired be complemented with manually curated links.
3. Integration of the two models to allow for a more holistic model of yeast metabolism. Here the outputs from the GRN are converted to constraints of the GEM.
4. Iterative improvements of the GRN using simple methods and evaluation techniques.

1.3 Limitations

In this project the main focus lies with co-regulation and the effect of transcription factors and the approach used is therefore not suited for all types of GRN generation. The project was intended to be a 30 credit master's thesis and some venues of improvement had to be re-prioritized due to time constraints. The process of constructing networks and adapting the models to real life conditions are also highly dependent on the availability of large amounts of relevant data, which is problematic when generating some types of models.

2

Theory

2.1 Genome Scale Metabolic Models

The basis of a GEM is to mathematically describe the reactions in an organism in a stoichiometry-based way while maintaining the mass and charge balances. The first GEM was constructed for *Haemophilus influenzae* and published in the year 2000 [8]. Ever since a growing number of ever more complex models for an increasing number of organisms are being developed [2]. The GEMs have been instrumental in providing a comprehensive understanding of the metabolic capabilities of an organism. The systems of linear equations set up by a GEM will result in more than one solution vector. Combining all vectors of possible solutions allows for the construction of solution spaces where each point in space represents a combination of fluxes that satisfies the equations of the model. As more and more different types of data are incorporated into the model, such as enzyme availability or kinetic properties, the solution space can be shrunk as a result, ideally homing in on values matching those found in nature under those conditions. This shrinkage of solution space allows for more accurate predictions regarding cellular phenotypes under varying environmental and genetic conditions.

The Flux Balance Analysis (FBA), a widely used computational method, is often employed to analyze GEMs. FBA uses linear optimization to identify the optimal metabolic flux distribution that maximizes or minimizes the flow through (flux) of some reaction in the model (referred to as an objective function), such as growth rate or production of a specific metabolite. The most common unit used for flux is mmol/g(dry cell weight)/h which via inclusion of the cell weight allows for generalization between single cells and much larger populations [9]. The FBA approach has been successfully used to predict growth rates, substrate uptake rates, and by-product secretion rates of various organisms under different conditions [10]. Some assumptions are made when solving the FBA, the foremost one being that steady-state is assumed. Rephrasing the steady-state assumption as a mathematical constraint yields Equation 2.1:

$$S \cdot v = 0 \tag{2.1}$$

Where S represents the stoichiometric matrix, which has all included reactions as columns and all included metabolites as rows, values indicating the metabolites stoichiometry in a given reaction. v represents the vector of reaction fluxes that satisfies the equation (see Figure 2.1). This methodology of constructing matrices where the

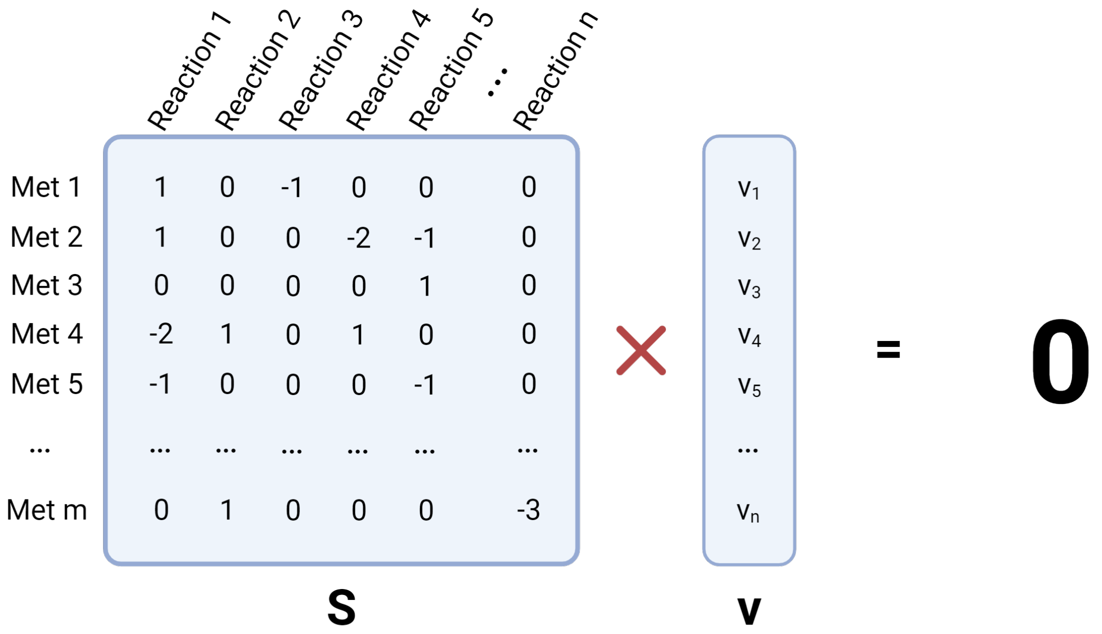

Figure 2.1: Representation of the metabolic network under the steady-state assumption. Stoichimoteric matrix (S), vector of fluxes that satisfies condition (v) [12].

stoichiometry constrains the flow of metabolites is the pillar on which FBA operates. Other, more theory based approaches/models exist where the focus instead lies on kinetic parameters and biophysical equations, the former of which may be hard to accurately measure [11]. After defining the initial stoichiometric constraints the FBA also needs an objective function to optimize for. The most common objective function used is biomass formation, expressed as a pseudo-reaction which consumes biomass constituents [13]. The choice of reaction, and more importantly setting it as the objective, mimicks the selective pressure on organisms to maximize growth rate. Since the constraints create a solution space where all possible points within are mathematically feasible solutions, the objective function, in the case of biomass production, helps to find the biologically relevant points within. With all constraints defined as systems of linear equations, which are quite efficiently solved using linear optimization, the point, or set of points, whose n dimensional coordinates represents the fluxes for the n reactions, symbolizes the solution to the problem [14]. Shown below in Figure 2.2 is what such a solution space could look like.

If given more data about the organism the solution space could be further reduced which would (hopefully) approach the states found in experimental data. Given a accurate enough model, the results of genetic modifications or other disturbances can be estiamated *in silico* without the need for time consuming experimental verification.

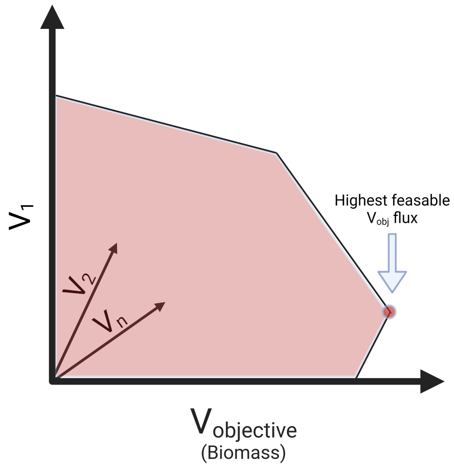

Figure 2.2: Lower dimensional representation of solution space created by linear equations. The point which maximizes the objective function is marked [12].

There are also variations of the FBA, such as the Parsimonious Flux Balance Analysis (pFBA). The difference between pFBA and a regular FBA is that pFBA assumes that there is not only a selective pressure for highest exponential growth among organisms, but also a pressure for efficient enzyme usage which frees up resources for other things [15]. This assumption means that the solution provided by pFBA not only maximizes the objective function, like a regular FBA does, but provides the flux distribution with the lowest total flux which enables this maximized value.

Worth noting is that, much like all models, the accuracy of GEMs and their predictions are heavily reliant on the quality and completeness of the data used to construct it. The inclusion of high-throughput omics data can be a great tool to address incompleteness but suffers from generally being of lower quality, leading to more type 1 and type 2 errors, than manually curated data [16]. In this context however, it must be mentioned that *Saccharomyces cerevisiae* is one of the most well studied organisms. Therefore it does not suffer from a lack of annotation which is often a problem in other organisms.

2.2 Gene Regulatory Networks

A Gene Regulatory Network (GRN) is a representation of molecular regulation which takes place in a cell. A regulator could be DNA, RNA, proteins or complexes of one ore more types that directly or indirectly determine cell function. While there are many different types of regulators, one of the major ones are Transcription Factors (TFs) which are proteins that interact with the promoter regions of coding DNA (often referred to as Target Genes, TG) to either increase or decrease the rate of transcription for one or a set of TGs [17]. Depending on the scope a GRN could

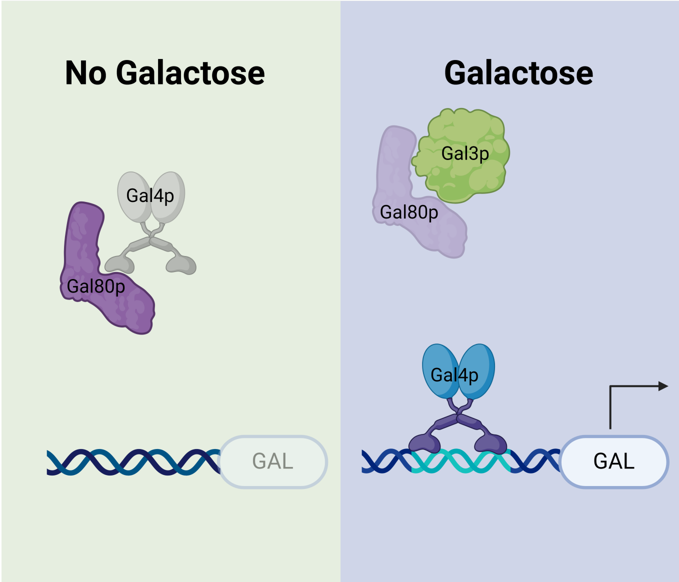

Figure 2.3: Figure showing the interactions between Gal regulators and the GAL genes [12].

be defined as contained in a single cell, which senses stimuli and responds or in the case of multicellular organisms include larger signaling cascades and inter-cellular signaling. An example could be the interplay taking place when yeast *S. cerevisiae* consumes galactose. In the yeast *S. cerevisiae*, the GAL genes are responsible for the metabolism of galactose. When glucose is present, the GAL genes are repressed, but when galactose is present and glucose is absent, the GAL genes are activated [18]. This is accomplished by the GAL genes being controlled by the TF Gal4p which activates transcripton. Gal4p is normaly repressed by Gal80p which binds to the TF, preventing it from acting as a TF. In the presence of galactose the Gal3p transcriptional regulator binds to Gal80p, thus preventing Gal80p from deactivating Gal4p which can now increase transcription of the GAL genes, allowing for galactose metabolism. This is just a short example of one of many different interactions that constantly occur in cells and allow them to adapt to their environments and is illustrated below in Figure 2.3.

2.3 Bayesian Models

2.3.1 The concept of Bayesian statistics

The basic concept of Bayes theorem is that evidence can be used to estimate the probability of a state, case or event based on prior data of related events. This is a vital tool when constructing some types of statistical models where. An example of the application of Bayes theorem (also reffered to as Bayes Rule) is self-tests for viral infections, which have become more common in recent times. When being tested one might wonder, given that they got a positive result on a test, what is the probability that they actually are infected. Bayes' theorem helps to answer this by proposing Equation 2.2.

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} \tag{2.2}$$

In the equation above (2.2) P(A|B) represents the probability of A given that B is true, P(B|A) is the probability of B given A being true and P(A) & P(B) represent the probabilities of observing states A & B and is in this context referred to as prior probabilities.

In the context of medical testing the equation can be rewritten as seen in Equation 2.3 [19].

$$P(I^+|T^+) = \frac{P(I^+) \cdot P(T^+|I^+)}{P(T^+)} \tag{2.3}$$

Where I^+ represents infection and T^+ represents a positive test result. To solve equation 2.3 the probability of a test being positive needs to be calculated. This can be calculated as follows in Equation 2.4 [19].

$$P(T^+) = [P(I^+) \cdot P(T^+|I^+)] + [P(I^-) \cdot P(T^+|I^-)] \tag{2.4}$$

In the equation above I^- represents not being infected, and as such $P(T^+|I^-)$ is the probability of testing positive while *not* carrying the infection. The two equations 2.3 and 2.4 are combined to form Equation 2.5 [19].

$$P(I^+|T^+) = \frac{P(I^+) \cdot P(T^+|I^+)}{[P(I^+) \cdot P(T^+|I^+)] + [P(I^-) \cdot P(T^+|I^-)]} \tag{2.5}$$

Assume that the infection has a prevalence of 1/100 in a population and the test has a sensitivity (True positives given infection) of 95% and a specificity (True negatives given non-infected) of 95%. The prior probability of being infected $P(I^+)$ is the prevalence and the probability of a positive test given infection $P(T^+|I^+)$ is the sensitivity. The remaining terms $P(I^-)$ and $P(T^+|I^-)$ are the probabilities of the prevalence and specificity **not** being true, i.e $1 - Prevalence$ & $1 - Specificity$. Which finally leads to Equation 2.6 which can be solved for the updated probability using the values given before.

$$P(I^+|T^+) = \frac{Prevalence \cdot Sensitivity}{Prevalence \cdot Sensitivity + (1 - Prevalence) \cdot (1 - Specificity)} = 0.16 \tag{2.6}$$

The result here is as stated before an updated probability, in this context referred to as the posterior probability which conveys that the likelihood of someone being infected by the imaginary virus, given that the also imagined test came back positive is approximately 16%.

While a 16% likelihood of being infected sounds rather low in comparison to the 0.95% specificity and sensitivity but the key here is that if 10 000 people are tested there are on average 100 infected individuals in that crowd and 9 900 healthy ones. Of those 100 who are infected 5 will get a false negative result due to the 95% sensitivity, and 495 individuals from the healthy group will get false positive results stemming from the 0.95 specificity. So out of the total population, 590 will get a positive result and 95 of those are actually infected, leading to a $\frac{95}{590} = 0.16$ proportion who are actually sick given a positive test.

Where the Bayesian way of thinking about statistics really shines is its basis in updating prior probabilities which can easily be applied to the example above to calculate the probability of someone having the infection given two positive tests. Using the calculated 0.16 as a prior probability, instead of the 1/100 used before, the new posterior probability of having the infection given two positive tests is 0.78 or 78%.

2.3.2 Bayesian Inference

Bayesian Inference is the process of utilizing a Bayesian view of statistics to gradually fit a set of parameters to match the given data. This stands in contrast to typical frequentist inference, where proportions of outcomes in the data are seen as the distribution from which the data came. As such, using purely frequentist statistics, tossing a coin 10 times and getting heads 7 times would lead to the conclusion that if you flip a coin you have a 70% chance of getting head given the data (shown in 2.4 A). Bayesian inference would instead start of with a base assumption (prior), for example "a coin has approximately a $\frac{1}{2}$ chance of landing on tails" and produce a posterior probability based on outcomes [20]. Flipping the same coin 10 times and getting the same results as before would using Bayesian inference only shift the probability distribution, in this case best described by a beta distribution, towards the observed value (0.7) [21].

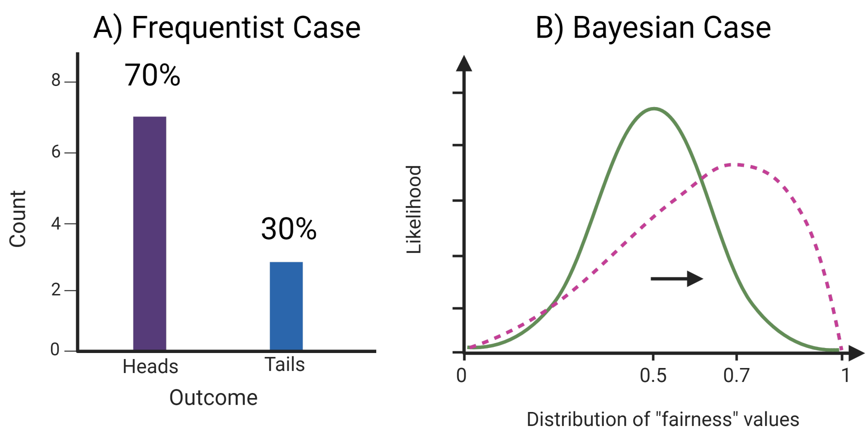

Figure 2.4: Illustration of the frequentist statistics (A) and Bayesian statistics (B) where both are estimating the fairness of a coin being flipped [12].

2.3.3 Bayesian Networks

One particular application of Bayesian methods are Bayesian networks, sometimes also referred to as Belief Networks, which are a type of graphical model which uses a Directed Acyclic Graph (DAG) to represent the probabilistic relationships among a set of variables. Each node in the graph represents a random variable and the edges denote probabilistic dependencies between the variables and the lack of an edge between two nodes implies conditional independence between the involved variables. The terms "probabilistic dependency" and "conditional independence" simply refer to if the occurrence of one event is dependent on another. The usual example of dependent events are picnics and rain, where the likelihood of a picnic are in most cases inversely related to the chances of rain as most prefer to not sit outside in the rain. Shown below in Figure 2.5 is a representation of the example above. Note how while the picnic is ruined by both the rain and the wet grass, the wet grass, and lack of a picnic, does not dictate that it must be raining.

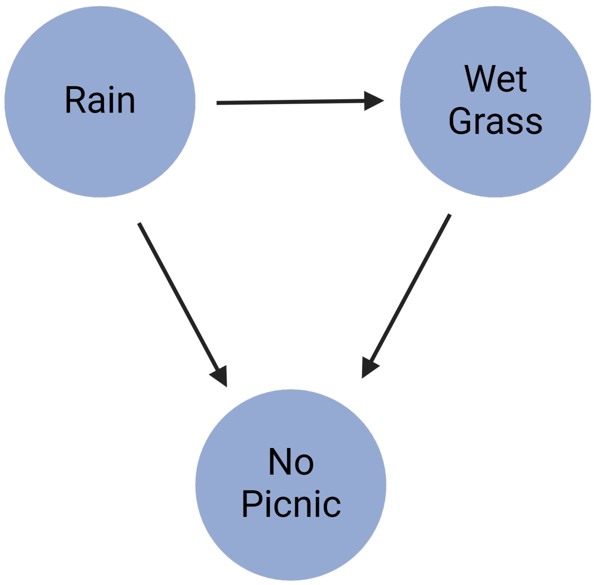

Figure 2.5: Simple representation of a DAG. Edge direction shows that both the "Wet Grass" and "No Picnic" are dependent on the "Rain" node, and that the first two nodes both contribute to the "No Picnic" node [12].

Each node in a Bayesian network has a corresponding Conditional Probability Distribution (CPD) that quantify its dependencies. For any node X_i the CPD is constructed as $P(X_i|Parents(X_i))$ where $Parents(X_i)$ are the given states of the parent nodes in the DAG. These CPDs are what forms the parameters of any Bayesian network. The joint probability distribution of a set of states can be expressed as the product of the distributions of the substates $X_1, X_2, ..., X_n$ as shown in equation 2.7

[22]:

$$P(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} P(X_i | Parents(X_i)) \tag{2.7}$$

Given this equation any possible state of the Bayesian network can be given a probability of being in that specific state, which is the product of all the probabilities of all the nodes in the network being in their current states.

2.4 Decision Trees in Machine Learning

Decision trees are one way that computers can begin to make thought-through decisions. If you decide to not have your morning coffee out on the patio due to it being cold outside, that feels like an easy and straight-forward decision. But if you were to build a robot that either drinks its coffee indoors or outdoors following your own criteria, specifying those gets harder. How cold is too cold? Can it be too hot? Does rain affect the decision? Does the same apply for tea? The number of potential if/else statements can be overwhelming, not to mention tedious to encode. If you instead record the weather and how long you spent outside drinking coffee you might be able to off-load the boring parts to machine learning. Assume for a moment that after gathering data for 14 days you are left with the table below (2.1).

Table 2.1: Data collected regarding conditions and time spent outside

Outlook	Cold?	Rain?	Drink	Day-of-week	Time-spent-outside
Sunny	No	No	Coffee	Mon	15 min
Cloudy	A bit	No	Coffee	Tue	12 min
Cloudy	A bit	No	Coffee	Wed	13 min
V.Cloudy	Yes	Yes	Coffee	Thu	0 min
Sunny	No	No	Coffee	Fri	20 min
Sunny	A bit	No	Coffee	Sat	22 min
Sunny	A bit	Yes	Coffee	Sun	0 min
Cloudy	No	No	Coffee	Mon	16 min
Sunny	No	No	Coffee	Tue	15 min
Cloudy	A bit	No	Coffee	Wed	11 min
Sunny	No	Yes	Tea	Thu	0 min
Sunny	No	No	Coffee	Fri	26 min
Sunny	No	No	Coffee	Sat	45 min
Sunny	No	No	Coffee	Sun	35 min

Given the data above we can construct a decision tree (Figure 2.6) which based on this data can guide the coffee robot to take appropriate length coffee breaks in the morning.

The decision tree (2.6) shows that the rain data is indeed one of the best predictor of time spent outside. Due to the limited data however both type of drink and how cloudy it is are equally important, which may or may not also be a valid link. The

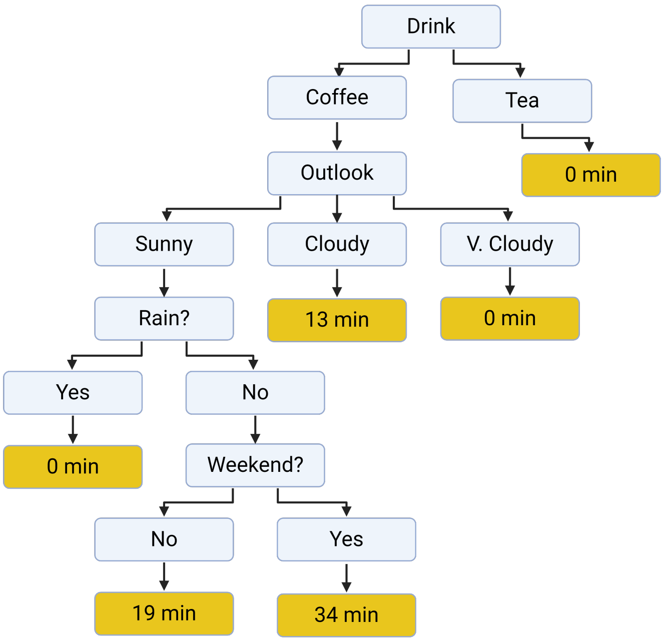

Figure 2.6: Example decision tree created from data in table 2.1. Terminating nodes (yellow), also called leaves in the context of decision trees, show the average time for data-points that fall into that leaf node [12].

network can be further expanded to fit more and more of the data as questions to answer, but there may be some variation here that is not explained by the measured data, and we run the risk of overfitting if every possible label is made into a branching decision.

2.4.1 Discrete Classification

In the case of using decision trees to classify data, we need to figure out how to construct the tree to produce the best division. To decide which branches on the tree to split we can utilize something called Gini Impurity, which is a measurement of the likelihood of a incorrect classification of any additional data. Gini is a type of loss function, which is a measure of the "cost" associated with an action or decision [23]. One of the limitations of Gini is that it only works on categorical targets but similar methods are used when dealing with continuous targets. So for the following example, we want to look at if we went outside or not, i.e is time spent outside more than 0 minutes? The formula for Gini Impurity (GI) is show in Equation 2.8. In the equation we have k targets (k=2 in our case; Outside/Inside).

$$GI = 1 - \sum_{i=1}^{k} p_i^2 \tag{2.8}$$

In the example above (Figure 2.6) we split the tree based on a few categories (Drink, Weekend, Rain etc.) and end up with an eventual target, outside Yes/No (0 < Time-Spent-Outside; True/False). To calculate the GI for a split, we calculate the GI for each sub-node created by the split and take a weighted score based on node population. If we look at the split for "Outlook" we have three labels in the category; Sunny, Cloudy & V.Cloudy. Within each sub-node, we have a varying population of data. If we determine that going outside is the success in our probabilities we have a p(Outside) = 7/9 in the Sunny sub-node, p(Outside) = 4/4 in the Cloudy sub-node and p(Outside) = 0/1 chance in the V.Cloudy sub-node. We also get the probability of not going outside (Failure), which is 1-p(Success). For each sub-node we get a GI of one minus the square of p(Outside) plus the square of p(Inside).

$$Sunny\_GI = 1 - [(7/9)^2 + (2/9)^2] = 0.346 \tag{2.9}$$

$$Cloudy\_GI = 1 - [(4/4)^2 + (0/4)^2] = 0 \tag{2.10}$$

$$V.Cloudy\_GI = 1 - [(0/1)^2 + (1/1)^2] = 0 \tag{2.11}$$

We now weight the sub-nodes based on population. Given that all the data have a entry in the Outlook category we have a total population of 14. The weighted GI for the split based on Outlook is therefore calculated as:

$$GI_{Outlook-Split} = \frac{9}{14} * 0.346 + \frac{4}{14} * 0 + \frac{1}{14} * 0 = 0.222 \tag{2.12}$$

Doing the same calculations on the type of drink yields a GI of 0.241. The higher value for the Drink split indicates that the split based on Outlook yielded a more homogeneous split, and should be ranked higher than splitting on Outlook. Based on the data however, there is a split that results in perfectly homogeneous sub-nodes, splitting on the Rain category, which yields a GI of 0. Using this methodology, trees can be constructed.

2.4.2 Decision Trees with Continuous Targets

The original data in Table 2.1 has a continuous target variable which means that the time spent outside is a numerical value and not just a Yes/No. When dealing with continuous data, splitting the nodes based on continuous values using the previously discussed method is unfeasible. For this reason there is also the need for decision tree methods with continuous targets, also called regression trees [24]. One method to decide where to split nodes with continuous targets is using Variance Reduction. The basic formula for calculating variance is shown below in Equation 2.13.

$$Variance = \frac{\sum_{i=1}^{n}[(X_i - \mu)^2]}{n} \tag{2.13}$$

In the equation above (2.13) X_i is sample i, and μ is the mean within the node. Redoing the example above using the Outlook node, but now with variance and a continuous target the result is 196.78 for the split. Looking at the Drink node the split variance is 144.67, which again points to Drink being a better candidate for an early split than Outlook. Much like before the Rain node would be an ideal starting node (Root) as it has a variance of 0 for both child-nodes. The order of nodes and branches (decisions) is selected such that the variance is minimized.
Based on both the discrete and continuous decision tree methods it can be concluded that the tree presented in Figure 2.6 is poorly optimized.

You can also compare two trees by using similar methods. One commonly used is Mean Square of Error, which uses the same equation as for the variance function presented in Equation 2.13, but without the denominator, and additionally summed over all leaf nodes. Something not brought up in examples is that the loss functions can also include terms that disincentivizes branches, and instead focusing on efficient partitioning, which can help prevent overfitting.

2.4.3 Random Forest

The main decision tree method used in this project is called Random Forest (RF). The decision trees constructed by RF do have some differences to standard algorithms. RF is an ensemble method, meaning that it incorporates several different types of learning algorithms [25]. In this case the base of RF is that several different trees are created and later merged to create a sort of average. Each tree is created using a subset of the provided data in a process called bootstrapping [24]. Bootstrapping works by randomly sampling the data, and allowing for the sampler to select the same data more than once. This introduces a degree of natural variance to the data that other samplers may miss as each tree might only contain a few of the features. Averaging several trees also make the result more resistant to overfitting which is more unlikely after several randomized samplings[24]. Since the creation of each tree in the forest is independent of all other trees/forests the process is easily parallelized, which is an important feature for time efficiency.

There is also a variant of this approach called Extra Trees, which is short for Extremely Randomized Trees [26]. While Random Forest bootstraps subsets of the data, Extra Trees uses all data for the creation of every tree. Another difference is that while RF chooses the cut-points based on what generates the lowest loss according to some loss function (like Gini-Impurity), Extra Trees chooses randomly. This added degree of randomness means that Extra Tress produces more diversified trees which are more robust to noise and bias.

2.4.3.1 GENIE3

Before leaving the topic of Random Forest there should probably be a quick introduction to GENIE3, GEne Network Inference with Ensemble of trees, which is an algorithm used for inferring gene regulatory networks from expression data [27]. The algorithm aims to predict links between transcription factors (TFs) and target genes (TGs) based on expression data. It does this by assuming that all other genes are potential TFs for each target gene (g) and defining a regression problem where the expression of g is the response of all other genes, which are used as predictor variables. Several RF models are then trained to predict the response for g based on the predictor variables (other genes) and the predictors are scored based on how much of the model variance they explained. Since this approach treats every gene as a potential regulator of every other gene the resulting output will contain every possible connection, and a ranking followed by defining a cut-off is needed.

2.4.4 Naïve Bayesian Classifiers

While on the topic of classifiers, we'll have one last look on use cases of Bayesian statistics, Naïve Bayes methods. As previously mentioned, a Bayesian approach involves assigning a prior probability and and calculate likelihoods for each event. Bayesian classifiers are no different, and the priors are based on frequencies of the different labels in the training data. The naïvety of the Naïve Bayes Classifier is that each feature of the data is assumed to be independent from all other features [28]. A feature in this context could be something akin to a gene you have measured the expression of using transcriptomics, while the class/label could be the condition the data was collected under. This assumption simplifies computations while in most cases retaining predictive ability. After calculating a posterior based on previously trained parameters the classifier returns the class with the highest posterior probability of describing the data being classified.

2.5 Evaluation

When evaluating machine learning models there are several strategies and measures that can be employed. Here we go through the important ones that are used in this project.

2.5.1 R-squared

The R^2 score, also known as the coefficient of determination, is a commonly used metric when analysing regressions which speaks to the goodness of fit in a model. R^2 is a measure between 1 and minus infinity of the proportion of the variance of the predicted data that is predicted based on the the original data. A score of 0 means that none of the variance of the prediction is explained by the input data while a score of 1 means that the prediction can be perfectly explained by the input data. While usually less common, negative R^2 values indicate that the model is performing worse than a straight horizontal line representing the mean would.
The equation for calculating the R^2 score is quite simple, and includes the sum of squares for all variance of all observed data (SS_{tot}) and the sum of squares of the variance not explained by the model, the residuals, (SS_{res}). This is shown below in equation 2.14.

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} \tag{2.14}$$

While the R^2 score is a fantastic tool it does have its limitations. One of these is the disregard for biases and validity of model. A model that proposes a straight line through only two data points would get a perfect R^2 score even though the model is likely not that useful for other data. The same is true for over-fitted models, which might perfectly explain the variation, while have poor performance for any data it was not fitted to.

2.5.2 ROC/AUC

ROC stands for Receiver Operating Characteristic is usualy shown as a curve and is a graphical representation of the True Positive Rate (TPR) and False Positive Rate (FPR) at various thresholds. The TPR is defined as seen in Equation 2.15 and FPR is defined in Equation 2.16. A False Positive (FP) indicates that the model predicted something to be true while it was not and False Negative (FN) means that something was predicted to be false while being true. True Positive (TP) and True Negative (TN) are simply correct guesses of either true/false. In an ROC the model is tested for a single label at a time, meaning that the act of labeling can be described as a binary true/false classification. An example of a ROC curve is also shown below in Figure 2.7.

$$TPR = \frac{TP}{TP + FN} \tag{2.15}$$

$$FPR = \frac{FP}{FP + TN} \tag{2.16}$$

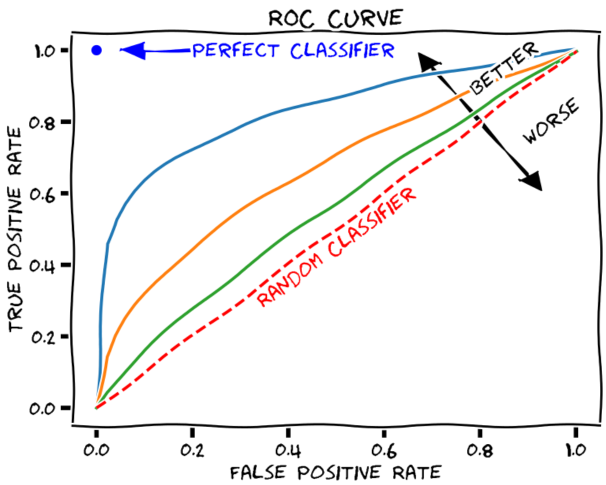

Figure 2.7: Example of ROC curves. Created by MartinThoma, distributed under a CC0 license.

AUC is an abreviation for Area Under ROC Curve and is as the name applies the full area under the curve and ranges between 0 and 1. An AUC score of 1 indicates that the true positive rate is 100% and everything is perfectly predicted. A score of 0.5 however means that the classifier is performing on the same level as randomly assigning labels. While an AUC of less than 0.5 is possible, it likely is a result of improper testing. One major limitation with ROC/AUC is that for heavily imbalanced data-sets, such as a set of 100 points where 99 of them have the same target label, they could result in high scores for faulty predictors, like one who always guesses the same label and only gets one guess wrong.

2.5.3 Cross-validation

When validating machine learning models there is a possibility that the results are due to a favorable split in training vs test data. To deal with this Cross-Validation (CV) is often employed. CV is also typically used when working with limited data and involves splitting the data into k subsets (folds). The model is then trained on k-1 folds and tested/validated on the remaining fold. This is repeated k times until all data has been validated on. There are also various sub-types of CV, like stratified CV where each fold is given approximately the same makeup in regards to target-class/label. Another common example is Leave-One-Out (LOO) CV which is special in that the k is set to the same number as there are samples (n), meaning that each sample is used to evaluate the model once and is part of the training data n-1 times. An example of this is shown in Figure 2.8

Figure 2.8: Example of a five-fold cross validation. In each re-run a new fold is chosen as the test data and remaining folds are used to train the model [12].

3

Methodology

3.1 GRN construction

A Gene Regulatory Network (GRN), as stated before, is a web of interacting DNA, RNA and proteins that form a signaling web that governs much of what happens in a cell. In this work a lot of attention has been on Transcription Factors (TFs) and their ability to influence expression of one or more genes. Simple models of a GRN are easy to construct and could be as simple as "if A is high, B will be repressed" or as complex as thousands of regulators all influencing each other to certain degrees. The complexity here lies not only in discerning what TFs are actually affecting the expression of some gene but also to what degree.

3.1.1 Network Construction

There are several alternatives for trying to elucidate the network structure found in yeast cells from expression data. The key with data selection here is to include samples with varying expression profiles as it is the changes in expression that guide the process. ring the earlier parts of the project the main method studied was a binned co-regulation search followed by a real valued regression for selection of performant links.

Since the initial implementations used suffered from a few years without continued maintenance the goal was to find more recent software. Eventually the GReNaDIne python package (version 0.1.9) was found which implements several algorithms for inferring GRNs from high-throughput data [29]. The algorithm which was chosen was GENIE3 [27] which is a tree based (RandomForest/Extra-Trees) ensemble method used to score the potential connections between network nodes. GENIE3 was the first place winner in the DREAM4 *In Silico* multifactorial challenge, which was a crowd-source competition to identify biological networks. While the GReNaDIne paper states that the best choice of algorithm depends on several factors the GENIE3 algorithm was used as an initial basis for all models constructed afterwards for the sake of consistency.

3.1.2 Choice of network model

After GENIE3/GReNaDIne produced a list of scored connections the top scoring connections were used to build a network. Here the genes were nodes and the edges represents interactions between the genes/gene-products/nearby regulatory sequences. Below is a representation using 56 nodes and connecting edges presented as a directed graph.

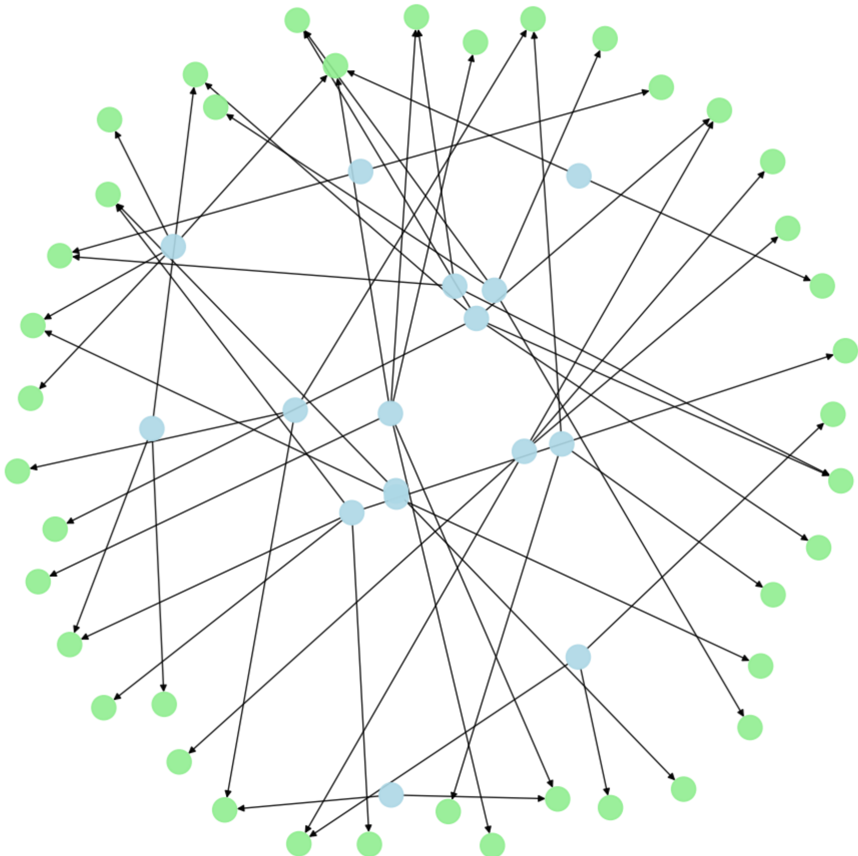

Figure 3.1: Small example network using 56 weakly connected nodes with labels hidden. Blue nodes are genes coding for documented transcription factors and green nodes are metabolic genes present in the GEM.

The figure shown above (Figure 3.1) is an example of one weakly connected network which forms out of the 500 highest scoring connections. Weakly connected means that while all nodes are connected by at least one edge (arrow), all nodes are not reachable from any other node if travel is only allowed in the direction of the linking arrows. The opposite would be a strongly connected network where each node is reachable from any other. Decreasing the score threshold for connections results in larger networks as more weakly connected networks are linked. The network shown in Figure 3.1 was created using a dataset of 700 responsive deletion-mutants where the deletion significantly alters the expression of at least two other genes [30].
The software used to train the network was the lgnpy python package (version 1.0.0)

[31]. In this case training the network refers to updating parameter distributions as opposed to assigning edge weights since the network structure used is Gaussian. The Gaussian approach offers some advantages over discrete networks such as the uncertainty included in a Gaussian distribution which can be biologically relevant. The intrinsic smoothing in fitting a distribution also helps with noise tolerance. Initial prototype networks created used a discrete network but the binning, which categorizes gene expression into arbitrary bins, proved too blunt to handle the nuance of some expression patterns.

3.1.3 Data selection

Over the course of the project, as the scope and goal has changed, different datasets have been considered. The initial dataset used was composed of expression data under varying environmental conditions [32]. The lack of wildtype controls and some missing condition overlaps were some of the drawbacks of the dataset. When the focus later shifted towards more focus on the role of TFs and working with continuous models more datasets were tested. The dataset that later saw most use contains expression data of deletion mutants at standard conditions [30]. The reasoning behind this is that it allows for a more nuanced picture of the connections between TF expression and TG expression due to the larger sample size. Given the correct data, this new approach could still allow for environmental condition modeling by incorporating data of how environmental factors affect TFs and manipulate the TF levels to mimic the conditions.

3.2 GEM implementation

The genome scale metabolic model used in this project was Yeast9 [4] which contains 4130 reactions, 2805 metabolites and 1162 genes. The model also contains information about the localization of reactions and metabolites across 14 different cellular compartments, such as cytoplasm, mitochondrion or the cell nucleus. The Yeast9 model is distributed along with a set of Matlab scripts which along with either the Raven or Cobra Matlab toolbox allow you to interact with the model. Since this project is primarily Python based the included tools were not utilized, but compareable alternatives were developed in Python scripts utilizing CobraPy (version 0.29.0) [33], the Python variant of the previously mentioned Cobra toolbox for metabolic models. For the medium a document of relative fluxes for different growth media was adapted from Snitkin et al. [34]. Yeast Extract–Peptone–Dextrose (YPD) medium was chosen as the model medium since it allows for close to maximum growth rate [35].

The project has a focus on modeling chemostat reactors where the cells are kept in an environment with one or more limiting nutrients, most often the carbon-source. To mimic this in a metabolic simulation a slightly more complex objective function can be used. Most often the objective function of a genome scale model is set to maximize the growth rate within the import/export bounds set in the model [13]. To capture the characteristics of a population of cells growing in a chemostat reactor two things need to be included in the model constraints. Firstly the growth rate needs to be set to the dilution rate of the reactor as this is the only viable growth rate of the average cell in the reactor, as a higher or lower value would lead to unrealistic overcrowding or eventual wash-out respectively. Secondly the model needs to minimize the flux through the carbon-source uptake exchange reaction given the growth rate, as to match the average uptake in the largest possible population given the carbon-source available.

3.3 GRN integration into GEM

While a "lifelike" network of gene interactions is interest in itself, the aim of this project is to improve metabolic predictions based on this data. To facilitate this, there needs to be a framework to translate the output states of a GRN into constraints for the GEM. This problem can be split in two parts, converting gene expression into a biologically meaningful value and using these values to constrain reactions.

3.3.1 Linear Gaussian approach

The project eventually integrated linear Gaussian approaches to modeling Bayesian networks as this allows for inferring states not present in the training date due to the probability distributions including all possible states however unlikely. The approach used for the linear gaussian bayesian network was adapted from a paper where the known specific growth rate of *Saccharomyces cerevisiae* is used to find the correction factor C that minimizes the error in growth rate [36]. The paper multiplied the count data for each gene with the factor C and used the result for setting bounds in the GEM. The very same approach was used in this case to determine the value that best allows for conversion of expression values to flux bounds. The main differences in the application of this method is the network type and data used to train the network.

3.3.2 Gene-Reaction Rule logic -> Numerical value

The reactions in the genome scale model have corresponding Gene-Protein-Reaction rules (GPRs) which detail which genes (and by extension, which enzymes) are needed for the reaction to take place. To be able to use the gene expression data to control the reactions, the GPRs need to be translated from the typical "A and B or C and D or A and D" into numerical values based on the gene expression. The general approach taken in this project closely follows what the CoRegFlux [37] R package uses, but with some added improvements such as giving parentheses special attention, as opposed to the original methodology which disregards them. Parentheses in this case would indicate genes whose enzymes form complexes or otherwise need to all be present. An example would be: "(A and B) or (A and C)" where either A+B or A+C would work, but B+C would not. The new implementation recursively handles parentheses to maintain the intended order of operation, while the first line of code in CoRegFlux replaces parentheses with empty spaces.

The basic algorithm used is:

1. Convert gene names into a numerical representation of their expression levels.
2. Split GPRs into subgroups based on parentheses (if applicable) and handle recursively.
3. For genes separated by "and," the minimum value of the elements to the left and right is used:
$$\text{"A and B"} \Rightarrow \min(A, B)$$
4. For elements separated by "or," the sum of the separated values is used:
$$\text{"A or B"} \Rightarrow A + B$$

These translations allow the incorporation of gene expression data into the metabolic model.

4

Results

4.1 Pipeline

The general pipeline designed in this project is shown below in figure 4.1. The stages are: Data selection, network structure generation, network parameter estimation, inference of gene states, and lastly utilizing network output to constrain a metabolic model. The pipeline is designed to run wholly in python, which was one of the unstated goals with the project, to possibly allow for easier integration with in-house workflows.

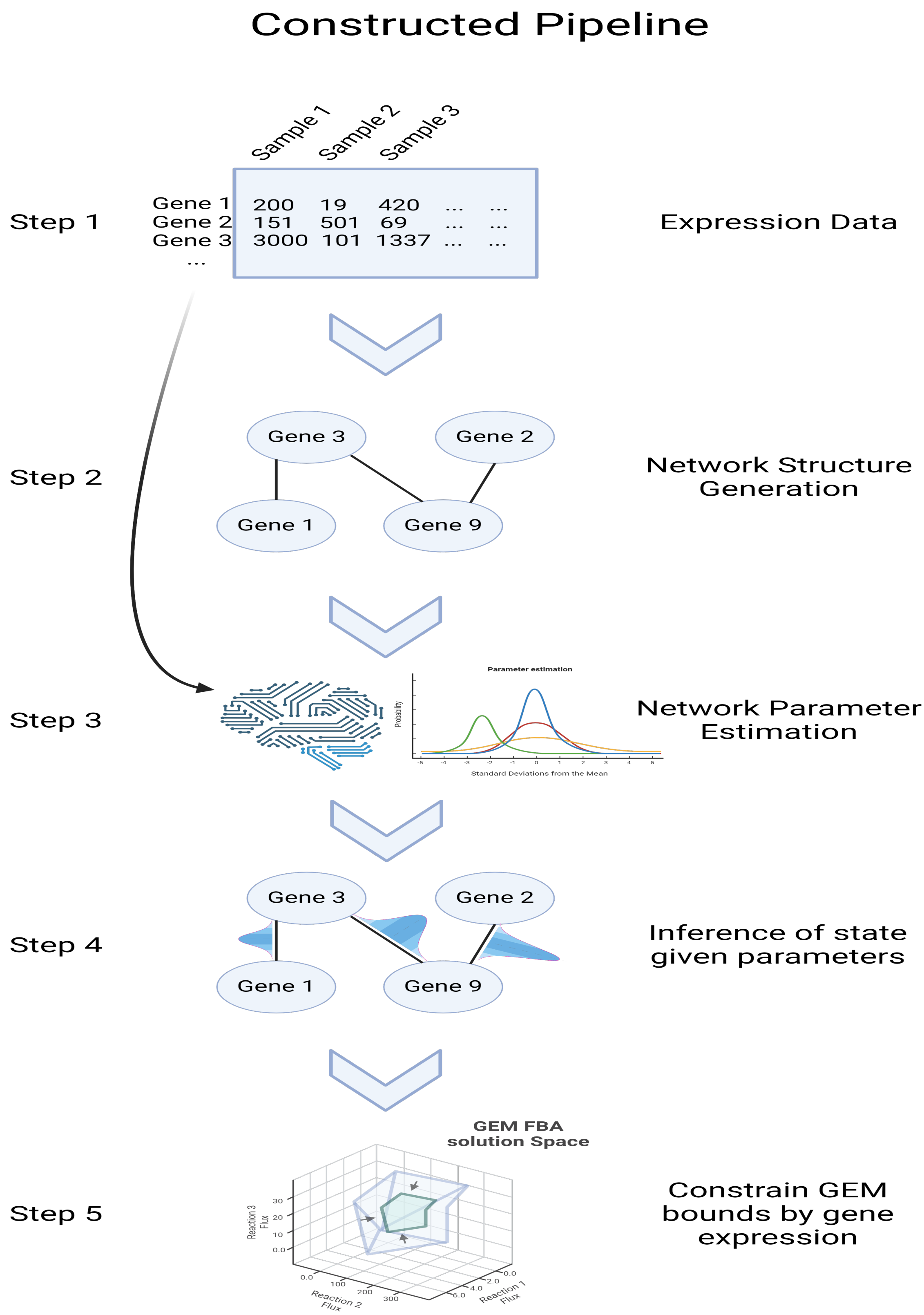

Figure 4.1: Illustration of developed pipeline for constraining GEM using GRN [12].

4.1.1 Stage 1: Data

The first stage is highly dependent on the phenomena being studied. In this project the interplay between transcription factors and target genes guided the decision to work with expression data with deletion mutants as samples. The full dataset has over 1400 deletion mutants, but in this project only the subset of deletants that cause significant change in at least two other genes are used, resulting in $\sim$ 700 samples.

4.1.2 Stage 2: Network Structure

The second stage uses the decision tree learning approach described in the theory section 2.4 to create a number of networks from which the best performers are chosen. The result of the GENIE3 based network construction is a score matrix of size (n x m) where n is the number of TGs and m is the number of TFs of interest. The values populated in the matrix are the relative score of the co-expression between TG and TF. The scores are then ranked and the top performers are used to build the network. Since the matrix includes all TFs and TGs a cutoff is needed, since in reality all TGs are not controlled by all TFs. Given that the scores are relative to the number of TFs/TGs the cutoff is a number of links which are to be included.

4.1.3 Stage 3: Estimation

The network parameter estimation results in fitting Gaussian distributions, also commonly referred to as normal distributions, to the expression of each node in the network. The distributions also have CPDs associated with them which dictates the relationship between the expected distribution of a node given the state of the parent nodes.

4.1.4 Stage 4: Inference

The inference produces updated means and variance-values for all affected genes based on the evidence provided. In this context evidence refers to set values that a node should have and the model updates the remaining nodes to accommodate. If no evidence is provided the fitted mean and variance is returned.

4.1.5 Stage 5: Integration

The method chosen to integrate the GRN with the GEM in this project was to constrain the reaction bounds of the GEM. The bounds dictate the largest and smallest allowed flux through each reaction. Since the GRN output is a measure of gene expression, a conversion factor is needed to convert the value into flux bounds. Since there is one reaction in the GEM for which a set value for the flux is known under optimal condition, the biomass pseudo-reaction, the conversion factor was chosen to minimize the error between this flux and the empirical evidence. Using exchange fluxes that mimic an optimal media (YPD) the C factor was fitted such that the flux through the reaction best described the highest biochemical growth-rate of *Saccharomyces cerevisiae*, which with a generation time of 90 minutes on glucose gives it a specific growth-rate of approximately 0.46 h^{-1} [38]. The errors for varying values of C are shown in figure 4.2.

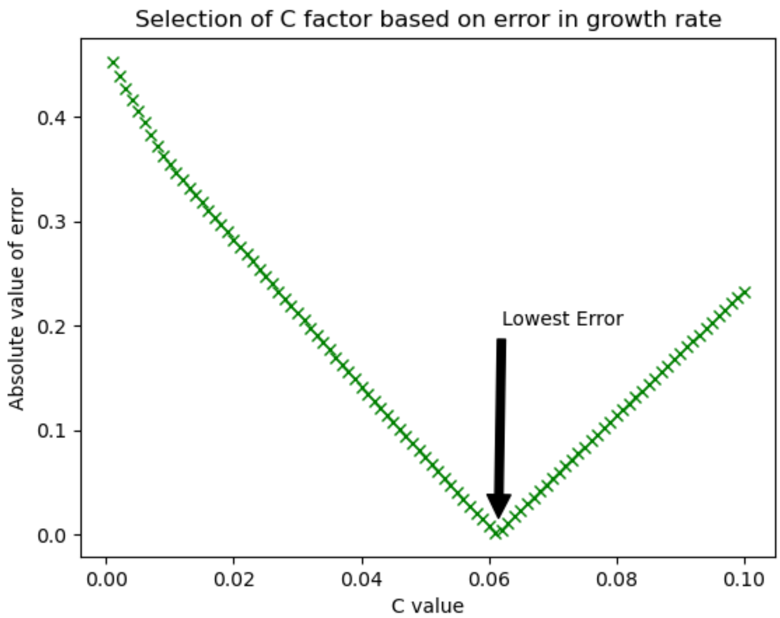

Figure 4.2: Plot of absolute error in growth rate for varying values of conversion factor C.

The C factor was optimized by running a greedy search over a range of possible values and selecting the one which yields the lowest error. Given the step size in the figure above (4.2) the optimal value for the conversion factor was determined to be 0.061.

4.2 Results of integration on the GEM

After the integration had been verified the task of evaluating the performance begun. There were several key points where the integration of a suitable GRN was theorized to improve the accuracy of the GEM. One of these points was gene essentiality prediction where there was potential for not only correctly predicting current false negatives but also seeing the effects of essential transcription factors. Using the essentiality predictor, provided by the Cobrapy python package, the unmodified Yeast9 GEM was predicted to have 99 essential genes. Comparing the genes predicted to data from YeastMine reveals that 30 of these genes are false positives while the whole Yeast9 model contains 159 genes that are experimentally determined to be essential meaning that there are 90 false negatives. After integration with the GRN the predictor was rerun yielding the same results, thus showing that the integration did not lead to any improvement regarding essentiality. The same was true for when essentiality among the TFs included in the GRN, which are not present in the GEM, was predicted as high degrees of down-regulation of these genes did not yield an inviable phenotype even in the cases of verified essential TFs. Shown below in figure 4.3 are confusion matrices showing the data discussed above.

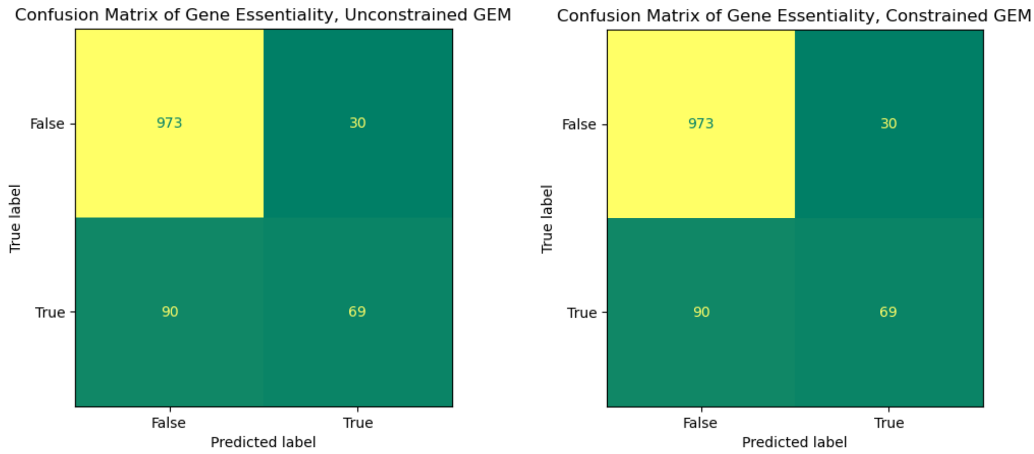

Figure 4.3: Confusion matrices off essentiality prediction for the constrained and unconstrained GEM .

Another metric that is of value is if the constrained network improves phenotype predictions on the model output. Two models were used in the following experiments, the first one being an unmodified Yeast9 GEM, and the second a variant constrained by the GRN. Both models used the same medium, YPD, which in this context is a non-limiting medium. On both models, flux data corresponding to single gene deletions for each relevant gene was simulated using parsimonious flux balance analysis (pFBA). A gene was labeled as relevant if it was found in the GRN and in the Yeast9 GEM, which depending on choice of GRN varies, but leaves 646 genes using the small iteratively improved network described in Table 4.1. For the GRN constrained model each gene knockout was accompanied by inference of the gene expression of all other genes given the knockout, which was used to constrain the model during that pFBA run. The output was a list of the fluxes for each and every reaction in the model (4130) for each relevant deletion (646). The resulting data is then evaluated based on how characteristic the remaining gene expressions are for each deletion, ie. the performance of machine learning based classifiers in predicting the function of each deleted gene. Gene Ontology (GO) process terms were used as target labels for deleted genes. For the choice of classifier a number of alternatives were tested, and in accordance with Zhang, C. et al [4] a Naïve-Bayes based approached proved performant. Shown below in figure 4.4 are the receiver operating characteristics (ROC) for three different GO terms ("generation of precursor metabolites and energy", "lipid metabolic process" & "amino acid metabolic process"). All examples are done using a Guassian Naïve Bayes classifier and a 10-fold stratified, shuffled, cross-validation. Lipid metabolism was the GO term with the most applicable labels in the network evaluated, amino acid metabolism is the third most common label and the generation of precursor metabolites and energy label

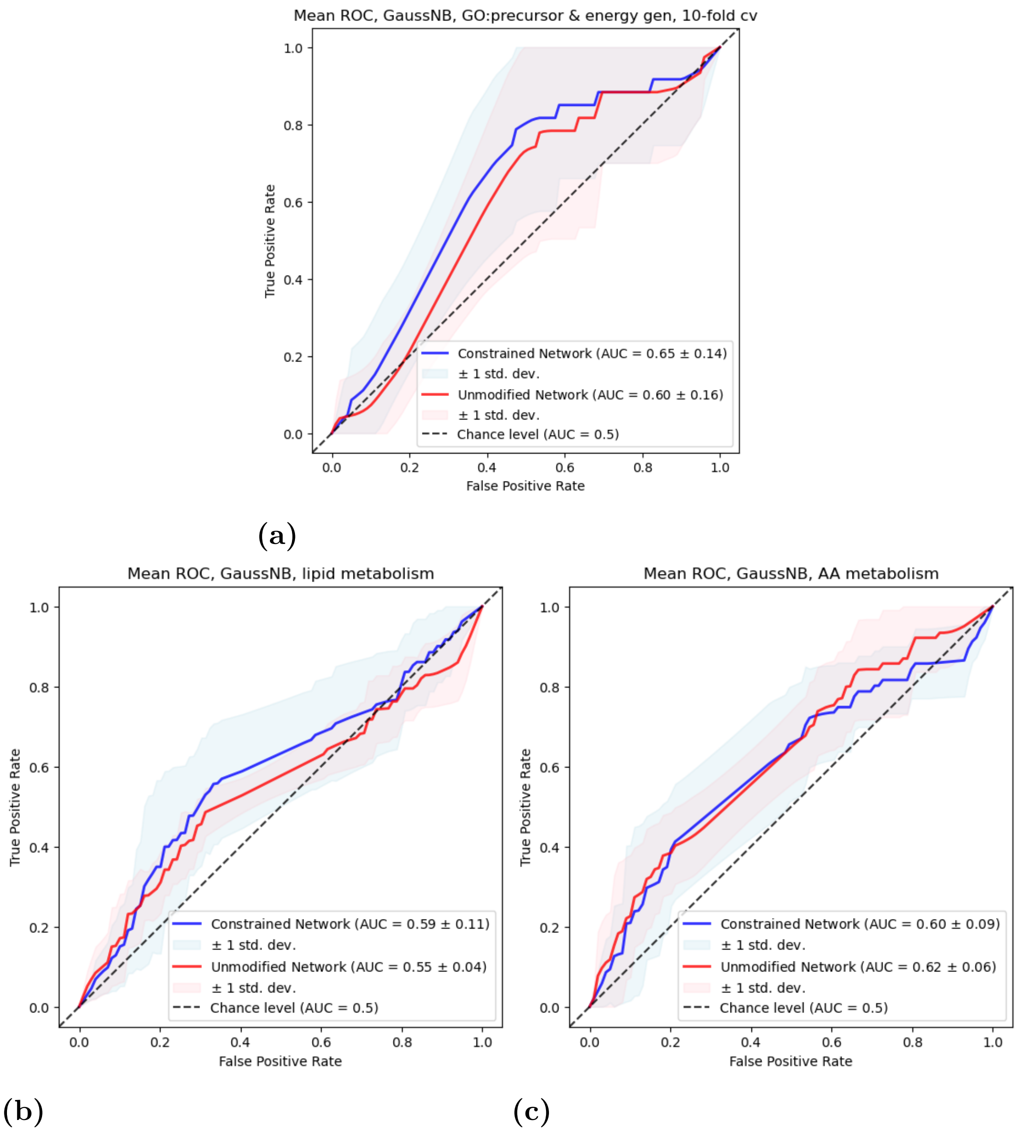

Figure 4.4: ROC curves for three different classification labels.

had both networks performing well. As shown in Figure 4.4 the model performed marginally better in some categories, while on par or worse in others.

4.2.1 Network performance and improvements

To compare the network resulting from the Random Forest (RF) approach two other networks were created. The two networks were based on data from the YeastMine database for the S288C strain [39]. One of the networks uses only Manually Curated (MC) TF-TG pairs while the other uses all available High-Throughput (HT) connections. Both TF-TG datasets where pruned to remove self-loops, TFs in the TG list (to prevent larger loops) and pairs containing TFs/TGs for which expression data is lacking. The eventual size of the MC network was 479 edges and the HT network was 24629 edges. All networks were then evaluated using LOO R^2-scoring using the scikit-learn python package [40]. The LOO cross validation uses all available data except for one sample to train the network and uses the excluded one for testing. This is then repeated until all samples have been excluded one by one. Since the data used has deletion mutants as samples the LOO technique is more suitable than strategies that use larger sets of test data, due to the unique characteristics of many deletions. The LOO method is also more similar in execution to how the network is intended to be used, with single samples being generated with novel conditions/-constraints.

Since a R^2 score gives a simple overview of the quality of a network, or more specifically the variance of the data explained by it, iterative improvements can easily be compared to it by calculating its R^2. As a proof of concept a short algorithm for random changes of the network followed by score comparison was set up and is presented below in pseudo-code, listing 4.1.

Listing 4.1: Pseudo-code example of algorithm used for iterative improvements.

```
1 Best_Net = Original_Network
2 Highest_Score = Score(Original_Network)
3 RandomNumberGenerator.seed = XXX
4 Nodepairs = Node_Picker(Best_Net,num_pairs)
5 New_Nets = Rand_Mod(Best_Net,Nodepairs[i in 0->num_pairs])
6 If Score(New_Nets[k in 0->n_pairs]) > Highest_Score:
        Highest_Score = Score(New_Nets[k])
        Best_Net = New_Nets[k]
7 Go to 4 (DO X10)
8 Save(Best_Net) as Network_XXX.net
```

In the code segment above the following is taking place. Line 1 sets the currently best performing network as the original input network. The next line (2) scores the starting network and saves for future comparison. The third line (3) sets a fixed seed for all future Random Number Generator (RNG) calls which ensures reproducibility. Line 4 selects some specified number of random nodes from the network using the seed from 3. Step number five (5) creates new networks by modifying the currently best performing one. The modifications allowed are: Reverse edge, delete edge and add new edge with some probability assigned to each action with outcomes dependent on calls to the RNG. The sixth (6) step is to score the new network(s) and see if it outperforms the current leader. The scoring methods previously used are mean square of error and, as mentioned, R^2-score. If a network outperforms the current best a new best is set. This process is repeated a number of times, with 10 being the example given in step seven (7). The last step (8) is to save the top performer with a name indicating the seed used to a file which could in theory be the input for new rounds of unsupervised modifications.

The results of the previously mentioned models are shown below in table 4.1. Worth noting is the relatively narrow span between best and worst performance, and medians that are above the average, indicating that the minimum values greatly affect the score.

Table 4.1: R^2 scores for the networks

Network	Avg R^2	Min R^2	Max R^2	Median R^2
Manually Curated	0.883	0.563	0.99	0.911
RF + iter. improve	0.868	0.326	0.99	0.900
Random Forest	0.864	0.305	0.99	0.897
High-Throughput	0.859	-1.68*	0.99	0.906

*The negative score indicates a performance worse than simply predicting the mean value.

As seen above in table 4.1 the random forest network performed slightly better than the HT network and slightly worse than the MC network, all while only having access to the expression dataset. The iteratively improved model performs 0.4 percentage points better than the base model after just 100 iterations of the algorithm in code listing 4.1. The iterative improvement algorithm has the potential to be further improved with modified modification probabilities and possibly refined logic but further work on this lies outside the scope of this project.

5
Discussion

5.1 Limitations in essentiality predictions

One of the major limitations present in the essentiality prediction lies in the design of the GEM. As the name implies the genome scale metabolic model is primarily focused on cell metabolism and does not include many of the processes that take place in the cell, many of them vital. While the model does include some processes that are not strictly metabolic, such as cell growth, the lack of GPRs tied to the biomass pseudo-reaction is likely a major source of false negative essentiality predictions. Manual sampling of the false negatives reveal that many of them are vital in processes not covered by the Yeast9 model such as tRNA export (YGR090W) or actin cytoskeleton-regulation (YIR006C). While possible but not verified it is also likely that all tested TFs are also only essential in their regulation of genes outside of the scope of the GEM and the metabolism of *Saccharomyces cerevisiae* is resilient to single TF deletions.

The choice of data used to train the GRN is also a potential source of error in this matter. While the model proved highly accurate in predicting the expression phenotypes that arise in deletion mutants the fluorescence readings, unlike true RNA counts, has baseline which is relatively high. The lack of a zero reading for deleted genes is not an issue when predicting expression but may turn into one during the translation into reaction flux bounds. The conversion factor C was chosen as it allowed for the smallest error in growth rate as compared to real life data but may not constrain the model enough for traditional essentiality prediction.

5.2 GRN scoring

When choosing a scoring strategy the aim is always to minimize introducing outside bias into the testing. The choice of scoring method does not, however, negate the inherent bias introduced through repeated optimization. Here the models are not necessarily only being improved in their similarity to experimental data, but also introduces potential for overfitting which may lead to worse results for prediction of novel conditions or inclusion of additional data. However, the risk of overfitting, while a real concern, should be comparable to the bias introduced in the initial network construction, and given large enough datasets remain a less than critical issue.

The high R^2-scores are both cause for celebration and a potential warning. There are several potential reasons for over-inflated scores, including data leakage, overfitting or high collinearity between the sample features. Overfitting appears to be the likelier culprit as the model complexity is rather high compared to the number of samples with the largest networks created (not shown) having a higher degree of freedom than the number of samples they were trained on. For the case seen in Table 4.1 raw data was used, so data leakage due to normalization is unlikely, and no contamination between test and training data was found after examination. However it is worth noting that the high collinearity of some genes (highly similar expression profiles) could in theory lead to similar levels of overestimation of predictive performance as data leakage.

5.3 Variability in ROC results

The ROC-AUC results, shown in Section 4.2, performed differently for the various GO-term labels. Several factors contribute to this behavior. One significant reason is the relative imbalance in the labeling task; some GO-terms have very few associated genes, providing limited training data for those labels and leading to poorer results. Additionally, the inherent complexity of certain processes may involve a higher degree of variability, making them less predictable. This variability, coupled with constraints on the model that may be unfavorable to some terms, can negatively impact performance. In several instances, neither the constrained nor unconstrained fluxes enabled the classifier to predict the labels better than pure chance. This outcome suggests that those GO-terms might be more significantly affected by the factors mentioned above.

5.4 Why integrate the two models?

As we saw in the results section, constraining the GEM resulted in a moderately increased accuracy in labeling by the classifier for some GO terms. This suggests that the constrained model provides more accurate and biologically relevant predictions about gene functions. Improved accuracy in functional annotation enhances the reliability of predictions, crucial for understanding and manipulating biological systems. The constrained model likely captures essential biological interactions and regulatory constraints absent in the unconstrained model, leading to a more precise representation of cellular processes. In synthetic biology and metabolic engineering, better prediction accuracy facilitates more effective genetic interventions and optimizes the design and implementation of experiments. The improved classifier performance also validates the model constraints, highlighting their importance and guiding further model refinement.

Additionally, integrating a GRN with the GEM enables the simulation of genetic manipulations such as overexpression and double knockouts, which are challenging to represent with implicit models. The explicit representation provided by GRNs allows for detailed modeling of these conditions, giving insights into complex regulatory mechanisms. For example, simulating overexpression can reveal the effects of increased transcription factor activity on target genes, while double knockouts can help identify synthetic lethality or pathway robustness. These capabilities are crucial for designing targeted genetic interventions and understanding the regulatory dynamics within metabolic networks. Therefore, the integration of GRNs with GEMs not only improves prediction accuracy but also broadens the scope of possible simulations, offering a more comprehensive toolkit for metabolic engineering and synthetic biology.

5.5 Use of time-invariable GRNs

One valid concern regarding the methodology is the models disregard for temporal changes and basing the predictions solely on snapshots of TF and TG expression with no information about the trends that led to them. A way to include this could be to work with dynamical networks, such as a dynamic Bayesian network, for the construction of the GRN. A time slice network would in practice work as several networks that communicate back and forth and thus simulating changes over time. The major drawbacks with this approach are increased model complexity, a greatly increased training time and the need for a large quantity of high quality time dependent data. The project was thus forced to work with simpler models to finish within the given time frame. However, while the use of dynamic models could provide deeper insights into temporal changes, static models can effectively capture the essential regulatory relationships and interaction strengths. These models can predict gene expression levels based on the presence and activity of relevant transcription factors. Despite the lack of temporal resolution, static networks remain valuable tools for understanding the fundamental structure and function of gene regulatory networks and should not be overlooked.

5.6 Future Improvements

Future improvements for the project could include upgrading the Gene Regulatory Network to a dynamic Bayesian network, which would enhance the model's capability to capture temporal dependencies in gene expression data at the expense of the factors discussed before. Additionally, incorporating data normalization and shuffling techniques would allow the integration of several sources of expression data, increasing the robustness and accuracy of the model, while minimizing the bias introduced by discrepancies between separate data sources. Evaluating how different network sizes affect the outcomes is also crucial, as it can provide insights into the optimal complexity for achieving the best predictive performance.

6
Conclusion

The result of this project has been the construction of a pipeline that uses expression data to construct a statistical network for constraining a genome-scale metabolic model (GEM). Applying these network constraints to the GEM altered the overall flux distribution, which, in certain contexts, improved the accuracy of supervised labeling. This approach could be particularly useful for working with more sparsely annotated organisms and for rapidly evaluating flux distributions following modifications to expression levels.

There are numerous potential improvements to the current methodology that could be explored in future projects. These include protocols for data merging and normalization to integrate multiple data sources, as well as the use of dynamic Bayesian networks to better capture the time-dependent nature of transcription factor-target gene (TF-TG) interactions. Additionally, further exploration of the impact of network size and complexity on model performance could provide insights into optimizing model configurations for various applications.

Bibliography

[1] Shou-Fu Duan et al. "The origin and adaptive evolution of domesticated populations of yeast from Far East Asia". In: *Nature Communications* 9.1 (July 2018), p. 2690.

[2] G.B. Gu C.and Kim and W.J. et al. Kim. "Current status and applications of genome-scale metabolic models." In: *Genome Biol* (2019). URL: `https://doi.org/10.1186/s13059-019-1730-3`.

[3] R. Barbuti, R. Gori, and P. et al. Milazzo. "A survey of gene regulatory networks modelling methods: from differential equations, to Boolean and qualitative bioinspired models." In: *J Membr Comput* (2020). URL: `https://doi.org/10.1007/s41965-020-00046-y`.

[4] Chengyu Zhang et al. "Yeast9: A Consensus Yeast Metabolic Model Enables Quantitative Analysis of Cellular Metabolism By Incorporating Big Data". In: *bioRxiv* (2023). DOI: `10.1101/2023.12.03.569754`. eprint: `https://www.biorxiv.org/content/early/2023/12/05/2023.12.03.569754.full.pdf`. URL: `https://www.biorxiv.org/content/early/2023/12/05/2023.12.03.569754`.

[5] D. Seçilmiş, T. Hillerton, and D. et al. Morgan. "Uncovering cancer gene regulation by accurate regulatory network inference from uninformative data." In: *npj Syst Biol Appl* (2020). URL: `https://doi.org/10.1038/s41540-020-00154-6`.

[6] Fei Liu et al. "Inference of Gene Regulatory Network Based on Local Bayesian Networks". In: *PLOS Computational Biology* 12.8 (Aug. 2016), pp. 1–17. DOI: `10.1371/journal.pcbi.1005024`. URL: `https://doi.org/10.1371/journal.pcbi.1005024`.

[7] P. Badia-i Mompel, L. Wessels, and S. et al. Müller-Dott. "Gene regulatory network inference in the era of single-cell multi-omics." In: *Nat Rev Genet* (2023). URL: `https://doi.org/10.1038/s41576-023-00618-5`.

[8] Jeremy S. Edwards and Bernhard Ø. Palsson. "Systems Properties of the Haemophilus influenzaeRd Metabolic Genotype*". In: *Journal of Biological Chemistry* 274.25 (1999), pp. 17410–17416. ISSN: 0021-9258. DOI: `https://doi.org/10.1074/jbc.274.25.17410`. URL: `https://www.sciencedirect.com/science/article/pii/S0021925819727197`.

[9] HSUAN-CHAO CHIU and DANIEL SEGRÈ. "COMPARATIVE DETERMINATION OF BIOMASS COMPOSITION IN DIFFERENTIALLY ACTIVE METABOLIC STATES". In: *Genome informatics. International Conference on Genome Informatics* 20 (2008), 171–182. ISSN: 0919-9454. URL: `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3245841/`.

[10] Jeffrey D Orth, Ines Thiele, and Bernhard Ø Palsson. "What is flux balance analysis?" In: *Nature Biotechnology* 28.3 (Mar. 2010), pp. 245–248.

[11] Markus W. Covert et al. "Metabolic modeling of microbial strains in silico". In: *Trends in Biochemical Sciences* 26.3 (2001), pp. 179–186. ISSN: 0968-0004. DOI: https://doi.org/10.1016/S0968-0004(00)01754-0. URL: https://www.sciencedirect.com/science/article/pii/S0968000400017540.

[12] Created with Biorender.com. In: (2024).

[13] Adam M Feist and Bernhard O Palsson. "The biomass objective function". In: *Current Opinion in Microbiology* 13.3 (2010). Ecology and industrial microbiology • Special section: Systems biology, pp. 344–349. ISSN: 1369-5274. DOI: https://doi.org/10.1016/j.mib.2010.03.003. URL: https://www.sciencedirect.com/science/article/pii/S1369527410000512.

[14] Michael J Todd. "The many facets of linear programming". In: *Mathematical Programming* 91.3 (Feb. 2002), pp. 417–436.

[15] Nathan E Lewis et al. "Omic data from evolved E. coli are consistent with computed optimal growth from genome-scale models". en. In: *Mol. Syst. Biol.* 6.1 (July 2010), p. 390.

[16] Purvesh Khatri, Marina Sirota, and Atul J. Butte. "Ten Years of Pathway Analysis: Current Approaches and Outstanding Challenges". In: *PLOS Computational Biology* 8.2 (Feb. 2012), pp. 1–10. DOI: 10.1371/journal.pcbi.1002375. URL: https://doi.org/10.1371/journal.pcbi.1002375.

[17] Nicholas M Luscombe et al. "Genomic analysis of regulatory network dynamics reveals large topological changes". In: *Nature* 431.7006 (Sept. 2004), pp. 308–312.

[18] Ping Xin Ly Ma Melissa Morgan Colleen M. Lebo Michael P. Woods Alok Kumar Sil Samina Alam and James E. Hopper. "The Gal3p-Gal80p-Gal4p Transcription Switch of Yeast: Gal3p Destabilizes the Gal80p-Gal4p Complex in Response to Galactose and ATP". In: *Molecular and Cellular Biology* 19.11 (1999). PMID: 10523671, pp. 7828–7840. DOI: 10.1128/MCB.19.11.7828. eprint: https://doi.org/10.1128/MCB.19.11.7828. URL: https://doi.org/10.1128/MCB.19.11.7828.

[19] M.P.K. Webb and D. Sidebotham. "Bayes' Formula: A powerful but counterintuitive tool for medical decision-making". In: *BJA Education* 20.6 (2020), 208–213. DOI: 10.1016/j.bjae.2020.03.002. URL: https://www.sciencedirect.com/science/article/pii/S205853492030041X.

[20] Marco Taboga. *Bayesian inference.* https://www.statlect.com/fundamentals-of-statistics/Bayesian-inference.. Accessed: 2024-6-3.

[21] H Pishro-Nik. *Chapter 9 Statistical Inference II: Bayesian Inference.* Blue Bell, Pensylvania; Kappa Research, LLC, 2014.

[22] Sarah Herbert et al. *13.5: Bayesian network theory.* 2023. URL: https://eng.libretexts.org/Bookshelves/Industrial_and_Systems_Engineering/Chemical_Process_Dynamics_and_Controls_%28Woolf%29/13%3A_Statistics_and_Probability_Background/13.05%3A_Bayesian_network_theory.

[23] C M Bishop. *1.5.2 Minimizing the expected loss.* New York, NY: Springer, 2006, pp. 41–42.

[24] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. "The Elements of Statistical Learning: Data Mining, Inference, and Prediction". In: 2nd ed. Ensemble Learning. New York, NY: Springer, 2009, pp. 587–604.

[25] L Breiman. "Random Forests". In: *Machine Learning* 45.1 (2001), pp. 5–32.

[26] Pierre Geurts, Damien Ernst, and Louis Wehenkel. "Extremely randomized trees". In: *Machine Learning* 63.1 (Apr. 2006), pp. 3–42.

[27] Vân Anh Huynh-Thu et al. "Inferring Regulatory Networks from Expression Data Using Tree-Based Methods". In: *PLOS ONE* 5.9 (Sept. 2010), pp. 1–10. DOI: 10.1371/journal.pone.0012776. URL: https://doi.org/10.1371/journal.pone.0012776.

[28] Irina Rish. "An Empirical Study of the Naïve Bayes Classifier". In: *IJCAI 2001 Work Empir Methods Artif Intell* 3 (Jan. 2001).

[29] Pauline Schmitt et al. "GReNaDIne: A Data-Driven Python Library to Infer Gene Regulatory Networks from Gene Expression Data". In: *Genes* 14.2 (2023). ISSN: 2073-4425. DOI: 10.3390/genes14020269. URL: https://www.mdpi.com/2073-4425/14/2/269.

[30] Patrick Kemmeren et al. "Large-Scale Genetic Perturbations Reveal Regulatory Networks and an Abundance of Gene-Specific Repressors". In: *Cell* 157.3 (2014), pp. 740–752. ISSN: 0092-8674. DOI: https://doi.org/10.1016/j.cell.2014.02.054. URL: https://www.sciencedirect.com/science/article/pii/S0092867414003420.

[31] Prasad Ostwal. *ostwalprasad/LGNpy: v1.0.0*. 2020. DOI: 10.5281/ZENODO.3902122. URL: https://zenodo.org/record/3902122.

[32] Nookaew I Pizarro FA Agosin E Hellgren LI Nielsen J. Jewett MC Workman CT. "Mapping condition-dependent regulation of lipid metabolism in Saccharomyces cerevisiae." In: *G3* (2013). URL: https://doi.org/10.1534/g3.113.006601.

[33] Ebrahim A, Lerman J A, Palsson B O, Hyduke D R. "COBRApy: COnstraints-Based Reconstruction and Analysis for Python". In: *BMC Systems Biology* (Aug. 2013). URL: https://doi.org/10.1186/1752-0509-7-74.

[34] Evan S Snitkin et al. "Model-driven analysis of experimentally determined growth phenotypes for 465 yeast gene deletion mutants under 16 different conditions". In: *Genome Biology* 9.9 (Sept. 2008), R140.

[35] Fred Sherman. "Getting started with yeast". In: *Guide to Yeast Genetics and Molecular and Cell Biology - Part B*. Methods in enzymology. Elsevier, 2002, pp. 3–41.

[36] Ehsan Motamedian et al. "TRFBA: an algorithm to integrate genome-scale metabolic and transcriptional regulatory networks with incorporation of expression data". In: *Bioinformatics* 33.7 (Dec. 2016), pp. 1057–1063. DOI: 10.1093/bioinformatics/btw772.

[37] Elati M. Banos D T Trébulle P. "Integrating transcriptional activity in genome-scale models of metabolism". In: *BMC Systems Biology* (2017). URL: https://doi.org/10.1186/s12918-017-0507-0.

[38] Yudong Li et al. " Genomic Evolution of Saccharomyces cerevisiae under Chinese Rice Wine Fermentation ". In: *Genome Biology and Evolution* 6.9 (Sept. 2014), pp. 2516–2526. ISSN: 1759-6653. DOI: 10.1093/gbe/evu201. eprint:

https://academic.oup.com/gbe/article-pdf/6/9/2516/17925562/evu201.pdf. URL: https://doi.org/10.1093/gbe/evu201.

[39] Rama Balakrishnan et al. "YeastMine—an integrated data warehouse for Saccharomyces cerevisiae data as a multipurpose tool-kit". In: *Database* 2012 (Feb. 2012), bar062. ISSN: 1758-0463. DOI: 10.1093/database/bar062. eprint: https://academic.oup.com/database/article-pdf/doi/10.1093/database/bar062/16728459/bar062.pdf. URL: https://doi.org/10.1093/database/bar062.

[40] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

DEPARTMENT OF LIFE SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY