

KYOTO UNIVERSITY
GRADUATE SCHOOL OF INFORMATICS

Data-Driven Frameworks for Predicting Growth-Coupled Gene Deletion Strategies in Genome-Scale Metabolic Models

ゲノムスケール代謝モデルに基づく増殖連動型遺伝子削除戦略予測のため
のデータ駆動型フレームワーク

Ziwei Yang

楊子威

Supervisor: Takeyuki Tamura

田村 武幸

A dissertation presented for the degree of
Doctor of Informatics



京都大学
KYOTO UNIVERSITY

DEPARTMENT OF INTELLIGENCE SCIENCE AND TECHNOLOGY
Graduate School of Informatics

February, 2026

Abstract

Designing gene deletion strategies to achieve growth-coupled production (GCP) of target metabolites is a central problem in metabolic engineering. Although databases of computational gene deletion strategies for genome-scale metabolic models (GEMs) exist, current methods rarely utilize these resources and instead perform de novo optimization for each metabolite, leading to redundancy and high computational costs. To address this challenge, we present three computational frameworks that progressively advance from database-enhanced optimization to deep learning-based prediction of GCP-related gene essentiality.

First, we introduce DBgDel, which incorporates prior strategies from the MetNetComp database to constrain the search space, resulting in a substantial acceleration of optimization while maintaining competitive success rates. Next, we propose DeepGDel, which reformulates gene deletion strategy design as a machine learning prediction problem using deep models on sequential representations of genes and metabolites. Finally, we develop GraphGDel, which extends DeepGDel by constructing graph representations of GEMs to jointly capture sequential patterns and metabolite interactions, thereby improving gene essentiality prediction.

Together, these frameworks represent a shift from purely optimization-based approaches toward data-driven methodologies for designing gene deletion strategies in GEMs.

Acknowledgements

During my time at Kyoto University, I have been fortunate to receive support, encouragement, and guidance from many people who have played an important role in my academic journey.

First and foremost, I would like to express my deepest gratitude to my advisor, Professor Takeyuki Tamura, for his invaluable guidance, patience, and insightful feedback throughout my doctoral studies. His expertise and thoughtful advice have been instrumental in shaping my research ideas and academic growth.

I am also sincerely grateful to all current and former members of the Akutsu Laboratory, as well as visiting students, for creating a stimulating and supportive research environment. The discussions, collaborations, and daily interactions with all of you have made my time in the lab both productive and enjoyable. It has truly been a pleasure to work and learn alongside such talented and kind colleagues.

Finally, I wish to express my heartfelt thanks to my family for their endless love, understanding, and encouragement. Your unwavering support has given me the strength and motivation to overcome challenges and complete my doctoral journey. I could not have achieved this without you.

Contents

List of Figures	ix
List of Tables	xi
1 Introduction	1
1.1 Background and Motivation	1
1.2 Research Gaps and Challenges	4
1.3 Research Objectives and Contributions	6
1.3.1 Database-Enhanced Optimization (DBgDel)	6
1.3.2 Deep Learning-Based Prediction (DeepGDel)	6
1.3.3 Graph-Enhanced Structure-Aware Learning (GraphGDel) .	7
1.4 Dissertation Organization	7
2 Preliminaries	9
2.1 Constraint-Based Metabolic Models	9
2.1.1 Metabolic Network	9
2.1.2 Gene-Protein-Reaction Rules	10
2.2 Flux Balance Analysis	10
2.3 Growth-Coupled Production	11
2.4 Gene Deletion Strategy Calculation	11
2.5 Toy Metabolic Model	12
2.5.1 Toy Model 1	12
Model Description	12
Gene Deletion Analysis	14
2.5.2 Toy Model 2	18
Model Description	18
Gene Deletion Analysis	19
2.6 Existing Computational Methods	22
2.7 MetNetComp Database	23

2.8	Deep Learning Fundamentals	23
2.8.1	Autoencoders	23
2.8.2	Long Short-Term Memory Networks	23
2.8.3	Graph Neural Networks	24
2.9	Data Representations	24
3	DBgDel	27
3.1	Introduction	27
3.2	DBgDel Framework	29
3.2.1	STEP 1: Extracting Remaining Gene Set	29
3.2.2	STEP 2: Extended gDel_minRN Algorithm	30
3.3	Toy Model Illustration	31
3.3.1	Toy Model Settings	33
3.3.2	Behaviors of the DBgDel Framework	33
3.4	Computational Experiments	36
3.4.1	Experimental Setup	36
	Baseline Methods	38
	Evaluation Metrics	38
3.4.2	Results and Analysis	39
	Performance Comparison with Baselines	39
	Impact of Initial Gene Pool Settings	40
	Comparison of PR Percentile Ranks	42
3.5	Discussion	43
4	DeepGDel	47
4.1	Introduction	47
4.2	Problem Formulation: Gene Deletion Strategy Prediction	49
4.3	DeepGDel Framework	50
4.3.1	Framework Overview	50
4.3.2	Gene Deletion Strategy Prediction	51
	Meta-M: Metabolite Representation Module	51
	Gene-M: Gene Representation Module	54
	Pred-M: Prediction Module	54
4.3.3	Implementation Details	55
	Device and Optimization	55
	Data Splitting and Balancing	55

	Hyperparameters	56
	Computational Complexity and Runtime	56
4.4	Toy Model Illustration	57
4.4.1	DeepGDel Application	57
4.4.2	BPM Baseline Illustration	57
4.5	Computational Experiments	59
4.5.1	Experimental Setup	59
	Data Balancing	59
	Baseline Methods	60
	Evaluation Metrics	61
4.5.2	Results and Analysis	61
	Performance Comparison with Baselines	61
	Ablation Study	62
4.6	Discussion	64
5	GraphGDel	67
5.1	Introduction	67
5.2	Graph Construction Pipeline	69
5.2.1	Step 1: Temporary Tripartite Graph	70
5.2.2	Step 2: Metabolite Graph Projection	70
5.2.3	Step 3: Graph Attribute-Based Node Filtering	70
5.2.4	Step 4: Knowledge-Based Node Refinement	71
5.3	GraphGDel Framework	71
5.3.1	Framework Overview	73
5.3.2	Gene-M and Meta-M	73
5.3.3	Graph-M: Graph Representation Module	74
5.3.4	Pred-M: Prediction Module	76
5.3.5	Implementation Details	76
5.4	Computational Experiments	77
5.4.1	Experimental Setup	77
	Baseline Methods	78
	Evaluation Metrics	79
5.4.2	Results and Analysis	79
	Performance Comparison with Baselines	79
	Module Ablation	79

Graph Representation Evaluation	80
5.5 Discussion	81
6 Conclusion and Future Directions	83
6.1 Summary of Contributions	83
6.2 Methodological Evolution: From Optimization to Learning	84
6.3 Limitations and Open Challenges	86
6.4 Future Directions	87
6.5 Concluding Remarks	90
References	92

List of Figures

- 2.1 A toy example of the constraint-based model where circles and rectangles represent metabolites and reactions, respectively. Black and white rectangles denote external and internal reactions, respectively. r_1, r_2 correspond to two substrate uptake reactions. r_8, r_9 correspond to cell growth, and target metabolite production reactions, respectively. The reaction rates are constrained by the range $[l_i, u_i]$. This example shows only part of the model, the rest after m_7 is omitted. 13
- 2.2 A toy example of the constraint-based model where circles and rectangles represent metabolites and reactions, respectively. Black and white rectangles denote external and internal reactions, respectively. r_1, r_2 correspond to two substrate uptake reactions. r_8, r_9 correspond to cell growth, and target metabolite production reactions, respectively. The reaction rates are constrained by the range $[l_i, u_i]$. In this example, all stoichiometric ratios are 1 or -1. 19
- 3.1 System overview of the DBgDel framework. The framework comprises two steps: (1) STEP 1, extracting the gene set G_{remain} from the MetNetComp database; and (2) STEP 2, using an extended gDel_minRN algorithm that incorporates G_{remain} to calculate gene deletion strategies for new target metabolites. 29
- 3.2 The toy model with a new target metabolite. Circles and rectangles represent metabolites and reactions, respectively. Black rectangles denote external and internal reactions. r_1, r_2 correspond to two substrate uptake reactions. r_8, r_{10} correspond to cell growth, and target metabolite production reactions, respectively. The reaction rates are constrained by the range $[l_i, u_i]$ 33

4.1	System overview of the DeepGDel framework. The framework comprises three neural network-based modules: (1) Meta-M , which learns the metabolite latent representation Z_{meta} ; (2) Gene-M , which learns the gene latent representation Z_{gene} ; and (3) Pred-M , which integrates the two upstream latents into a unified latent representation Z for final gene deletion prediction.	50
4.2	Toy model with new target metabolite m_1 for DeepGDel illustration. An auxiliary reaction r_{10} was added to simulate the secretion of m_1	58
5.1	System overview of the GraphGDel framework. The framework comprises four neural network-based modules: (1) Gene-M and (2) Meta-M , which learn latent representations from sequential data; (3) Graph-M , which learns structure-aware metabolite representations from the constructed graph; and (4) Pred-M , which integrates all representations for gene deletion prediction.	73

List of Tables

2.1	Gene deletion strategies for the toy example in Fig. 2.1 are classified into eight types according to the resulting flux distributions (reaction rates).	14
2.2	The resulting flux distribution of Types 1 to 8 gene deletion strategies. For each gene deletion strategy type, the most optimistic (best) and pessimistic (worst) PR (v_9) at GR (v_8) maximization are shown.	15
2.3	Gene deletion strategies for the toy example in Fig. 2.2 are classified into five types based on the resulting flux (reaction rate) distributions.	20
2.4	The resulting flux distributions for gene deletion strategy Types 1 to 5. For each type, the most optimistic (best) and pessimistic (worst) target metabolite production rate (PR, v_9) at cell growth rate (GR, v_8) maximization are provided.	20
3.1	The remaining genes correspond to the gene deletion strategies that enable growth-coupled production of the target metabolite in the toy example.	34
3.2	Gene deletion strategies for the toy example in Fig. 3.2 are classified into four types according to the resulting flux distributions (reaction rates).	34
3.3	The resulting flux distribution of Types 1 to 4 gene deletion strategies. For each gene deletion strategy type, the most optimistic (best) and pessimistic (worst) PR (v_{10}) at GR (v_8) maximization are shown.	35
3.4	The constraint-based models that were used in the computational experiments.	36

3.5	Performance comparison between the proposed method DBgDel (using Predicted- G_{remain} genes as the initial remaining gene pool) with GDLS, optGene, gDel_minRN, and gMCSE. Each gene deletion strategy was deemed successful if it achieved a minimum GR and PR of 0.001 or higher during GR maximization. The unit of elapsed time is second (s).	40
3.6	Performance comparison of DBgDel with different initial remaining gene pool settings. Each gene deletion strategy was deemed successful if it achieved a minimum GR and PR of 0.001 or higher during GR maximization. The unit of elapsed time is second (s). . .	42
3.7	Distribution of percentile ranks for target metabolites PRs across three metabolic models using DBgDel, with MetNetComp PRs as baselines.	43
4.1	Grid search for hyperparameter selection. The optimal values are highlighted in <u>underline</u>	56
4.2	Summary of constructed datasets for three metabolic models.	59
4.3	Performance comparison between DeepGDel and baseline methods. Best results in bold. M-Avg. = Macro-averaged.	63
4.4	Ablation comparison: Met-only vs. Gene-only inputs. Best results in bold.	64
5.1	Performance comparison: GraphGDel vs DeepGDel vs DNN. Best in bold. M-Avg. = Macro-averaged.	80
5.2	Ablation study: Ge-SMILES vs Ge-Graph vs GraphGDel. Best in bold.	80
5.3	Performance with different graph representations. Best in bold. . . .	81

Chapter 1

Introduction

1.1 Background and Motivation

Computational approaches are essential in metabolic engineering, enabling the systematic design and optimization of microbial strains for the production of valuable biochemical compounds [1, 2, 3, 4, 5]. A key application is computational strain design, which leverages mathematical models to simulate cellular metabolism and guide the optimization of target metabolite biosynthesis. Among available modeling frameworks, the **constraint-based model**—often referred to as a genome-scale metabolic model (GEM)—is widely used in genome-scale metabolic engineering [6, 7]. Genome-scale metabolic models have been developed for numerous industrially important microorganisms, with prominent examples including *Escherichia coli* and *Saccharomyces cerevisiae*, which serve as workhorse organisms for metabolic engineering applications.

A constraint-based metabolic model comprises two key components: (1) a **metabolic network** and (2) **gene-protein-reaction (GPR) rules**.

The metabolic network defines the biochemical transformations among metabolites within the cell, representing the stoichiometric relationships between substrates and products in each reaction. These reactions are catalyzed by enzymes encoded

by specific genes, establishing a direct link between genotype and metabolic phenotype. GPR rules, typically expressed as Boolean functions, describe the logical relationships between genes, proteins, and reactions, specifying whether a reaction remains active based on the presence or absence of particular genes [8]. By systematically modifying gene content, the metabolic network can be reprogrammed to achieve desired metabolic outcomes.

One widely applied genetic operation in metabolic engineering is **gene deletion**, which involves removing genes that encode essential enzymes, thereby disabling specific reactions within the metabolic network [9, 10]. This approach offers a rational means to rewire metabolism, redirecting metabolic flux toward the enhanced biosynthesis of target compounds of industrial or pharmaceutical interest. However, the objective of gene deletion in constraint-based models is not merely to maximize the production rate of a target metabolite in isolation. Rather, the goal is to achieve **growth-coupled production (GCP)**—a metabolic state in which the synthesis of the target metabolite is intrinsically coupled with cellular growth [11, 12].

Growth-coupled production is critical for industrial biotechnology applications. When a metabolite’s production is coupled with cell growth, microbial strains that produce higher amounts of the target compound also exhibit higher growth rates. During repeated passaging in industrial fermentation processes, these high-producing, fast-growing strains naturally outcompete slower-growing or non-producing variants, ensuring stable and sustained metabolite production over time without the need for continuous selection pressure.

However, under native metabolic conditions, only a limited number of metabolites naturally exhibit growth-coupling properties. For the vast majority of industrially relevant target compounds, achieving growth-coupled production requires the careful design of **gene deletion strategies**—specific sets of genes that, when removed, restructure the metabolic network such that target metabolite synthesis becomes obligatorily linked to cellular growth [13]. Identifying these gene deletion strategies is far from trivial. The complexity arises from the combinatorial nature of the problem: for a genome-scale metabolic model containing hundreds to thousands of genes, the search space of possible gene deletion combinations grows exponentially, making exhaustive enumeration computationally prohibitive.

To address this challenge, a range of computational methods have been developed over the past two decades to identify gene deletion strategies capable of enabling growth-coupled production. Early approaches, such as OptKnock [9], formulated the problem as a bi-level optimization program, where the outer level minimizes the target metabolite production rate while the inner level maximizes cellular growth rate, subject to stoichiometric and thermodynamic constraints. Subsequent methods, including OptStrain [10], OptForce [14], EMILiO [15], Redirector [16], Truncated [17], Omic [18], IdealKnock [19], FastPros [20], and Grid [21], introduced various algorithmic innovations to improve computational efficiency and solution quality. More recently, methods based on minimal cut sets (MCS) [22] and mixed-integer linear programming (MILP), such as gDel_minRN [23], have demonstrated the capability to compute gene deletion strategies for a broad range of metabolites across multiple genome-scale models of industrially relevant organisms, including *Escherichia coli* and *Saccharomyces cerevisiae*.

These computational gene deletion strategies have been successfully applied to real-world metabolic engineering challenges, facilitating the development of engineered microbial strains for applications such as biofuel production and the industrial biosynthesis of platform chemicals [24, 25, 26]. Despite these successes, a fundamental limitation persists: nearly all existing methods employ a **de novo** calculation paradigm, wherein gene deletion strategies are computed independently for each target metabolite without leveraging prior knowledge or shared patterns across different metabolites. This approach overlooks potential commonalities in the metabolic rewiring required for growth-coupled production, leading to substantial redundancy and high computational costs, particularly for genome-scale models.

In parallel with algorithmic development, recent efforts have focused on establishing dedicated databases to curate and disseminate computational gene deletion strategies. A prominent example is MetNetComp [27], the first web-based platform to systematically catalog gene deletion strategies for growth-coupled production in constraint-based metabolic models. MetNetComp employs extensions of the gDel_minRN algorithm to compute both minimal and maximal gene deletion strategies—strategies that achieve growth-coupled production with the fewest or greatest number of gene deletions, respectively. Currently, MetNetComp houses

a comprehensive repository of over 85,000 gene deletion strategies spanning numerous metabolites across multiple constraint-based models from diverse microbial species. The availability of such large-scale databases presents a significant opportunity to transition from purely algorithmic approaches toward **data-driven methodologies** that can extract, learn from, and generalize patterns embedded within existing gene deletion knowledge.

1.2 Research Gaps and Challenges

Despite the considerable progress in both algorithmic development and database curation, several critical challenges remain unaddressed in the field of computational gene deletion strategy design:

Challenge 1: Computational Inefficiency of De Novo Calculations. Current state-of-the-art methods for computing gene deletion strategies, while effective, remain computationally intensive when applied to genome-scale metabolic models. For models containing thousands of reactions and hundreds of genes, computing gene deletion strategies for even a single target metabolite can require minutes to hours of computation time, depending on the model complexity and the algorithm employed. When gene deletion strategies are needed for hundreds or thousands of metabolites within the same organism, the cumulative computational burden becomes prohibitive for routine industrial or research applications. This inefficiency stems from the de novo nature of current approaches, which treat each metabolite independently and fail to exploit shared metabolic structures or common gene deletion patterns that may exist across different targets.

Challenge 2: Underutilization of Existing Gene Deletion Databases. While databases such as MetNetComp provide extensive repositories of pre-computed gene deletion strategies, current computational methods rarely leverage this wealth of prior knowledge. The strategies stored in these databases encode valuable information about the metabolic rewiring necessary for growth-coupled production, including patterns of which genes are frequently deleted or retained, which metabolic pathways are commonly disrupted, and which core metabolic functions must remain intact to sustain cellular growth. However, extracting and integrating this information into downstream gene deletion calculations in a systematic and automated manner remains an open problem. Hand-crafted heuristics or manually designed rules can capture only limited aspects of these patterns and do not scale to the complexity and diversity of genome-scale metabolic models.

Challenge 3: Lack of Data-Driven Prediction Frameworks. The field of metabolic engineering has witnessed the successful application of machine learning and deep learning techniques to various related problems, such as predicting enzyme kinetic parameters [28], inferring metabolic pathway dynamics from multi-omics data [29], and designing synthetic pathways [30, 31]. However, the problem of gene deletion strategy design has remained largely confined to optimization-based approaches, with minimal exploration of data-driven prediction paradigms. Recent advances in deep learning [32], particularly in domains involving sequential data (e.g., protein sequences, chemical structures) and structured data (e.g., biological networks), have demonstrated remarkable capabilities in learning complex patterns and making accurate predictions. Yet, a fundamental framework that reformulates gene deletion strategy design as a prediction problem, leveraging deep learning to automatically learn from existing strategies and generalize to new target metabolites, remains absent from the literature.

Challenge 4: Ignoring the Networked Structure of Metabolic Systems. Metabolic networks are inherently structured systems in which metabolites and reactions are interconnected through complex biochemical pathways. The topology of these networks—including concepts such as metabolite hubs, pathway modularity, and upstream-downstream dependencies—plays an important role in determining metabolic phenotypes and the feasibility of growth-coupled production [33, 34, 35]. Despite this, most computational approaches, including emerging data-driven methods, primarily rely on sequential representations of genes and metabolites (e.g., amino acid sequences, chemical notation) while largely ignoring the graph-theoretic structure of the underlying metabolic network. Graph-based learning approaches, particularly graph neural networks (GNNs), have shown significant promise in capturing relational information in biological systems [36, 37, 38]. However, their application to gene deletion strategy design is hindered by a fundamental issue: genome-scale metabolic models, as typically formulated, do not explicitly define a graph structure. Instead, they are represented algebraically through stoichiometric matrices, Boolean GPR expressions, and flux bounds. Constructing biologically meaningful and computationally tractable graph representations from these algebraic formulations, while preserving essential metabolic relationships and mitigating confounding factors such as ubiquitous currency metabolites, remains a significant methodological challenge.

1.3 Research Objectives and Contributions

This dissertation addresses the aforementioned challenges through a systematic progression of three interrelated studies, each building upon the insights and limitations of its predecessor. Together, these studies establish a comprehensive framework for transitioning from purely optimization-based gene deletion strategy design to fully data-driven, structure-aware prediction methodologies.

1.3.1 Database-Enhanced Optimization (DBgDel)

The first objective is to develop a framework that integrates prior knowledge from gene deletion databases with downstream optimization algorithms to accelerate computation while maintaining high success rates.

We introduce **DBgDel** (Database-enhanced Gene Deletion framework), which addresses the inefficiency of de novo calculation by extracting core metabolic components from existing strategies in the MetNetComp database. These core components—genes consistently retained across known strategies—serve as constraints that narrow the algorithmic search space, substantially reducing computational time. DBgDel achieves an average 6.1-fold acceleration across three genome-scale metabolic models while maintaining competitive success rates, demonstrating that systematic integration of database knowledge with optimization algorithms provides a practical path toward more efficient gene deletion strategy calculation.

1.3.2 Deep Learning-Based Prediction (DeepGDel)

While DBgDel successfully integrates database knowledge to accelerate optimization, it relies on hand-crafted extraction rules and remains dependent on downstream optimization algorithms. The second objective is to fundamentally reformulate gene deletion strategy design as a **prediction problem** and develop a deep learning framework that automatically learns patterns from existing strategies.

We introduce **DeepGDel** (Deep learning-based Gene Deletion framework), which represents the first systematic formulation of gene deletion strategy design as a supervised machine learning task. By learning intrinsic characteristics of genes and metabolites from their sequential representations and their pairwise interdependencies from observed strategies, DeepGDel enables end-to-end prediction without requiring explicit optimization or manual rule engineering. DeepGDel achieves

substantial improvements in overall accuracy (14.69%, 22.52%, and 13.03% across three models) compared to baseline methods, while enabling millisecond-scale predictions compared to hours required by optimization-based approaches, demonstrating the feasibility and effectiveness of the data-driven prediction paradigm.

1.3.3 Graph-Enhanced Structure-Aware Learning (GraphGDel)

Although DeepGDel successfully transitions to a data-driven prediction paradigm, it operates solely on sequential representations of genes and metabolites, thereby ignoring the inherent network structure of metabolic systems. The third objective is to incorporate the topological relationships among metabolites into the prediction framework to further enhance performance.

We introduce **GraphGDel** (Graph-based Gene Deletion framework), which extends DeepGDel by integrating graph-based learning. GraphGDel addresses two fundamental challenges: systematically constructing biologically meaningful graph representations from the algebraic formulations of genome-scale metabolic models, and mitigating confounding effects of ubiquitous currency metabolites that obscure pathway-specific structure. GraphGDel achieves the best overall performance among all methods, with accuracy improvements of 14.0%, 16.3%, and 13.2% over baselines across three models, and further gains of 2–5% over DeepGDel on large-scale models, demonstrating that incorporating metabolic network topology provides meaningful additional signal beyond sequential representations alone.

1.4 Dissertation Organization

This dissertation is organized as follows:

Chapter 2: Preliminaries provides the foundational concepts and mathematical formulations necessary to understand the subsequent chapters. We introduce constraint-based metabolic models, flux balance analysis (FBA), growth-coupled production, and the formulation of gene deletion strategy calculation as an optimization problem. We also review existing computational methods, the MetNetComp database, and relevant deep learning techniques, including autoencoders, LSTM networks, and graph neural networks. A toy metabolic model is introduced and used throughout Chapters 3, 4, and 5 to illustrate the application of each framework.

Chapter 3: DBgDel—Database-Enhanced Gene Deletion Framework presents the first contribution of this dissertation. We formulate the problem of integrating database knowledge with optimization algorithms, describe the DBgDel framework in detail, and provide a step-by-step illustration using the toy model. We then present comprehensive computational experiments on three genome-scale metabolic models (e_coli_core, iMM904, iML1515), comparing DBgDel against state-of-the-art baseline methods. We analyze the impact of different initial gene pool settings and discuss the trade-offs between computational efficiency and solution quality.

Chapter 4: DeepGDel—Deep Learning Framework for Gene Deletion Prediction presents the second contribution, marking the transition from optimization to prediction. We reformulate gene deletion strategy design as a supervised learning problem and introduce the DeepGDel framework, detailing its three-module architecture and the sequential data encoding strategies for genes and metabolites. We provide a detailed illustration using the toy model and present extensive computational experiments, including performance comparisons with baseline methods and ablation studies to assess the contribution of each module.

Chapter 5: GraphGDel—Graph-Enhanced Gene Deletion Prediction presents the third contribution, extending DeepGDel with graph-based learning. We first present the four-step pipeline for constructing metabolic graphs from algebraic GEM representations, addressing the challenges of currency metabolites and topological noise. We then introduce the GraphGDel framework, which integrates graph neural networks with the DeepGDel architecture. Comprehensive experiments evaluate GraphGDel against DeepGDel and other baselines, and we analyze the impact of different graph construction strategies and refinement techniques.

Chapter 6: Conclusion and Future Directions synthesizes the contributions of this dissertation, providing a comparative analysis of the three frameworks across multiple dimensions, including computational efficiency, prediction accuracy, scalability, and interpretability. We discuss the biological and computational insights gained from the progression of methods, acknowledge the limitations of the current work, and outline promising directions for future research, including the integration of transformer architectures, transfer learning across organisms, multi-objective optimization, and the development of explainable AI techniques for metabolic engineering applications.

Chapter 2

Preliminaries

This chapter provides the foundational concepts and mathematical formulations necessary to understand the subsequent chapters of this dissertation. We introduce constraint-based metabolic models, flux balance analysis, growth-coupled production, and the formulation of gene deletion strategy calculation as an optimization problem. We also review existing computational methods for gene deletion strategy design, the MetNetComp database, and relevant deep learning techniques. Finally, we present a toy metabolic model that will be used throughout Chapters 3, 4, and 5 to illustrate the application of each framework.

2.1 Constraint-Based Metabolic Models

Let $\mathcal{C} = \{M, R, S, L, U, G, F, P\}$ be a constraint-based model. The elements of $\mathcal{C}$ can be further divided into two parts: (1) the metabolic network and (2) the GPR rule.

2.1.1 Metabolic Network

Let $\mathcal{C}_1 = \{M, R, S, L, U\}$ be a metabolic network. $M = \{m_1, \dots, m_{|M|}\}$ denotes a set consisting all metabolites, with one of them being the target metabolite m_{target} . $R = \{r_1, \dots, r_{|R|}\}$ denotes a set consisting all reactions, including the cell growth

reaction r_{growth} and the target metabolite production reaction r_{target} . To facilitate clarity, we introduce an additional set $V = \{v_1, \dots, v_{|R|}\}$ to represent the reaction rates per unit time (flux) corresponding to reactions in R . Notably, we distinguish the rate of the growth reaction (v_{growth}) as the **Growth Rate (GR)** and that of the target metabolite production reaction (v_{target}) as the **Production Rate (PR)**. The stoichiometry matrix S contains elements $S_{ij} = k$, indicating that reaction r_j either produces (+) or consumes (-) k units of metabolite m_i per unit time. Lower and upper bounds for the reaction rates for V are denoted by $L = \{l_1, \dots, l_{|R|}\}$ and $U = \{u_1, \dots, u_{|R|}\}$, respectively.

2.1.2 Gene-Protein-Reaction Rules

Let $\mathcal{C}_2 = \{G, F, P\}$ be a GPR rule. $G = \{g_1, \dots, g_{|G|}\}$ represents a set of genes, while $F = \{f_1, \dots, f_{|R|}\}$ represents Boolean functions. $P = \{p_1, \dots, p_{|R|}\}$ is a collection of outputs generated by applying the functions in F to the gene set G . Each output p_j can be expressed as $p_j = f_j(G)$, where both p and g are binary values, i.e., $p, g \in \{0, 1\}$. Note that if $p_j = 0$, this imposes a constraint on both the lower bound l_j and the upper bound u_j to be 0, effectively restraining reaction r_j .

2.2 Flux Balance Analysis

When analyzing the metabolic network within a constraint-based model, flux balance analysis (FBA) assumes steady states where all metabolic reaction speeds (fluxes) are constant [6]. Specifically, we give the following definitions for FBA: (1) for each compound, the sum of the producing fluxes is equal to the sum of the consuming fluxes; (2) in each reaction, the fluxes of substrates and products must satisfy the ratio in the chemical reaction equation, and (3) the upper and lower bounds are given for each flux. In the standard procedure for FBA on the given metabolic network $\mathcal{C}_1$, the objective is to maximize the cell growth reaction rate v_{growth} using the following linear programming (LP) given by Algorithm 1:

Algorithm 1 LP Formalization of FBA Using $\mathcal{C}_1$ of a Constraint-Based Model.

Given: $\mathcal{C}_1$

Maximize: v_{growth}

1: **Such that:**

2: $\sum_j S_{ij}v_j = 0$ for all i ;

3: $l_j \leq v_j \leq u_j$ for all j ;

4: $i = \{1, \dots, |M|\}, \quad j = \{1, \dots, |R|\}$

2.3 Growth-Coupled Production

Growth-coupled production is a special situation of microorganism's metabolic processes, wherein the cell growth co-occurs with the synthesis of the target metabolite. We adopt the paradigm of **weakly growth-coupled production**, which requires that the synthesis rate of the target metabolite is equal or larger than the threshold if growth rate is equal or larger than the threshold. This differs from **strongly growth-coupled production**, where the synthesis rate of the target metabolite is equal or larger than the threshold for any growth rate [11, 12]. Weakly GCP is generally more realistic to achieve and sufficient for industrial applications.

In this dissertation, we consider the synthesis rate of the target metabolite under the condition of maximizing the growth rate. During simulations of constraint-based models, growth-coupled production is defined by two indices: the lower bound of the target metabolite production rate ($PR_{\text{threshold}}$) and the lower bound of the cell growth reaction rate ($GR_{\text{threshold}}$). When the GR is maximized, growth-coupled production is considered achieved if the simulated fluxes v_{target} and v_{growth} exceed their respective lower bounds in all cases, with both thresholds set to $PR_{\text{threshold}} = GR_{\text{threshold}} = 0.001$. This can be regarded as weakly growth-coupled production, because the growth threshold is defined by the maximized growth rate itself, which is distinct from the fixed growth threshold $GR_{\text{threshold}}$ described above.

2.4 Gene Deletion Strategy Calculation

In metabolic model simulations and calculations, the gene deletion strategy refers to selecting genes to be deleted to optimize metabolic networks for desired outcomes, such as the growth-coupled production of specific metabolites. The calculation of the gene deletion strategy for growth-coupled production is typically defined as finding a set G_{del} of genes designated for deletion, as presented below in Algorithm 2.

This formulation represents a bi-level optimization problem where the outer objective minimizes the production rate while the inner objective maximizes the growth rate, subject to stoichiometric, thermodynamic, and GPR constraints.

Algorithm 2 Calculation of Gene Deletion Strategy for Growth-Coupled Production.

Given: $\mathcal{C}, r_{\text{target}}, PR_{\text{threshold}}, GR_{\text{threshold}}$

Find: $G_{\text{del}} \subset G$ such that $v_{\text{target}} \geq PR_{\text{threshold}}$ and $v_{\text{growth}} \geq GR_{\text{threshold}}$

- 1: **Such that Minimize:**
 - 2: v_{target}
 - 3: **Such that Maximize:**
 - 4: v_{growth}
 - 5: **Such that:**
 - 6: $\sum_j S_{ij}v_j = 0$ for all i ;
 - 7: $\begin{cases} v_j = 0 & \text{if } p_j = 0, \\ l_j \leq v_j \leq u_j & \text{otherwise;} \end{cases}$
 - 8: $p_j = f_j(G)$;
 - 9: $\begin{cases} g = 0 & \text{if } g \in G_{\text{del}}, \\ g = 1 & \text{otherwise;} \end{cases}$
-

2.5 Toy Metabolic Model

To illustrate the application of the frameworks introduced in subsequent chapters, we present two toy constraint-based metabolic models that demonstrate different structural configurations.

2.5.1 Toy Model 1

Model Description

Fig. 2.1 shows a toy example where $\mathcal{C} = \{M, R, S, L, U, G, F, P\}$ represents a constraint-based model. Specifically, for its metabolic network part $\mathcal{C}_1$, we have $M = \{m_1, \dots, m_7\}$, $R = \{r_1, \dots, r_9\}$,

$$S = \begin{pmatrix} 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \end{pmatrix}, \quad (2.1)$$

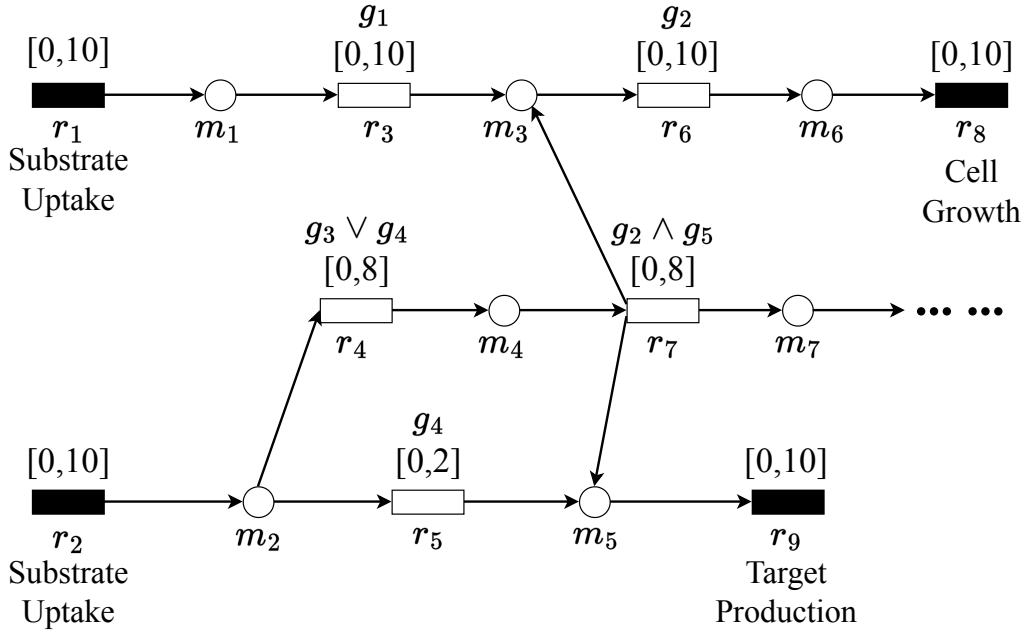


Figure 2.1: A toy example of the constraint-based model where circles and rectangles represent metabolites and reactions, respectively. Black and white rectangles denote external and internal reactions, respectively. r_1, r_2 correspond to two substrate uptake reactions. r_8, r_9 correspond to cell growth, and target metabolite production reactions, respectively. The reaction rates are constrained by the range $[l_i, u_i]$. This example shows only part of the model, the rest after m_7 is omitted.

and $L = \{0, 0, 0, 0, 0, 0, 0, 0, 0\}$, $U = \{10, 10, 10, 8, 2, 10, 8, 10, 10\}$. For its gene reaction rule part $\mathcal{C}_2$, we have $G = \{g_1, \dots, g_5\}$, $F = \{f_1, \dots, f_9\}$, and $P = \{p_1, \dots, p_9\}$, where

$$f_i : p_i = \begin{cases} 1, & i = 1 \\ 1, & i = 2 \\ g_1, & i = 3 \\ g_3 \vee g_4, & i = 4 \\ g_4, & i = 5 \\ g_2, & i = 6 \\ g_2 \wedge g_5, & i = 7 \\ 1, & i = 8 \\ 1, & i = 9. \end{cases}$$

Table 2.1: Gene deletion strategies for the toy example in Fig. 2.1 are classified into eight types according to the resulting flux distributions (reaction rates).

Type	Gene deletion strategies classified by flux distributions
1	$\emptyset, \{g_3\}$
2	$\{g_1\}, \{g_1, g_3\}$
3	$\{g_2\}, \{g_1, g_2\}, \{g_1, g_5\}, \{g_2, g_3\}, \{g_2, g_5\}, \{g_1, g_2, g_3\},$ $\{g_1, g_2, g_5\}, \{g_1, g_3, g_5\}, \{g_2, g_3, g_5\}, \{g_1, g_2, g_3, g_5\}$
4	$\{g_4\}$
5	$\{g_5\}, \{g_3, g_5\}$
6	$\{g_1, g_4\}$
7	$\{g_2, g_4\}, \{g_1, g_2, g_4\}, \{g_1, g_3, g_4\}, \{g_1, g_4, g_5\}, \{g_2, g_3, g_4\},$ $\{g_2, g_4, g_5\}, \{g_1, g_2, g_3, g_4\}, \{g_1, g_2, g_4, g_5\}, \{g_1, g_3, g_4, g_5\},$ $\{g_2, g_3, g_4, g_5\}, \{g_1, g_2, g_3, g_4, g_5\}$
8	$\{g_3, g_4\}, \{g_4, g_5\}, \{g_3, g_4, g_5\}$

In this toy model, the gene reaction rule for r_7 is given as $f_7 : p_7 = g_2 \wedge g_5$, the reaction rate of r_7 (denoted as v_7) is forced to be 0 if one of g_2 or g_5 is 0 (deleted), while $0 \leq v_7 \leq 8$ is held if both of g_2 and g_5 are 1 (not deleted). However, for r_4 , its reaction rate v_4 is forced to be 0 only if both g_3 and g_4 are 0, while $0 \leq v_4 \leq 8$ is held if at least one of g_3 or g_4 is 1. For r_1, r_2, r_8 and r_9 , since $p_1 = p_2 = p_8 = p_9 = 1$ always holds, none of v_1, v_2, v_8 or v_9 can be forced to be 0 by any gene deletions.

In this example, we set m_5 as the target metabolite produced by the target metabolite production reaction r_9 .

Gene Deletion Analysis

Table 2.1 describes the patterns of gene deletions in this example: $2^5 = 32$ patterns are classified into eight types according to the flux distributions. The detailed flux distributions for all types can be found in Table 2.2.

In the original state of the toy model, where no genes are deleted, when we maximize GR, we can obtain its maximum value $\max(GR) = \max(v_8) = 10$. However, there are two paths to reach from the substrate uptake reactions r_1 and r_2 to the growth reaction r_8 : ($r_1 \rightarrow r_3 \rightarrow r_6 \rightarrow r_8$) and ($r_2 \rightarrow r_4 \rightarrow r_7 \rightarrow r_6 \rightarrow r_8$). Besides, there is another path to reach from the substrate uptake reactions r_2 to the target production reaction r_9 : ($r_2 \rightarrow r_5 \rightarrow r_9$). If all three paths mentioned above are used, and the second and third paths achieve maximum fluxes, $GR = 10$ and

Table 2.2: The resulting flux distribution of Types 1 to 8 gene deletion strategies. For each gene deletion strategy type, the most optimistic (best) and pessimistic (worst) PR (v_9) at GR (v_8) maximization are shown.

ID	Gene deletion strategy	PR situation	Flux distribution								
			v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9
1	Type 1	best	2	10	2	8	2	10	8	10	10
2		worst	10	0	10	0	0	10	0	10	0
3	Type 2	best	0	10	0	8	2	8	8	8	10
4		worst	0	8	0	8	0	8	8	8	8
5	Type 3	best	0	2	0	0	2	0	0	0	2
6		worst	0	0	0	0	0	0	0	0	0
7	Type 4	best	2	8	2	8	0	10	8	10	8
8		worst	10	0	10	0	0	10	0	10	0
9	Type 5	best	10	2	10	0	2	10	0	10	2
10		worst	10	0	10	0	0	10	0	10	0
11	Type 6	best	0	8	0	8	0	8	8	8	8
12		worst	0	8	0	8	0	8	8	8	8
13	Type 7	best	0	0	0	0	0	0	0	0	0
14		worst	0	0	0	0	0	0	0	0	0
15	Type 8	best	10	0	10	0	0	10	0	10	0
16		worst	10	0	10	0	0	10	0	10	0

$PR = 10$ can be obtained, as shown in ID 1 in Table 2.2. This is the most optimistic case regarding the value of PR. However, if only the first path is used, $GR = 10$ and $PR = 0$ can be obtained, as shown in ID 2 of Table 2.2. This is the most pessimistic case regarding the value of PR. In this study, we evaluate the most pessimistic value of PR when the GR is maximized. Therefore, in the original state of the toy model, we have $GR = 10$ and $PR = 0$. Gene deletion strategy $\{g_3\}$ is also classified as Type 1 because the same flux distribution is obtained.

IDs 3 and 4 of Table 2.2 describe the most optimistic and pessimistic flux distributions regarding the PR value when g_1 is deleted. When g_1 is deleted, $f_3 : p_3 = g_1 = 0$ is obtained and v_3 is forced to be zero. The maximum GR is $v_8 = 8$ since only the second path reaches the growth reaction (from r_2 to r_8) can be used. In the most optimistic case, the third path ($r_2 \rightarrow r_5 \rightarrow r_9$) that reach from the substrate uptake reaction r_2 to the target production reaction r_9 achieve maximum fluxes, therefore $GR = 10$ and $PR = 10$ are obtained. In the most pessimistic case, the above path is not used, therefore $GR = 10$ and $PR = 8$ are obtained. The gene deletion strategy $\{g_1, g_3\}$ is also classified as Type 2 as shown in Table 2.2.

IDs 5 and 6 of Table 2.2 describe the flux distributions when g_2 is deleted. When g_2 is deleted, $f_6 : p_6 = g_2 = 0$ and $f_7 : p_7 = g_2 \wedge g_5 = 0$ are obtained, both v_6 and v_7 are forced to be zero. The maximum GR is $v_8 = 0$ since none of the two paths (from r_1 or r_2 to r_8) that reach cell growth reaction can be used. In the most optimistic case, the path ($r_2 \rightarrow r_5 \rightarrow r_9$) that reach from the substrate uptake reaction r_2 to the target production reaction r_9 achieve maximum fluxes, therefore $GR = 0$ and $PR = 2$ are obtained. In the most pessimistic case, the above path is not used, therefore $GR = 0$ and $PR = 0$ are obtained. The gene deletion strategies $\{g_1, g_2\}$, $\{g_1, g_5\}$, $\{g_2, g_3\}$, $\{g_2, g_5\}$, $\{g_1, g_2, g_3\}$, $\{g_1, g_2, g_5\}$, $\{g_1, g_3, g_5\}$, $\{g_2, g_3, g_5\}$, and $\{g_1, g_2, g_3, g_5\}$ are also classified as Type 3 as shown in Table 2.2.

IDs 7 and 8 of Table 2.2 describe the flux distributions when g_4 is deleted. When g_4 is deleted, $f_5 : p_5 = g_4 = 0$ is obtained, and v_5 is forced to be zero. The maximum GR is $v_8 = 10$ since both of the two paths (from r_1 or r_2 to r_8) that reach cell growth reaction can be used. In the most optimistic case, both two path above are used and the second one achieves maximum fluxes, therefore $GR = 10$ and $PR = 8$ are obtained. In the most pessimistic case, only the first path (from r_1 to r_8) is used, therefore $GR = 10$ and $PR = 0$ are obtained.

IDs 9 and 10 of Table 2.2 describe the flux distributions when g_5 is deleted. When g_5 is deleted, $f_7 : p_7 = g_2 \wedge g_5 = 0$ is obtained, and v_7 is forced to be

zero. The maximum GR is $v_8 = 10$ since only the first path (from r_1 to r_8) that reaches cell growth reaction can be used. In the most optimistic case, the path ($r_2 \rightarrow r_5 \rightarrow r_9$) that reach from the substrate uptake reaction r_2 to the target production reaction r_9 is used and achieves maximum fluxes, therefore $GR = 10$ and $PR = 2$ are obtained. In the most pessimistic case, only the first path (from r_1 to r_8) is used, therefore $GR = 10$ and $PR = 0$ are obtained. The gene deletion strategy $\{g_3, g_5\}$ is also classified as Type 5 as shown in Table 2.2.

IDs 11 and 12 of Table 2.2 describe the flux distributions when $\{g_1, g_4\}$ are deleted. When $\{g_1, g_4\}$ are deleted, $f_3 : p_3 = g_1 = 0$ and $f_5 : p_5 = g_4 = 0$ are obtained, both v_3 and v_5 are forced to be zero. The flux distribution is uniquely determined, $v_1 = 0$, $v_2 = v_4 = v_6 = v_7 = 8$ are obtained, and GR and PR are 8 for both optimistic and pessimistic cases regarding the PR value.

IDs 13 and 14 of Table 2.2 describe the flux distributions when $\{g_2, g_4\}$ are deleted. When $\{g_2, g_4\}$ are deleted, $f_5 : p_5 = g_4 = 0$ and $f_6 : p_6 = g_2 = 0$ are obtained, both v_5 and v_6 are forced to be zero. The maximum GR is $v_8 = 0$ since none of the two paths (from r_1 or r_2 to r_8) that reach cell growth reaction can be used. The flux distribution is uniquely determined where all reaction rates become zero, therefore PR is 0 for both optimistic and pessimistic cases regarding the PR value. The gene deletion strategies $\{g_1, g_2, g_4\}$, $\{g_1, g_3, g_4\}$, $\{g_1, g_4, g_5\}$, $\{g_2, g_3, g_4\}$, $\{g_2, g_4, g_5\}$, $\{g_1, g_2, g_3, g_4\}$, $\{g_1, g_2, g_4, g_5\}$, $\{g_1, g_3, g_4, g_5\}$, $\{g_2, g_3, g_4, g_5\}$, and $\{g_1, g_2, g_3, g_4, g_5\}$ are also classified as Type 7 as shown in Table 2.2.

IDs 15 and 16 of Table 2.2 describe the flux distributions when $\{g_3, g_4\}$ are deleted. When $\{g_3, g_4\}$ are deleted, $f_4 : p_4 = g_3 \vee g_4 = 0$ and $f_5 : p_5 = g_4 = 0$ are obtained, both v_4 and v_5 are forced to be zero. The maximum GR is $v_8 = 10$ since only the first path (from r_1 to r_8) can be used. The flux distribution is uniquely determined, $v_1 = v_3 = v_6 = 10$, $v_2 = v_7 = 0$ are obtained, and PR is 0 for both optimistic and pessimistic cases regarding the PR value. The gene deletion strategies $\{g_4, g_5\}$ and $\{g_3, g_4, g_5\}$ are also classified as Type 8 as shown in Table 2.2.

Through all the cases above, we find that for gene deletion strategies of Types 2 and 6, even in the most pessimistic case, we still meet the criteria of $PR \geq PR_{\text{threshold}}$ and $GR \geq GR_{\text{threshold}}$. Consequently, in our example of the toy model, these two types of deletions enable growth-coupled production of the target metabolite m_5 .

2.5.2 Toy Model 2

Model Description

Fig. 2.2 shows a toy example in which $\mathcal{C} = \{M, R, S, L, U, G, F, P\}$ represents a constraint-based model. Specifically, for its metabolic network part $\mathcal{C}_1$, we have $M = \{m_1, \dots, m_6\}$, $R = \{r_1, \dots, r_9\}$,

$$S = \begin{pmatrix} 1 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 \\ 0 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & -1 \end{pmatrix}, \quad (2.2)$$

and $L = \{0, 0, 0, 0, 0, 0, 0, 0, 0\}$, $U = \{10, 10, 6, 4, 10, 6, 4, 10, 10\}$. For its gene reaction rule part $\mathcal{C}_2$, we have $G = \{g_1, \dots, g_5\}$, $F = \{f_1, \dots, f_9\}$, and $P = \{p_1, \dots, p_9\}$, where

$$f_i : p_i = \begin{cases} 1, & i = 1 \\ 1, & i = 2 \\ g_1, & i = 3 \\ g_2, & i = 4 \\ g_3 \vee g_4, & i = 5 \\ g_4, & i = 6 \\ g_2 \wedge g_5, & i = 7 \\ 1, & i = 8 \\ 1, & i = 9 \end{cases}$$

In this toy model, because the gene reaction rule of r_7 is given as $f_7 : p_7 = g_2 \wedge g_5$, the reaction rate of r_7 (denoted as v_7) is forced to be 0 if either g_2 or g_5 is 0 (deleted), while if both g_2 and g_5 are 1 (not deleted), then $0 \leq v_7 \leq 4$. However, for r_5 , the reaction rate v_5 is forced to be 0 only if both g_3 and g_4 are 0. However, if at least one of g_3 or g_4 is 1, v_5 can vary within the range $0 \leq v_5 \leq 10$. For r_1 , r_2 , r_8 , and r_9 , since $p_1 = p_2 = p_8 = p_9 = 1$ always holds, none of v_1 , v_2 , v_8 , or v_9 can be forced to be 0 by any gene deletions.

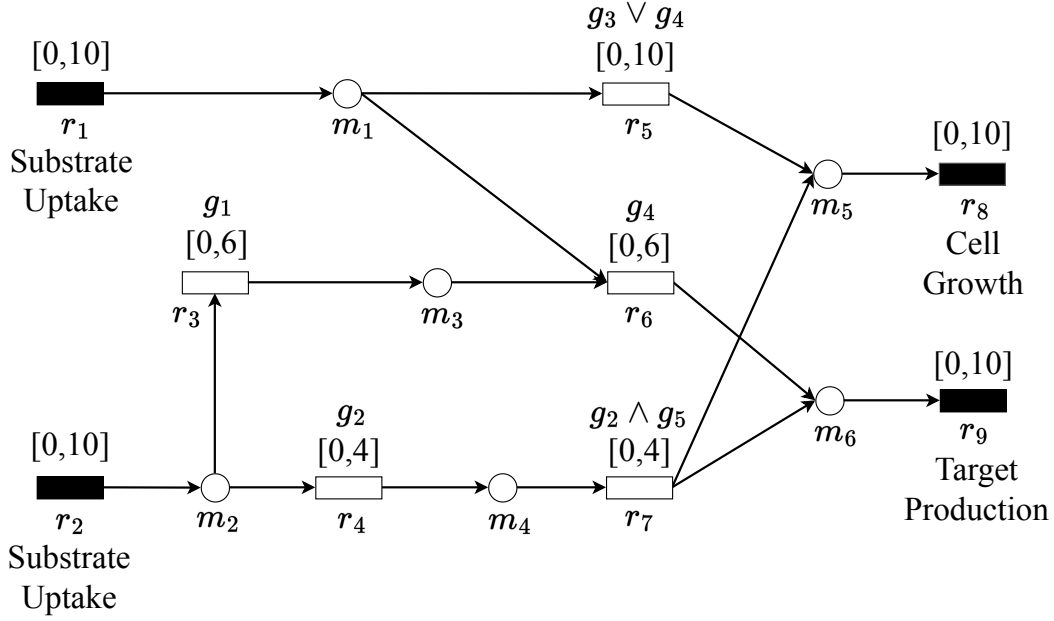


Figure 2.2: A toy example of the constraint-based model where circles and rectangles represent metabolites and reactions, respectively. Black and white rectangles denote external and internal reactions, respectively. r_1, r_2 correspond to two substrate uptake reactions. r_8, r_9 correspond to cell growth, and target metabolite production reactions, respectively. The reaction rates are constrained by the range $[l_i, u_i]$. In this example, all stoichiometric ratios are 1 or -1.

In this example, we set m_6 as the target metabolite, produced by the target metabolite production reaction r_9 .

Gene Deletion Analysis

Table 2.3 describes the patterns of gene deletions in this example: the $2^5 = 32$ patterns are classified into five types based on the flux distributions. The detailed flux distributions for all types can be found in Table 2.4.

In the original state of the toy model, where no genes are deleted, there are two paths to reach the cell growth reaction r_8 : one begins from the substrate uptake reaction r_1 : ($r_1 \rightarrow r_5 \rightarrow r_8$); the other one begin from the substrate uptake reaction r_2 : ($r_2 \rightarrow r_4 \rightarrow r_7 \rightarrow r_8$). $GR = 10$ can be obtained, but the flux distribution is not uniquely determined.

If both path are used and the second one ($r_2 \rightarrow r_4 \rightarrow r_7 \rightarrow r_8$) achieves maximum flux, $GR = 10$ and $PR = 8$ can be obtained, as shown in the flux distribution of ID 1 in Table 2.4. This represents the most optimistic case for the value of PR. In contrast, if only the first path ($r_1 \rightarrow r_5 \rightarrow r_8$) is used and achieves maximum

flux, then $GR = 10$ and $PR = 0$ are obtained, as shown in the flux distribution of ID 2 in Table 2.4. This represents the most pessimistic case for the value of PR. As previously defined, in the context of growth-coupled production, we need to evaluate the most pessimistic value of PR when GR is maximized. Therefore, in the original state of the toy model, we have $GR = 10$ and $PR = 0$. We denote this flux distribution as Type 1. There is another gene deletion strategy, $\{g_3\}$, which is also classified as Type 1, as it yield the same flux distribution.

IDs 3 and 4 in Table 2.4 describe the most optimistic and pessimistic flux distributions with respect to the PR value when g_1 is deleted. When g_1 is deleted, $f_3 : p_3 = g_1 = 0$ is obtained, and v_3 is forced to be zero. The maximum GR is $v_8 = 10$ since both the first and second paths to the growth reaction r_9 (from r_1 or r_2 to r_8) can still be used. In the most optimistic case, both the first path ($r_1 \rightarrow r_5 \rightarrow r_8$) and the second path ($r_2 \rightarrow r_4 \rightarrow r_7 \rightarrow r_8$) are used, with the second one achieving maximum flux; therefore, $GR = 10$ and $PR = 4$ are obtained. In the most pessimistic case, only the first path is used and achieves maximum flux, resulting in $GR = 10$ and $PR = 0$. We denote this flux distribution as Type 2. The gene deletion strategies $\{g_4\}$, $\{g_1, g_3\}$, and $\{g_1, g_4\}$ are also classified as Type 2, as shown in Table 2.4.

IDs 5 and 6 in Table 2.4 describe the flux distributions when g_2 is deleted. When g_2 is deleted, $f_4 : p_4 = g_2 = 0$ and $f_7 : p_7 = g_2 \wedge g_5 = 0$ are obtained, forcing v_4 and v_7 to be zero. The maximum GR is $v_8 = 10$ since the first path to the growth reaction (from r_1 to r_8) can still be used. This results in a uniquely determined flux distribution where $v_1 = v_5 = 10$ and $v_2 = v_3 = v_4 = v_6 = v_7 = 0$. Consequently, both the optimistic and pessimistic PR values are 0, with GR at 10. We denote this flux distribution as Type 3. Other 17 gene deletion strategies are also classified as Type 3, as shown in Table 2.4.

IDs 7 and 8 in Table 2.4 describe the flux distributions when g_3 and g_4 are deleted. When g_3 and g_4 are deleted, $f_5 : p_5 = g_3 \vee g_4 = 0$ and $f_6 : p_6 = g_4 = 0$ are obtained, forcing v_5 and v_6 to be zero. This results in a uniquely determined flux distribution where $v_2 = v_4 = v_7 = 4$ and $v_1 = v_3 = v_5 = v_6 = 0$. Consequently, both the optimistic and pessimistic PR values are 4, with GR also at 4. We denote this flux distribution as Type 4. Another gene deletion strategy $\{g_1, g_3, g_4\}$ is also classified as Type 4, as shown in Table 2.4.

IDs 9 and 10 in Table 2.4 describe the flux distributions when g_2 , g_3 , and g_4 are deleted. When g_2 , g_3 , and g_4 are deleted, $f_4 : p_4 = g_2 = 0$, $f_5 : p_5 = g_3 \vee g_4 = 0$,

$f_6 : p_6 = g_4 = 0$, and $f_7 : p_7 = g_2 \wedge g_5 = 0$ are obtained, forcing v_4, v_5, v_6 , and v_7 to be zero. The maximum GR is $v_8 = 0$, as no path to the growth reaction r_8 remains viable. This results in a uniquely determined flux distribution in which all reaction rates are zero, yielding a PR of 0 in both optimistic and pessimistic cases. We denote this flux distribution as Type 5. The gene deletion strategies $\{g_3, g_4, g_5\}$, $\{g_1, g_2, g_3, g_4\}$, $\{g_1, g_3, g_4, g_5\}$, $\{g_2, g_3, g_4, g_5\}$, and $\{g_1, g_2, g_3, g_4, g_5\}$ are also classified as Type 5, as shown in Table 2.4.

Across the five types of gene deletion strategies analyzed, we find that only Type 4 meets the criteria of $PR \geq PR_{\text{threshold}}$ and $GR \geq GR_{\text{threshold}}$, even in the most pessimistic scenarios. Consequently, in our toy model example, only two gene deletion strategies of Type 4, i.e., $\{g_3, g_4\}$ and $\{g_1, g_3, g_4\}$ enable the growth-coupled production of the target metabolite m_6 .

2.6 Existing Computational Methods

Over the past two decades, numerous computational methods have been developed to address the gene deletion strategy calculation problem for growth-coupled production. Early approaches (2003-2010), such as OptKnock [9], formulated the problem as a bi-level optimization program using mixed-integer linear programming (MILP). Subsequent methods, including OptStrain [10], OptForce [14], EMILiO [15], Redirector [16], Truncated [17], Omic [18], IdealKnock [19], FastPros [20], and Grid [21], introduced various algorithmic innovations to improve computational efficiency and solution quality.

More recently (2017 onwards), methods based on minimal cut sets (MCS) [22] have demonstrated the capability to compute gene deletion strategies for a broad range of metabolites across multiple genome-scale models. The basic idea of the MCS-based method is to identify the smallest possible set of gene knockouts that force a coupling between cell growth and target metabolite production, with computations based on Farkas Lemma [39]. Tamura et al. introduced gDel_minRN [23], a method that operates by maximizing the number of repressed reactions, thereby extracting essential core components for growth-coupled production. Experimental validations have demonstrated that gDel_minRN stands out as one of the most effective approaches currently available for calculating gene deletions in genome-scale models.

2.7 MetNetComp Database

Recent efforts have focused on establishing dedicated databases to curate and disseminate computational gene deletion strategies. A prominent example is MetNetComp [27], the first web-based platform to systematically catalog gene deletion strategies for growth-coupled production in constraint-based metabolic models. MetNetComp employs extensions of the gDel_minRN algorithm to compute both minimal and maximal gene deletion strategies—strategies that achieve growth-coupled production with the fewest or greatest number of gene deletions, respectively. Currently, MetNetComp houses a comprehensive repository of over 85,000 gene deletion strategies spanning numerous metabolites across multiple constraint-based models from diverse microbial species, including *E. coli* and *S. cerevisiae*.

2.8 Deep Learning Fundamentals

Recent advances in deep learning [32] have demonstrated remarkable capabilities in learning complex patterns and making accurate predictions across diverse domains, including biological applications.

2.8.1 Autoencoders

Autoencoders are neural network architectures designed to learn efficient representations of input data in an unsupervised manner. An autoencoder consists of two main components: an encoder that maps input data to a latent representation, and a decoder that reconstructs the input from the latent representation. By minimizing the reconstruction error, autoencoders learn to capture essential features of the input data in the latent space, which can then be used for downstream tasks. In this dissertation, autoencoders are employed for unsupervised representation learning of sequential data, particularly for encoding gene amino acid sequences and metabolite chemical structures.

2.8.2 Long Short-Term Memory Networks

Long Short-Term Memory (LSTM) networks [40] are a type of recurrent neural network (RNN) specifically designed to address the vanishing gradient problem in

learning long-range dependencies in sequential data. LSTMs use gating mechanisms (forget gate, input gate, output gate) to selectively remember or forget information across time steps, making them particularly effective for modeling sequential patterns in biological data. In this dissertation, LSTMs are utilized to encode sequential representations, specifically processing chemical structure notation (SMILES) for metabolites and amino acid sequences for genes and proteins.

2.8.3 Graph Neural Networks

Graph Neural Networks (GNNs) are a class of deep learning models designed to operate on graph-structured data [41]. GNNs learn node representations by iteratively aggregating information from neighboring nodes through message passing mechanisms. Common GNN variants include Graph Convolutional Networks (GCN), Graph Attention Networks (GAT), and GraphSAGE. In biological applications, GNNs have shown promise in capturing relational information in molecular graphs, protein-protein interaction networks, and metabolic networks. In this dissertation, Graph Convolutional Networks (GCN) are employed as the specific GNN architecture for learning structure-aware representations from metabolic network graphs.

2.9 Data Representations

To enable data-driven prediction of gene deletion strategies, appropriate data representations are required to encode biological information into formats that deep learning models can process. This dissertation employs three primary data representation schemes:

SMILES Notation for Metabolites.

SMILES (Simplified Molecular Input Line Entry System) is a chemical notation that encodes the structure of organic compounds as linear strings of characters [42]. For example, the metabolite glucose can be represented as C([C@@H]1[C@@H]([C@H]([C@@H]([C@H](O1)O)O)O)O)O. SMILES notation preserves essential structural information about molecular connectivity, functional groups, and stereochemistry, making it suitable for machine learning applications in cheminformatics and metabolomics.

Amino Acid Sequences for Genes. Genes encode information through their corresponding protein products, which are composed of sequences of amino acids.

Each amino acid is represented by a single-letter code (A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y). For example, a protein sequence might be represented as "MKTAYIAKQRQISFVKS". Amino acid sequences preserve functional and structural information about proteins, making them suitable for machine learning applications in protein function prediction and structural analysis [43, 44, 45]. In this dissertation, amino acid sequence data is obtained from the KEGG database [46].

Graph Representations for Metabolic Networks. Metabolic networks can be represented as graphs, where metabolites serve as nodes and biochemical reactions define edges connecting substrates to products. Graph representations naturally capture topological relationships, pathway modules, and metabolite interactions that are not captured in sequential data formats. In this dissertation, metabolic networks are constructed as metabolite-centric graphs with edges representing substrate-product relationships between metabolites. These graph representations enable the application of graph neural networks to learn structure-aware metabolite features that complement sequential representations.

Chapter 3

DBgDel

This chapter presents the first contribution of this dissertation: a database-enhanced framework for accelerating gene deletion strategy calculation by integrating prior knowledge from existing databases with downstream optimization algorithms. We first motivate the need for database integration, then present the DBgDel framework in detail, illustrate its application using the toy model from Chapter 2, and provide comprehensive experimental validation.

3.1 Introduction

As discussed in Chapter 1, gene deletion strategy calculation is computationally challenging, particularly for genome-scale metabolic models. Over the past two decades, numerous methods have been proposed to address this challenge, including OptKnock [9], OptStrain [10], OptForce [14], and MCS-based approaches [22]. While existing optimization-based methods such as gDel_minRN (Section 2.4) achieve high success rates, they perform de novo calculations for each target metabolite, overlooking potential commonalities across different metabolites. At the same time, databases such as MetNetComp (Section 2.7) have accumulated vast repositories of pre-computed gene deletion strategies—over 85,000 strategies across multiple genome-scale models from diverse microbial species.

The motivation for this work is straightforward: can we leverage existing strategies in databases to inform calculations for new target metabolites? The intuition is that certain **core metabolic components**—genes and reactions essential for growth-coupled production—are likely shared across multiple metabolites within the same metabolic model. If we can identify these core components from known strategies, we can constrain the search space for new calculations, potentially accelerating computational speed while maintaining solution quality.

However, extracting and integrating this information is non-trivial. Gene deletion strategies stored in databases may vary widely in the number of genes deleted, corresponding to different sub-networks of varying scales. Moreover, integrating extracted information into downstream optimization algorithms requires careful design to ensure that constraints derived from database patterns do not over-restrict the search space or sacrifice feasibility.

In this chapter, we introduce **DBgDel** (Database-enhanced Gene Deletion framework), a novel two-step approach that bridges database resources and algorithmic computation. The framework operates as follows: (1) **STEP 1** analyzes maximal gene deletion strategies from the MetNetComp database to extract the remaining gene set G_{remain} that represents core metabolic components, and (2) **STEP 2** employs an extended version of the gDel_minRN algorithm that incorporates G_{remain} as a constraint, protecting these genes from deletion and thereby narrowing the algorithmic search space.

The primary contributions of this work are:

- A systematic methodology for extracting and utilizing database information to inform gene deletion calculations, representing the first integration of pre-computed strategies into downstream algorithms.
- An extended gDel_minRN algorithm that incorporates database-derived constraints, achieving substantial computational acceleration while maintaining high success rates.
- Comprehensive empirical evaluation demonstrating that DBgDel achieves a favorable trade-off between computational efficiency and solution quality across three genome-scale metabolic models.

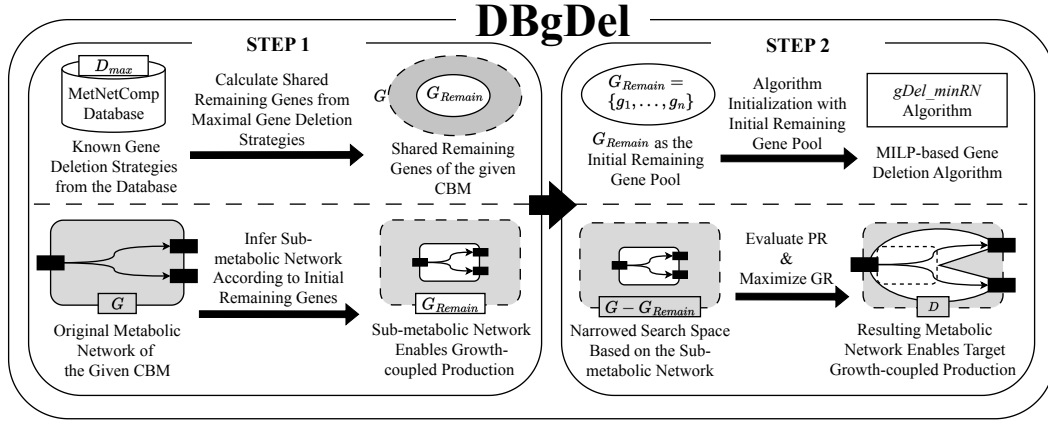


Figure 3.1: System overview of the DBgDel framework. The framework comprises two steps: (1) STEP 1, extracting the gene set G_{remain} from the MetNetComp database; and (2) STEP 2, using an extended gDel_minRN algorithm that incorporates G_{remain} to calculate gene deletion strategies for new target metabolites.

3.2 DBgDel Framework

The DBgDel framework operates in two sequential steps, as illustrated in Figure 3.1. STEP 1 extracts prior information from the MetNetComp database to construct the remaining gene set G_{remain} , representing core metabolic components essential for growth-coupled production. STEP 2 employs an extended version of the gDel_minRN algorithm that incorporates G_{remain} as a constraint, narrowing the search space and accelerating computation.

3.2.1 STEP 1: Extracting Remaining Gene Set

For a given constraint-based model, STEP 1 aims to extract the prior information about how the growth-coupled production of its metabolites is achieved.

The basic idea behind this aim is that for any given constraint-based model's metabolic network part C_1 , there should be some core components, denoted as C_1^{Core} , that are essential for achieving growth-coupled production for all of its metabolites (if possible).

The C_1^{Core} is given by all the metabolites in a constraint-based model and forms a sub-network of the full metabolic network C_1 . Therefore, based on information about how various metabolites achieve growth-coupled production through sub-networks, we can make a prediction of C_1^{Core} .

Such information can also be extracted from the gene deletion strategies, since each strategy gives the deleted genes D , according to which a sub-network of $\mathcal{C}_1$ can be derived.

However, multiple gene deletion strategies may exist for a target metabolite to achieve its growth-coupled production. These strategies may result in gene deletions D of varying sizes, leading to redundant sub-networks of $\mathcal{C}_1$ at different scales. To address this redundancy, we consider only the maximal gene deletion strategy $D_{\max}$ for each metabolite. This strategy maximizes the size of the deleted gene set D to achieve growth-coupled production, corresponding to the smallest sub-network of $\mathcal{C}_1$ necessary for this purpose.

To achieve the above-illustrated aim of extracting prior information, in STEP 1, we construct a remaining gene set G_{remain} based on the maximal gene deletion strategies $D_{\max}$ from the MetNetComp database.

Specifically, for a given constraint-based model, the G_{remain} is given by:

$$G_{\text{remain}} = G - (D_{\max}^1 \cup D_{\max}^2 \cup \dots \cup D_{\max}^{|M|}) \quad (3.1)$$

where G is a set of all genes, and $|M|$ is the number of available maximal gene deletion strategies $D_{\max}$ for different metabolites in the MetNetComp database.

For a constraint-based model, when maximal gene deletion strategies are available for all its metabolites in the MetNetComp database, the resulting G_{remain} exactly corresponds to $\mathcal{C}_1^{\text{Core}}$ of this model.

However, in most cases, when dealing with a new target metabolite without known gene deletion strategies for it, G_{remain} is constructed based on information about the other metabolites. Therefore, as the default setting of STEP 1, we only track the maximal gene deletion strategies for a subset of the metabolites in the model, excluding the target itself. When predicting gene deletions for a new target metabolite based on information from other metabolites, we refer to the genes present in the resulting G_{remain} as Predicted- G_{remain} genes.

3.2.2 STEP 2: Extended gDel_minRN Algorithm

STEP 2 employs an extended version of the gDel algorithm to calculate the gene set D to be deleted. The gDel is a mixed-integer linear programming (MILP)-based algorithm, which aims to identify growth-coupling gene deletion strategies that maximize the number of repressed reactions.

The MILP algorithm from Line 14 of the pseudocode searches for a candidate of the gene deletion strategy $D_{\text{candidate}}$ that satisfies the following criteria:

1. Ensuring that the GR and PR are above the given thresholds $GR_{\text{threshold}}$ and $PR_{\text{threshold}}$, respectively.
2. Maximizing the number of reactions repressed by gene deletions.
3. Maximizing GR.

Note that criterion (2) takes precedence over (3). Then, starting from Line 26, $D_{\text{candidate}}$ is verified whether it achieves growth-coupled production of the target metabolite. Here we incorporate the gDel with the G_{remain} from STEP 1 as the initial gene pool to narrow down the algorithmic search space. As a result, the extended gDel calculates gene deletion strategies without traversing the cases deleting genes in G_{remain} , and obtain the minimum reaction network for growth-coupled production.

The pseudo-code for the complete DBgDel framework is presented as Algorithm 3 and Algorithm 4.

Algorithm 3 DBgDel - STEP 1: Construct G_{remain}

Input: $\mathcal{C}$, MetNetComp database

Output: G_{remain}

```

1:  $G_{\text{remain}} \leftarrow G$  /*Initialize  $G_{\text{remain}}$  as a copy of  $G$ */
2:  $D_{\text{max\_union}} \leftarrow \emptyset$  /*Initialize an empty set for the union of maximal gene deletion
   strategies*/
3: /*Take  $D_{\text{max}}^i$  for the  $i$ -th metabolite from database*/
4: for  $i = 1$  to  $|M|$  do
5:    $D_{\text{max\_union}} \leftarrow D_{\text{max\_union}} \cup D_{\text{max}}^i$ 
6: end for
7:  $G_{\text{remain}} \leftarrow G_{\text{remain}} - D_{\text{max\_union}}$ 
8: return  $G_{\text{remain}}$ 

```

3.3 Toy Model Illustration

We apply the DBgDel framework to Toy Model 1 from Section 2.5.1 with a new target metabolite to illustrate its execution.

Algorithm 4 DBgDel - STEP 2: Extended gDel_minRN Algorithm**Input:** $\mathcal{C}$, v_{target} , G_{remain} , maxloop , α , β **Output:** $D_{\text{candidate}}$, v_{target} , v_{growth}

```

1: /* Calculate theoretical maximum PR and GR*/
2:  $PR_{\text{theormax}} = \mathbf{max} \ v_{\text{target}}$  /* theoretical maximum PR */
3:   such that  $\sum_j S_{ij}v_j = 0$  for all  $1 \leq i \leq a$ 
4:    $LB_j \leq v_j \leq UB_j$  for all  $1 \leq j \leq b$ 
5:  $PR_{\text{threshold}} = \alpha \cdot PR_{\text{theormax}}$ 
6:  $GR_{\text{theormax}} = \mathbf{max} \ v_{\text{growth}}$  /* theoretical maximum GR */
7:   such that  $\sum_j S_{ij}v_j = 0$  for all  $1 \leq i \leq a$ 
8:    $LB_j \leq v_j \leq UB_j$  for all  $1 \leq j \leq b$ 
9:  $GR_{\text{threshold}} = \beta \cdot GR_{\text{theormax}}$ 
10: /* Finding a gene deletion strategy candidate.*/
11:  $\text{prohibited\_list} = \emptyset$ ,  $\text{loop} = 1$ 
12: while  $\text{loop} \leq \text{maxloop}$  do
13:   /* maximize the number of repressed reactions first, GR second */
14:    $\mathbf{max} \ GR_{\text{theormax}} \cdot KO + v_{\text{growth}}$ 
15:   /*KO: the number of repressed reactions.*/
16:   such that  $\sum_j S_{ij}v_j = 0$  for all  $1 \leq i \leq a$ 
17:   
$$\begin{cases} v_j = 0 & \text{if } p_j = 0, \\ l_j \leq v_j \leq u_j, & \text{otherwise;} \end{cases}$$

18:    $p_j = f_j(G)$ 
19:   /* linear constraints for Boolean functions. */
20:   
$$g = \begin{cases} 0, & \text{if } g \in D, \\ 1, & \text{if } g \in G_{\text{remain}}, \\ 1, & \text{otherwise;} \end{cases}$$

21:   /*D is the set of deleted genes,  $G_{\text{remain}}$  as the initial remaining gene pool.*/
22:    $D \cap G_{\text{remain}} = \emptyset$ 
23:    $D \cap \text{prohibited\_list} = \emptyset$ 
24:    $GR_{\text{threshold}} \leq v_{\text{growth}}$  and  $PR_{\text{threshold}} \leq v_{\text{target}}$ 
25:    $D_{\text{candidate}} = D$ 
26:    $\mathbf{min} \ v_{\text{target}}$  /*Check whether growth-coupled production is achieved.*/
27:   such that  $\mathbf{max} \ v_{\text{growth}}$ 
28:   such that
29:      $\sum_j S_{ij}v_j = 0$  for all  $i$ 
30:     
$$\begin{cases} v_j = 0 & \text{if } p_j = 0, \\ l_j \leq v_j \leq u_j, & \text{otherwise;} \end{cases}$$

31:      $p_j = f_j(G)$ 
32:     
$$g = \begin{cases} 0 & \text{if } g \in D_{\text{candidate}}, \\ 1, & \text{otherwise;} \end{cases}$$

33:   if  $v_{\text{target}} \geq PR_{\text{threshold}}$  and  $v_{\text{growth}} \geq GR_{\text{threshold}}$  then
34:     return  $D_{\text{candidate}}$ ,  $v_{\text{target}}$ ,  $v_{\text{growth}}$ 
35:   else
36:      $\text{prohibited\_list} = \text{prohibited\_list} \cup D_{\text{candidate}}$ 
37:      $\text{loop} = \text{loop} + 1$ 
38:   end if
39: end while

```

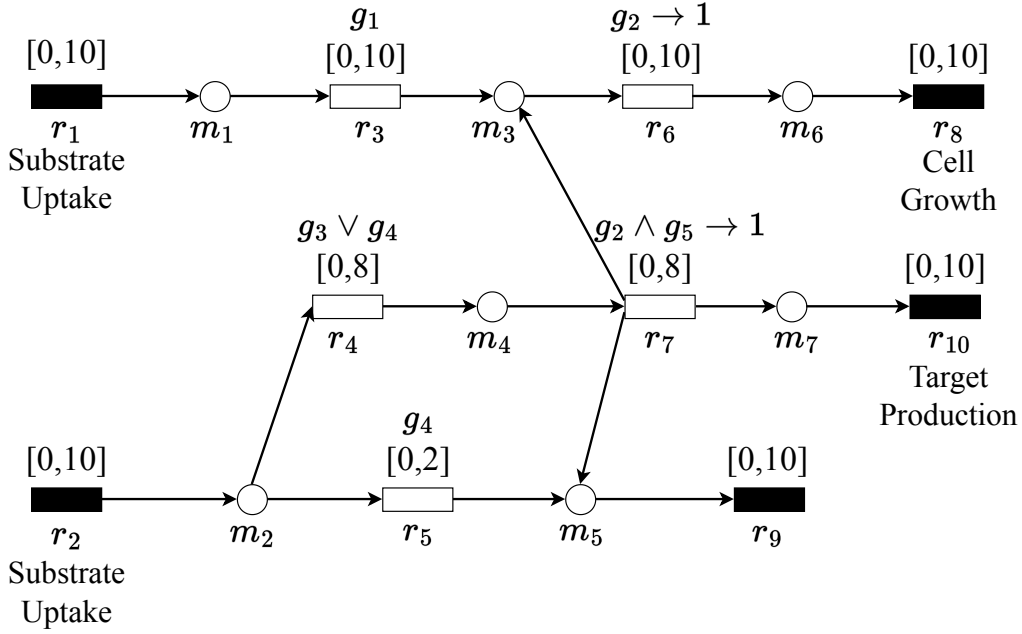


Figure 3.2: The toy model with a new target metabolite. Circles and rectangles represent metabolites and reactions, respectively. Black rectangles denote external and internal reactions. r_1, r_2 correspond to two substrate uptake reactions. r_8, r_{10} correspond to cell growth, and target metabolite production reactions, respectively. The reaction rates are constrained by the range $[l_i, u_i]$.

3.3.1 Toy Model Settings

As shown in Fig. 3.2, we set m_7 as the new target metabolite. In the original model, m_7 does not have an external reaction, therefore it is necessary to add an auxiliary reaction r_{10} to simulate its secretion. The other components of $\mathcal{C}_1$ and $\mathcal{C}_2$ part of the model are the same as the original one described in Section 2.5.1.

3.3.2 Behaviors of the DBgDel Framework

In STEP 1, the DBgDel framework constructs a gene set G_{remain} for the initial remaining gene pool. From the analysis in Section 2.5.1, we know that three gene deletion strategies enable growth-coupled production of the target metabolite m_5 : $\{g_1\}$, $\{g_1, g_3\}$, and $\{g_1, g_4\}$. These strategies are used as inputs of STEP 1 to generate their corresponding remaining genes. As shown in Table 3.1, the remaining genes from these strategies all include g_2 and g_5 . Consequently, we have Predicted- $G_{\text{remain}} = \{g_2, g_5\}$ for the target metabolite m_7 .

Table 3.1: The remaining genes correspond to the gene deletion strategies that enable growth-coupled production of the target metabolite in the toy example.

Gene deletion strategies	Corresponding remaining genes
$\{g_1\}$	$\{g_2, g_3, g_4, g_5\}$
$\{g_1, g_3\}$	$\{g_2, g_4, g_5\}$
$\{g_1, g_4\}$	$\{g_2, g_3, g_5\}$

Table 3.2: Gene deletion strategies for the toy example in Fig. 3.2 are classified into four types according to the resulting flux distributions (reaction rates).

Type	Gene deletion strategies classified by flux distributions
1	$\emptyset, \{g_3\}, \{g_4\}$
2	$\{g_1\}, \{g_1, g_3\}, \{g_1, g_4\}$
3	$\{g_3, g_4\}$
4	$\{g_1, g_3, g_4\}$

For comparison, we also constructed the toy model’s growth-essential gene set, denoted as G_{GE} , by examining its cell growth reaction situation. Specifically, since deleting g_2 always leads to $GR = 0$, in this example, we have $G_{GE} = \{g_2\}$.

Next in STEP 2, we set the initial remaining gene pool as Predicted- G_{remain} . The GPR rules related to the genes in G_{remain} are modified. Specifically, for g_2 and g_5 , we modify f_6 into $f_6 : p_6 = g_2 = 1$, and f_7 into $f_7 : p_7 = g_2 \wedge g_5 = 1$. According to the modified GPR rules, genes g_2 and g_5 will never be deleted, the gene deletion algorithm gDel in STEP 2 doesn’t need to compute their related gene deletions, leaving only three genes (g_1, g_3 and g_4) in the searching space. Therefore, only $2^3 = 8$ patterns of gene deletions are left in this example, which are classified into four cases according to the flux distributions, as described in Table 3.2. Similarly, in the case of using G_{GE} as the initial remaining gene pool ($G_{\text{remain}} = G_{GE}$), the gDel algorithm needs to compute $2^4 = 16$ patterns of gene deletions.

In the original state of the toy model, where no genes are deleted, when we maximize GR, we can obtain its maximum value $\max(GR) = \max(v_8) = 10$. However, there are two paths to reach from the substrate uptake reactions r_1 and r_2 to the growth reaction r_8 : $(r_1 \rightarrow r_3 \rightarrow r_6 \rightarrow r_8)$ and $(r_2 \rightarrow r_4 \rightarrow r_7 \rightarrow r_6 \rightarrow r_8)$. If both two paths mentioned above are used, and the second one achieves maximum fluxes, $GR = 10$ and $PR = 8$ can be obtained, as shown in ID 1 in Table 3.3. This is the most optimistic case regarding the value of PR. However, if only the first path is used, $GR = 10$ and $PR = 0$ can be obtained, as shown in ID 2 of Table 3.3. This

Table 3.3: The resulting flux distribution of Types 1 to 4 gene deletion strategies. For each gene deletion strategy type, the most optimistic (best) and pessimistic (worst) PR (v_{10}) at GR (v_8) maximization are shown.

ID	Gene deletion strategy	PR situation	Flux distribution									
			v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}
1	Type 1	best	2	8	2	8	0	10	8	10	8	8
2		worst	10	0	10	0	0	10	0	10	0	0
3	Type 2	best	0	8	0	8	0	8	8	8	8	8
4		worst	0	8	0	8	0	8	8	8	8	8
5	Type 3	best	10	0	10	0	0	10	0	10	0	0
6		worst	10	0	10	0	0	10	0	10	0	0
7	Type 4	best	0	0	0	0	0	0	0	0	0	0
8		worst	0	0	0	0	0	0	0	0	0	0

is the most pessimistic case regarding the value of PR. Therefore, in the original state of the toy model, we have $GR = 10$ and $PR = 0$. Gene deletion strategies $\{g_3\}$ and $\{g_4\}$ are also classified as Type 1 because the same flux distribution is obtained.

IDs 3 and 4 of Table 3.3 describe the most optimistic and pessimistic flux distributions regarding the PR value when g_1 is deleted. When g_1 is deleted, $f_3 : p_3 = g_1 = 0$ is obtained and v_3 is forced to be zero. The maximum GR is $v_8 = 8$ since only the second path (from r_2 to r_8) that reaches cell growth reaction can be used. The flux distribution is uniquely determined, $v_1 = v_5 = 0$, $v_2 = v_4 = v_7 = v_{10} = 8$ are obtained, and PR is 8 for both optimistic and pessimistic cases regarding the PR value. The gene deletion strategies $\{g_1, g_3\}$ and $\{g_1, g_4\}$ are also classified as Type 2 as shown in Table 3.3.

IDs 5 and 6 of Table 3.3 describe the flux distributions when $\{g_3, g_4\}$ are deleted. When $\{g_3, g_4\}$ are deleted, $f_4 : p_4 = g_3 \vee g_4 = 0$ and $f_5 : p_5 = g_4 = 0$ are obtained, both v_4 and v_5 are forced to be zero. The maximum GR is $v_8 = 10$ since only the first path (from r_1 to r_8) that reaches cell growth reaction can be used. The flux distribution is uniquely determined, $v_1 = v_3 = v_6 = 10$, $v_2 = v_7 = v_9 = 0$ are obtained, and PR is 0 for both optimistic and pessimistic cases regarding the PR value.

IDs 7 and 8 of Table 3.3 describe the flux distributions when $\{g_1, g_3, g_4\}$ are deleted. When $\{g_1, g_3, g_4\}$ are deleted, $f_3 : p_3 = g_1 = 0$, $f_4 : p_4 = g_3 \vee g_4 = 0$, and

Table 3.4: The constraint-based models that were used in the computational experiments.

Model	e_coli_core	iMM904	iML1515
#Genes	137	905	1516
#Growth essential genes	7	110	196
# G_{remain} genes	11	178	254
#Predicted- G_{remain} genes	23	195	267
#Randomly chosen genes	23	195	267
#Reactions	95	1577	2712
#Metabolites	72	1226	1877
#Target metabolites	48	782	1085
#Target metabolites (extracellular space)	8	82	94
#Target metabolites (non-extracellular space)	40	700	991

$f_5 : p_5 = g_4 = 0$ are obtained, v_3 , v_4 and v_5 are forced to be zero. The maximum GR is $v_8 = 0$ since none of the two paths (from r_1 or r_2 to r_8) that reach cell growth reaction can be used. The flux distribution is uniquely determined where all reaction rates become zero, therefore PR is 0 for both optimistic and pessimistic cases regarding the PR value.

Through all the cases above, we find that for gene deletion strategies of Type 2, even in the most pessimistic case, we still meet the criteria of $PR \geq PR_{\text{threshold}}$ and $GR \geq GR_{\text{threshold}}$. Consequently, in our example of the toy model, this type of deletions enable growth-coupled production of the target metabolite m_7 .

3.4 Computational Experiments

This section presents comprehensive computational experiments to validate the effectiveness of DBgDel. We first describe the experimental setup, including the metabolic models, baseline methods, and evaluation metrics. We then present detailed results comparing DBgDel with existing methods and analyze the impact of different initial gene pool settings.

3.4.1 Experimental Setup

In the computational experiments, we applied the DBgDel framework to three metabolic models *e_coli_core* [47], *iMM904* [48], and *iML1515* [49]. The number of reactions, genes, and metabolites for the three metabolic models are summarized in Table 3.4.

We define four types of genes used in the experiments, i.e., Growth Essential (GE) genes, G_{remain} genes, Predicted- G_{remain} genes, and Randomly Chosen (RC) genes as below:

- Growth Essential (GE) genes: A gene is defined as a growth essential gene when its deletion results in a maximum GR of zero in FBA analysis.
- G_{remain} genes: For a given metabolic model, we track the maximal gene deletion strategies for all its metabolites archived in the MetNetComp database, as outlined in Eq. 3.1 in STEP 1. Specifically, genes present in the resulting G_{remain} are remained in the maximal gene deletion strategies for all the metabolites. We defined them as G_{remain} genes.
- Predicted- G_{remain} genes: For a given metabolic model, we track only the maximal gene deletion strategies for the metabolites that exist in the extracellular space (metabolite whose name ends with '_e') archived in the MetNetComp database. The calculation is outlined in Eq. 3.1 in STEP 1. In this case, the resulting gene set is calculated based only on information about the extracellular space metabolites. We defined them as Predicted- G_{remain} gene.
- Randomly Chosen (RC) genes: A set of genes randomly chosen from the given metabolic model. RC genes have no biological significance and are used to ensure a fair comparison of effectiveness in narrowing the search space. The number of RC genes equals that of Predicted- G_{remain} genes.

We design the computational experiments as follows:

1. Evaluate the overall computational efficiency of different gene deletion calculation methods. We compared the performance of DBgDel with other baseline methods, including GDLS, optGene, gDel_minRN, and gMCSE.
2. Evaluate the efficacy of different initial gene pool settings of DBgDel. We compared the performance of DBgDel with different initial gene pools in STEP2: Predicted- G_{remain} genes, GE genes, G_{remain} genes, and RC genes. Here, Predicted- G_{remain} is the default choice of the initial gene pool, and G_{remain} serves as the performance benchmark.

Note that some metabolites cannot be produced through simulation-based growth-coupled production. We determined and excluded such metabolites from the target metabolites by calculating their theoretical maximum PR. When a target

metabolite does not have an external (exchange) reaction, an auxiliary exchange reaction was temporarily added to the model to simulate the secretion. The unit of every reaction rate is mmol/gDW/h, which will be omitted hereafter for simplicity of notation. Throughout this chapter, we set $GR_{\text{threshold}} = PR_{\text{threshold}} = 0.001$, ensuring both non-trivial growth and production while avoiding numerical artifacts in optimization.

All procedures in the computational experiments were implemented on a Ubuntu 20.04.6 LTS machine with an Intel Xeon Gold Processor with 2.30 GHz 64 cores/128 threads, 128 GB memory, and 1TB SSD. This workstation had CPLEX 12.10, COBRA Toolbox v3.0, CellNetAnalyzer ver.2023.1, and MATLAB R2019b installed and used for these analyses.

Baseline Methods

We compared DBgDel with four existing methods:

- **GDLS** [50]: A genetic algorithm-based method available in COBRA Toolbox [51].
- **optGene** [52]: A genetic algorithm-based strain design method available in COBRA Toolbox [51].
- **gDel_minRN** [23]: A MILP-based algorithm that maximizes repressed reactions.
- **gMCSE** [22]: A minimal cut set-based method available as API in CellNetAnalyzer [53]; we evaluated its first resulting MCS.

Evaluation Metrics

We evaluated each method using:

- **Success rate**: Proportion of target metabolites for which a gene deletion strategy achieves $GR \geq 0.001$ and $PR \geq 0.001$ in the most pessimistic flux distribution scenario.
- **Computational time**: Average elapsed time in seconds across all target metabolites.
- **Production rate percentile ranks**: Distribution of PRs compared to MetNet-Comp baselines, calculated as percentile ranks.

3.4.2 Results and Analysis

Performance Comparison with Baselines

For *e_coli_core*: Table 3.5 summarizes the performance comparison between the proposed method DBgDel (using Predicted- G_{remain} genes as the initial remaining gene pool) with GDLS, optGene, gDel_minRN, and gMCSE in the context of targeting 40 non-extracellular space metabolites of *e_coli_core*. Regarding the success rate, DBgDel achieved success in 24 out of 40 target metabolites, meeting the minimum GR and PR threshold of 0.001 or higher during GR maximization. In comparison, GDLS succeeded in only 5 out of 40 cases, optGene in 22 out of 40 cases, gDel_minRN in 28 out of 40 cases, and gMCSE in 7 out of 40 cases. Regarding computational efficiency, optGene required significantly more extended time, with an average of 1779.30 seconds, while the other methods cost about 1 second.

For *iMM904*: Table 3.5 summarizes the performance comparison between the proposed method DBgDel (using Predicted- G_{remain} genes as the initial remaining gene pool) with GDLS, optGene, gDel_minRN, and gMCSE in the context of targeting 700 non-extracellular space metabolites of *iMM904*. Regarding the success rate, DBgDel achieved success in 81 out of 700 target metabolites, meeting the minimum GR and PR threshold of 0.001 or higher during GR maximization. In comparison, GDLS failed in all 700 cases, optGene succeeded in only 27 out of 700 cases, gDel_minRN in 116 out of 700 cases, and gMCSE in 17 out of 700 cases. DBgDel demonstrated outstanding performance in average elapsed time, with a mean time of 79.92 seconds. GDLS had an average time of 243.42 seconds, optGene required significantly more extended time with an average of 1986.45 seconds, gDel_minRN had an average time of 422.32 seconds, and gMCSE had an average time of 584.24 seconds.

For *iML1515*: Table 3.5 summarizes the performance comparison between the proposed method DBgDel (using Predicted- G_{remain} genes as the initial remaining gene pool) with GDLS, optGene, gDel_minRN, and gMCSE in the context of targeting 991 non-extracellular space metabolites of *iML1515*. Regarding the success rate, DBgDel achieved success in 507 out of 991 target metabolites, meeting the minimum GR and PR threshold of 0.001 or higher during GR maximization. GDLS, gMCSE, and optGene failed in all 991 cases, and gDel_minRN succeeded in 508 out of 991 cases. DBgDel demonstrated outstanding performance with an average elapsed time of 431.75 seconds. GDLS had an average time of 244.58 seconds,

Table 3.5: Performance comparison between the proposed method DBgDel (using Predicted- G_{remain} genes as the initial remaining gene pool) with GDLS, optGene, gDel_minRN, and gMCSE. Each gene deletion strategy was deemed successful if it achieved a minimum GR and PR of 0.001 or higher during GR maximization. The unit of elapsed time is second (s).

Model	Method	Success rate	Average time (s)		
			All	Success=1	Success=0
<i>e_coli_core</i>	DBgDel	24/40	1.12	0.59	1.91
	GDLS	5/40	1.24	1.24	1.24
	gMCSE	7/40	1.73	1.73	1.73
	optGene	22/40	1779.30	1779.30	1779.30
	gDel_minRN	28/40	1.62	0.82	3.22
<i>iMM904</i>	DBgDel	81/700	79.92	22.89	87.38
	GDLS	0/700	243.42	–	243.42
	gMCSE	17/700	584.24	584.24	584.24
	optGene	27/700	1986.45	1986.45	1986.45
	gDel_minRN	116/700	422.32	189.90	459.31
<i>iML1515</i>	DBgDel	507/991	431.75	92.46	787.17
	GDLS	0/991	244.58	–	244.58
	gMCSE	0/991	1892.39	–	1892.39
	optGene	0/991	2193.45	–	2193.45
	gDel_minRN	508/991	2541.23	481.66	4561.56

optGene required significantly more extended time with an average of 2193.45 seconds, gDel_minRN had an average time of 2541.23 seconds, and gMCSE had an average time of 1892.39 seconds.

Impact of Initial Gene Pool Settings

Table 3.6 summarizes the performance comparison between the proposed method DBgDel with different initial remaining gene pool settings for *e_coli_core*. When utilizing Predicted- G_{remain} genes as the initial remaining gene pool, the success rate in achieving the minimum GR and PR threshold of 0.001 or higher during GR maximization was 24 out of 40 instances. Similarly, when using the GE genes as the initial remaining gene pool, the success rate was 28 out of 40 instances. When employing G_{remain} genes as the initial remaining gene pool, the success rate remained at 28 out of 40 instances. Additionally, when employing RC genes as the initial remaining gene pool, the success rate decreased to 21 out of 40 instances.

As for computational efficiency, since the model scale of *e_coli_core* is relatively small, all four settings of the initial remaining gene pool can achieve a computation time of around 1 second, which is short enough.

Table 3.6 summarizes the performance comparison between the proposed method DBgDel with different initial remaining gene pool settings for *iMM904*. The search space of gDel_minRN is larger than that of DBgDel, but the success rates of GE genes and G_{remain} are higher. This is because GE genes and G_{remain} effectively utilize information about the genes that must not be deleted, which gDel_minRN does not refer to. When utilizing Predicted- G_{remain} genes as the initial remaining gene pool, the success rate in achieving the minimum GR and PR threshold of 0.001 or higher during GR maximization was 81 out of 700 instances. When using the GE genes as the initial remaining gene pool, the success rate was 120 out of 700 instances. When employing G_{remain} genes as the initial remaining gene pool, the success rate remained at 126 out of 700 instances. Additionally, when employing RC genes as the initial remaining gene pool, the success rate decreased to 65 out of 700 instances. Examining the computational efficiency, the proposed method demonstrated varied performance regarding elapsed time for successful cases. For instances where success was achieved, the average elapsed time was 22.89 seconds with Predicted- G_{remain} genes, 189.90 seconds with GE genes, 47.82 seconds with G_{remain} genes, and 31.93 seconds with RC genes. In contrast, for cases where success was not achieved, the average elapsed time was 87.38 seconds with Predicted- G_{remain} genes, 459.31 seconds with GE genes, 224.19 seconds with G_{remain} genes, and 157.42 seconds with RC genes. Considering all cases, regardless of success or failure, the proposed method showed outstanding average elapsed times when using Predicted- G_{remain} genes, with values of 79.92 seconds.

Table 3.6 summarizes the performance comparison between the proposed method DBgDel with different initial remaining gene pool settings for *iML1515*. When utilizing Predicted- G_{remain} genes as the initial remaining gene pool, the success rate in achieving the minimum GR and PR threshold of 0.001 or higher during GR maximization was 507 out of 991 instances. When using the GE genes as the initial remaining gene pool, the success rate was 508 out of 991 instances. When employing G_{remain} genes as the initial remaining gene pool, the success rate remained at 508 out of 991 instances. Additionally, when employing RC genes as the initial remaining gene pool, the success rate decreased to 178 out of 991 instances. Examining the computational efficiency, the proposed method

Table 3.6: Performance comparison of DBgDel with different initial remaining gene pool settings. Each gene deletion strategy was deemed successful if it achieved a minimum GR and PR of 0.001 or higher during GR maximization. The unit of elapsed time is second (s).

Model	Initial gene pool	Success rate	Average time (s)		
			All	Success=1	Success=0
<i>e_coli_core</i>	Predicted- G_{remain}	24/40	1.12	0.59	1.91
	GE	28/40	1.33	0.68	2.83
	G_{remain}	28/40	1.12	0.60	2.57
	RC	21/40	1.17	0.59	2.34
<i>iMM904</i>	Predicted- G_{remain}	81/700	79.92	22.89	87.38
	GE	120/700	413.13	189.90	459.31
	G_{remain}	126/700	192.44	47.82	224.19
	RC	65/700	145.77	31.93	157.42
<i>iML1515</i>	Predicted- G_{remain}	507/991	431.75	92.46	787.17
	GE	508/991	2470.15	481.66	4561.56
	G_{remain}	508/991	463.70	90.30	858.01
	RC	178/991	713.74	80.26	852.43

demonstrated varied performance regarding elapsed time for successful cases. For instances where success was achieved, the average elapsed time was 92.46 seconds with Predicted- G_{remain} genes, 481.66 seconds with GE genes, 90.30 seconds with G_{remain} genes, and 80.26 seconds with RC genes. In contrast, for cases where success was not achieved, the average elapsed time was 787.17 seconds with Predicted- G_{remain} genes, 4561.56 seconds with GE genes, 858.01 seconds with G_{remain} genes, and 852.43 seconds with RC genes. Considering all cases, regardless of success or failure, the proposed method showed outstanding average elapsed times when using Predicted- G_{remain} or G_{remain} genes, with values of 431.75 and 463.70 seconds.

Comparison of PR Percentile Ranks

To further analyze the trade-offs in DBgDel and discuss the limitations of its resulting gene deletion solutions, we compare the PRs obtained by DBgDel with the PRs listed in MetNetComp for target metabolites in three models under study. The DBgDel PRs were derived from computational simulations of metabolic models using DBgDel gene deletion strategies. The MetNetComp PRs were obtained from

Table 3.7: Distribution of percentile ranks for target metabolites PRs across three metabolic models using DBgDel, with MetNetComp PRs as baselines.

Percentile range	e_coli_core	iMM904	iML1515
0%	8.82%	25.56%	0.91%
0-30%	38.23%	6.67%	1.21%
30-60%	0.00%	4.44%	31.72%
60-90%	0.00%	1.11%	0.91%
>90%	52.94%	62.22%	65.26%

the MetNetComp database as a baseline for comparison.

Table 3.7 summarizes the resulting distribution of percentile ranks for target metabolites PRs across three metabolic models using DBgDel. Percentile ranks were calculated using MetNetComp PRs as baselines. A percentile of 0% indicates a zero production rate and, consequently, a failure to achieve growth-coupled production. In the e_coli_core model, the largest proportion of target metabolites (52.94%) fell within the >90% percentile range, followed by 38.23% in the 0-30% range. Smaller proportions were observed in the 0% (8.82%) and the 30-60% and 60-90% ranges (both 0%). For the iMM904 model, the largest proportion of target metabolites (62.22%) were in the >90% range, followed by 25.56% in the 0% range. Smaller proportions were observed in the 0-30% (6.67%), 30-60% (4.44%), and 60-90% (1.11%) ranges. In the iML1515 model, the >90% range had the highest proportion of metabolites (65.26%), followed by 31.72% in the 30-60% range. The 0% range accounted for 0.91%, and the 0-30% and 60-90% ranges accounted for 1.21% and 0.91%, respectively.

3.5 Discussion

This chapter introduced DBgDel, a framework that bridges database knowledge with algorithmic computation to accelerate gene deletion strategy calculation. The key insight is that genome-scale metabolic models exhibit core metabolic components—genes consistently retained across successful strategies—that can be systematically extracted from databases and used to constrain downstream optimization. By protecting these core components from deletion, DBgDel substantially narrows the algorithmic search space while maintaining competitive solution quality.

Experimental results demonstrate that DBgDel achieves an average 6.1-fold acceleration across three genome-scale metabolic models compared to existing methods, with speedups of 1.45-fold, 2.4-fold, and 5.89-fold over the previously best-performing method gDel_minRN for *e_coli_core*, *IMM904*, and *iML1515*, respectively. Critically, this acceleration does not come at the expense of solution quality: success rate gaps compared to gDel_minRN are only 10%, 5%, and 0.1% across the three models, demonstrating a favorable efficiency-quality trade-off. The production rate percentile rank analysis reveals that DBgDel strategies achieve competitive local solutions, with a significant portion of target metabolites reaching >90% percentile production rates when compared to MetNetComp baselines. This indicates that the database-derived constraints effectively guide the optimization toward high-quality regions of the solution space rather than simply restricting it arbitrarily. This also implies that even compared to global optimization methods like RobustKnock [54], which cannot effectively compute large-scale gene deletions for genome-scale metabolic models, the iterative approach of DBgDel still delivers competitive local solutions in terms of resulting production rates.

The comparative analysis between different initial gene pool settings provides important validation of the framework’s core mechanism. The substantial performance degradation when using random candidate genes—with success rate differences of 7%, 9%, and 33% across the three models—demonstrates that STEP 1’s database extraction effectively captures meaningful biological constraints rather than simply reducing search space arbitrarily. Moreover, the observation that this effect becomes more pronounced as model scale increases confirms that database knowledge becomes increasingly valuable for large-scale metabolic networks where exhaustive search is prohibitive. The framework’s focus on maximal gene deletion strategies, which reveal core metabolic flows essential for growth-coupled production, provides both computational efficiency and biological interpretability. Moreover, based on the resulting maximal gene deletion strategies, it is also possible to derive many other gene deletion strategies with fewer gene deletions [55].

However, DBgDel’s achievements must be understood within the constraints of its design philosophy. The framework remains fundamentally dependent on downstream optimization algorithms, requiring iterative MILP solving for each new target metabolite. The extraction rules in STEP 1, while effective, are hand-crafted and rely on the availability of maximal strategies in the database, potentially limiting applicability to emerging organisms or novel metabolic contexts. Furthermore,

DBgDel targets acceptable local solutions rather than global optima, prioritizing rapid identification of viable strategies over exhaustive optimization.

While DBgDel successfully demonstrates that database integration can substantially accelerate gene deletion calculation, it does not fundamentally escape the calculation paradigm: each new target metabolite still requires explicit optimization with full network modeling and iterative algorithm execution. This raises a more ambitious question: rather than accelerating calculation, can we reformulate gene deletion strategy design as a prediction problem that learns patterns from existing strategies and generalizes to new metabolites without requiring explicit optimization at all? This paradigm shift from calculation to prediction forms the focus of Chapter 4.

Chapter 4

DeepGDel

This chapter presents the second contribution of this dissertation: a deep learning framework for predicting gene deletion strategies that transitions from optimization-based calculation to data-driven prediction. We first motivate the paradigm shift from calculation to prediction, then formulate the gene deletion strategy prediction problem, present the DeepGDel framework in detail, illustrate its application using the toy model, and provide comprehensive experimental validation.

4.1 Introduction

DBgDel [56] (Chapter 3) successfully demonstrated that integrating pre-computed database knowledge with optimization algorithms can substantially accelerate gene deletion strategy calculation while maintaining competitive success rates. However, this approach still relies on hand-crafted extraction rules to identify potentially deletable genes in STEP 1 and remains fundamentally dependent on iterative optimization algorithms in STEP 2 to explore the search space. More broadly, all existing methods in the gene deletion strategy literature employ a de novo calculation paradigm that requires explicitly modeling every biochemical interaction within the metabolic network—every stoichiometric constraint, every flux bound, every

Boolean gene-protein-reaction rule—performing complex mathematical optimization for each target metabolite. This comprehensive modeling approach, while theoretically sound, creates significant scalability bottlenecks for large genome-scale metabolic models and limits efficiency for rapid screening applications.

In this chapter, we propose a fundamentally different paradigm by reformulating gene deletion strategy design as a supervised machine learning prediction problem rather than an optimization-based calculation problem. The key insight is that instead of explicitly modeling all metabolic interactions and performing iterative search, we can learn patterns and relationships from existing successful gene deletion strategies and predict new strategies based on these learned patterns. This paradigm shift becomes feasible when we recognize that metabolites and genes possess intrinsic characteristics encoded in their sequential representations—SMILES chemical notation for metabolites captures molecular structure and properties, while amino acid sequences for genes encode protein function and constraints—and that deep learning architectures excel at extracting meaningful patterns from sequential data. By leveraging these representational foundations and the availability of labeled data from databases like MetNetComp, we can train models to predict gene deletion strategies without requiring explicit network modeling or manual rule engineering.

However, transforming this conceptual shift into a working framework presents several technical challenges. First, we must develop effective numerical representations for variable-length sequences: SMILES strings and amino acid sequences vary considerably in length, requiring robust encoding schemes that capture biological properties relevant to deletion decisions. Second, we must model complex pairwise relationships between genes and metabolites—predicting the deletion status (delete or retain) for each gene-metabolite combination—within a framework that handles severe class imbalance, as most genes are typically retained rather than deleted for any given target metabolite. Third, we must validate that learned patterns generalize to unseen metabolites through rigorous cross-validation designs that assess both predictive accuracy and biological interpretability.

We introduce DeepGDeI (Deep learning-based Gene Deletion framework), a data-driven prediction framework that addresses these challenges through a modular neural network architecture. DeepGDeI comprises three coordinated modules: Meta-M learns latent representations of metabolites from SMILES sequences using an LSTM-based autoencoder; Gene-M learns latent representations of genes from

amino acid sequences using an identical architecture; and Pred-M integrates these complementary representations in a pairwise manner to predict deletion status for each gene-metabolite pair. The framework employs a two-stage training regime: unsupervised representation learning in the autoencoder modules leverages large amounts of unlabeled sequence data to capture general structural and functional patterns, followed by supervised fine-tuning where the Pred-M module learns task-specific deletion prediction using labeled gene-metabolite pairs. This hybrid training approach maximizes the model’s ability to leverage sequence information while focusing on the specific prediction objective.

To the best of our knowledge, DeepGDel represents the first systematic formulation of gene deletion strategy design as a supervised machine learning prediction task, marking a fundamental paradigm shift from optimization-based calculation to data-driven prediction. Experimental validation across three metabolic models demonstrates substantial performance improvements over both proportion-based baselines and naive neural network approaches, achieving accuracy gains of 14.69%, 22.52%, and 13.03% on *e_coli_core*, *iMM904*, and *iML1515*, respectively, while enabling millisecond-scale prediction compared to hours required by optimization-based methods. Ablation studies confirm that both metabolite and gene sequence representations provide complementary information essential for accurate prediction.

4.2 Problem Formulation: Gene Deletion Strategy Prediction

We now formally define the gene deletion strategy prediction task. Given a constraint-based metabolic model $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2\}$ and a set of metabolites M , the task of gene deletion strategy prediction seeks to infer the gene deletion strategies DS of target metabolites $M_{\text{target}} \subseteq M$. This task operates under the assumption that gene deletion strategies for a subset of metabolites, denoted as DS_{known} , are already established, while the strategies for the target metabolites remain unknown.

The primary objective is to develop a computational agent $\mathcal{F}(\theta)$, parameterized by θ , which processes the known deletion strategies along with information on the metabolic model. This agent aims to generate predictions $\widetilde{DS}$ for the target metabolites, optimizing outputs to closely approximate the ground-truth deletion strategies DS^* .

The key distinction from calculation (Algorithm 2) is that prediction operates on learned representations and patterns from observed strategies rather than explicitly

Algorithm 5 Prediction of Gene Deletion Strategies for Target Metabolites.**Given:** $\mathcal{C}, DS_{\text{known}}$ **Objective:** A computational agent $\mathcal{F}(\theta)$, parameterized by θ , to generate deletion strategy predictions $\widetilde{DS}$ for the target metabolites $M_{\text{target}} \subseteq M$.1: **Such that:**

2: $\widetilde{DS} = \mathcal{F}(\theta; DS_{\text{known}}, \mathcal{C})$

3: $\widetilde{DS} \approx DS^*$

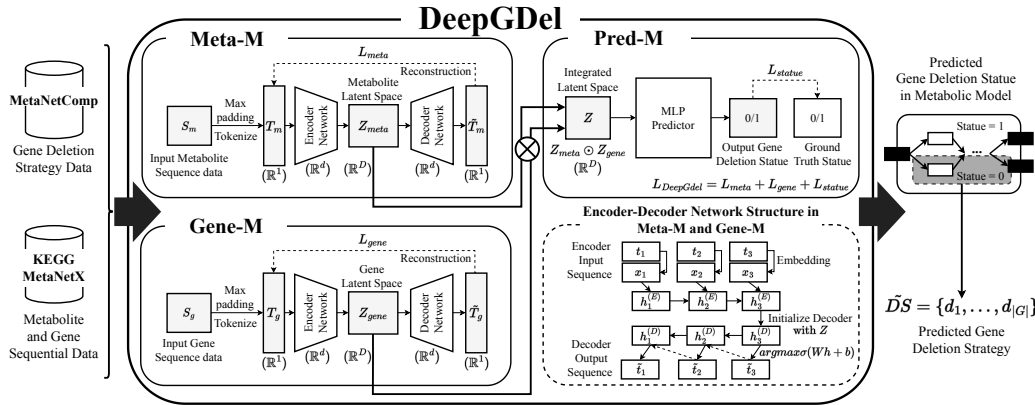


Figure 4.1: System overview of the DeepGDel framework. The framework comprises three neural network-based modules: (1) **Meta-M**, which learns the metabolite latent representation Z_{meta} ; (2) **Gene-M**, which learns the gene latent representation Z_{gene} ; and (3) **Pred-M**, which integrates the two upstream latents into a unified latent representation Z for final gene deletion prediction.

modeling biochemical interactions. While calculation requires a fully parameterized interaction map of all components in $\mathcal{C}_1$ and $\mathcal{C}_2$, prediction focuses on characterizing metabolites and genes as primary information sources.

4.3 DeepGDel Framework

This section presents the DeepGDel framework architecture, which consists of three neural network-based modules working in concert to predict gene deletion strategies.

4.3.1 Framework Overview

The framework operates in two stages. First, representation learning: Meta-M and Gene-M learn latent representations from SMILES and amino acid sequences via unsupervised autoencoder training. Second, prediction: Pred-M integrates these

representations and performs pairwise binary classification (delete/retain) for each gene-metabolite pair.

4.3.2 Gene Deletion Strategy Prediction

The DeepGDeI framework predicts gene deletion strategies through three coordinated modules that learn specialized representations and integrate them for final classification. Each module addresses a distinct aspect of the prediction task: Meta-M captures metabolite chemical structure, Gene-M encodes gene sequence information, and Pred-M combines these representations for deletion status prediction.

Meta-M: Metabolite Representation Module

Meta-M captures metabolite characteristics using an LSTM-based autoencoder [57, 58, 59]. We use SMILES notation as input (Section 2.9).

Given a metabolite SMILES sequence $\mathbf{S}_m$ of length L_m , we first tokenize the sequence into a discrete set of tokens and apply padding to ensure all sequences have uniform length $L_{\max}$. The tokenization step converts the raw metabolite sequence, represented as a string of characters, into a numerical format suitable for processing by the model. Since metabolite sequences consist of characters representing molecular substructures or chemical components, we adopt a character-level tokenization approach.

Formally, given a set of N metabolite sequences $\{\mathbf{S}_m^{(1)}, \mathbf{S}_m^{(2)}, \dots, \mathbf{S}_m^{(N)}\}$, we construct a vocabulary $\mathcal{V}$ containing all unique characters appearing in the dataset. Each sequence $\mathbf{S}_m^{(i)}$ is then mapped into a sequence of token indices:

$$\mathbf{T}_m^{(i)} = [t_1, t_2, \dots, t_{L_m}], \quad t_j \in \mathcal{V}, \quad (4.1)$$

where t_j represents the index corresponding to the j -th character in the sequence. To maintain consistency in sequence lengths, we apply max-padding to all tokenized sequences, ensuring they have a uniform length $L_{\max}$:

$$\mathbf{T}_m^{(i)} = [t_1, t_2, \dots, t_{L_m}, 0, 0, \dots, 0], \quad \text{if } L_m < L_{\max}. \quad (4.2)$$

Here, the padding token (denoted as 0) is appended to shorter sequences to match the maximum length found in the training metabolite sequence data. The max-padding ensures that all input sequences share a consistent dimensionality during

batch processing. During model training and inference, the model ignores padded positions to prevent them from influencing learned representations.

The padded sequence is then mapped to a continuous embedded representation $\mathbf{X}_m \in \mathbb{R}^d$ via an embedding layer:

$$\mathbf{X}_m = \text{Embedding}(\mathbf{T}_m) = W_e[\mathbf{T}_m], \quad (4.3)$$

where $W_e \in \mathbb{R}^{|\mathcal{V}| \times d}$ denotes the embedding weight matrix. Each token index in $\mathbf{T}_m$ selects the corresponding row from W_e , mapping discrete tokens to dense feature vectors. The resulting $\mathbf{X}_m$ captures the high-dimensional feature space of the tokenized input sequence $\mathbf{T}_m$, representing its categorical characteristics in a continuous embedding space.

The encoder processes $\mathbf{X}_m$ through an LSTM to produce hidden states $\mathbf{h}_l$ at each step l :

$$\mathbf{h}_l = \text{LSTM}(\mathbf{x}_l, \mathbf{h}_{l-1}, \mathbf{c}_{l-1}), \quad (4.4)$$

where $\mathbf{h}_l \in \mathbb{R}^D$ is the hidden state at step l , and $\mathbf{c}_l$ is the cell state that maintains long-term memory.

The LSTM computation involves the following gate operations:

$$\mathbf{i}_l = \sigma(\mathbf{W}_i \mathbf{x}_l + \mathbf{U}_i \mathbf{h}_{l-1} + \mathbf{b}_i), \quad (4.5)$$

$$\mathbf{f}_l = \sigma(\mathbf{W}_f \mathbf{x}_l + \mathbf{U}_f \mathbf{h}_{l-1} + \mathbf{b}_f), \quad (4.6)$$

$$\mathbf{o}_l = \sigma(\mathbf{W}_o \mathbf{x}_l + \mathbf{U}_o \mathbf{h}_{l-1} + \mathbf{b}_o), \quad (4.7)$$

$$\tilde{\mathbf{c}}_l = \tanh(\mathbf{W}_c \mathbf{x}_l + \mathbf{U}_c \mathbf{h}_{l-1} + \mathbf{b}_c), \quad (4.8)$$

$$\mathbf{c}_l = \mathbf{f}_l \odot \mathbf{c}_{l-1} + \mathbf{i}_l \odot \tilde{\mathbf{c}}_l, \quad (4.9)$$

$$\mathbf{h}_l = \mathbf{o}_l \odot \tanh(\mathbf{c}_l), \quad (4.10)$$

where $\mathbf{i}_l$, $\mathbf{f}_l$, and $\mathbf{o}_l$ denote the input, forget, and output gates, respectively; $\mathbf{c}_l$ is the cell state storing long-term memory; and $\tilde{\mathbf{c}}_l$ represents the candidate cell state.

The weight matrices and bias terms have the following dimensions:

$$\mathbf{W}_i, \mathbf{W}_f, \mathbf{W}_o, \mathbf{W}_c \in \mathbb{R}^{D \times d}, \quad (4.11)$$

$$\mathbf{U}_i, \mathbf{U}_f, \mathbf{U}_o, \mathbf{U}_c \in \mathbb{R}^{D \times D}, \quad (4.12)$$

$$\mathbf{b}_i, \mathbf{b}_f, \mathbf{b}_o, \mathbf{b}_c \in \mathbb{R}^D, \quad (4.13)$$

where $\mathbf{W}_*$ transforms the input embedding $\mathbf{x}_l \in \mathbb{R}^d$, $\mathbf{U}_*$ transforms the previous

hidden state $\mathbf{h}_{l-1} \in \mathbb{R}^D$, and $\mathbf{b}_*$ are bias terms. The operator $\odot$ denotes element-wise multiplication, while $\sigma(\cdot)$ and $\tanh(\cdot)$ represent the sigmoid and hyperbolic tangent activation functions, respectively.

We apply layer normalization to stabilize training:

$$\mathbf{h}_l^{\text{norm}} = \text{LayerNorm}(\mathbf{h}_l). \quad (4.14)$$

The final latent representation $\mathbf{Z}_{\text{meta}}$ is obtained via average pooling over normalized hidden states:

$$\mathbf{Z}_{\text{meta}} = \text{AvgPooling}(\{\mathbf{h}_l^{\text{norm}}\}_{l=1}^{L_m}), \quad (4.15)$$

where $\mathbf{Z}_{\text{meta}} \in \mathbb{R}^D$.

The decoder decodes the hidden state $\mathbf{h}_l$ at each time step from the latent representation $\mathbf{Z}_{\text{meta}}$:

$$\mathbf{h}_l = \text{LSTM-Decoder}(\mathbf{Z}_{\text{meta}}, \mathbf{h}_{l-1}), \quad (4.16)$$

where the LSTM-Decoder follows similar computation as the encoder.

The decoder is initialized using $\mathbf{Z}_{\text{meta}}$, which serves as the initial hidden and cell states:

$$\mathbf{h}_0 = \mathbf{W}_h \mathbf{Z}_{\text{meta}} + \mathbf{b}_h, \quad \mathbf{c}_0 = \mathbf{W}_c \mathbf{Z}_{\text{meta}} + \mathbf{b}_c. \quad (4.17)$$

The final output of the generated tokenized sequence $\tilde{\mathbf{T}}_m$ is given through:

$$\tilde{\mathbf{T}}_m = \text{GenerateToken}(\mathbf{h}_l), \quad (4.18)$$

where $\text{GenerateToken}(\cdot)$ represents a projection layer followed by a softmax function to generate $\tilde{\mathbf{t}}_l$, corresponding to each character in the tokenized sequence:

$$\tilde{\mathbf{t}}_l = \arg \max \sigma(\mathbf{W}_{\text{out}} \mathbf{h}_l + \mathbf{b}_{\text{out}}), \quad (4.19)$$

where $\sigma(\cdot)$ denotes the softmax activation function, and $\mathbf{W}_{\text{out}}$ and $\mathbf{b}_{\text{out}}$ are the weights and bias of the output projection layer.

Training minimizes the categorical cross-entropy reconstruction loss:

$$\mathcal{L}_{\text{meta}} = \frac{1}{L_m} \sum_{l=1}^{L_m} \text{CrossEntropyLoss}(\tilde{\mathbf{t}}_l, \mathbf{t}_l), \quad (4.20)$$

where $\mathbf{t}_l$ and $\tilde{\mathbf{t}}_l$ are ground-truth and predicted tokens at step l , respectively.

The categorical cross-entropy loss for each token is given by:

$$\text{CrossEntropyLoss}(\tilde{\mathbf{t}}_l, \mathbf{t}_l) = - \sum_{i=1}^C t_{l,i} \log \tilde{t}_{l,i}, \quad (4.21)$$

where C denotes the vocabulary size, $t_{l,i}$ represents the one-hot encoded ground-truth distribution, and $\tilde{t}_{l,i}$ is the predicted probability of token i at position l .

This loss quantifies the dissimilarity between predicted and true probability distributions, encouraging the model to produce a distribution that closely aligns with the true distribution. By minimizing this loss, Meta-M learns to encode the input metabolite sequence into $\mathbf{Z}_{\text{meta}}$ and leverage it to reconstruct the sequence with high fidelity.

Gene-M: Gene Representation Module

Gene-M mirrors Meta-M, using amino acid sequences (Section 2.9).

Given an amino acid sequence $\mathbf{S}_g$ of length L_g , we tokenize, pad, and embed:

$$\mathbf{X}_g = \text{Embedding}(\mathbf{T}_g), \quad (4.22)$$

where $\mathbf{X}_g \in \mathbb{R}^d$.

The encoder-decoder computes:

$$\mathbf{h}_l = \text{LSTM-Encoder}(\mathbf{X}_g, \mathbf{h}_{l-1}), \quad (4.23)$$

$$\mathbf{h}_l^{\text{norm}} = \text{LayerNorm}(\mathbf{h}_l), \quad (4.24)$$

$$\mathbf{Z}_{\text{gene}} = \text{AvgPooling}(\{\mathbf{h}_l^{\text{norm}}\}_{l=1}^{L_g}), \quad (4.25)$$

$$\tilde{\mathbf{T}}_g = \text{GenerateToken}(\text{LSTM-Decoder}(\mathbf{Z}_{\text{gene}}, \mathbf{h}_{l-1})), \quad (4.26)$$

yielding $\mathbf{Z}_{\text{gene}} \in \mathbb{R}^D$.

Training minimizes:

$$\mathcal{L}_{\text{gene}} = \frac{1}{L_g} \sum_{l=1}^{L_g} \text{CrossEntropyLoss}(\tilde{\mathbf{t}}_l, \mathbf{t}_l). \quad (4.27)$$

Pred-M: Prediction Module

Pred-M integrates the learned representations $\mathbf{Z}_{\text{meta}}$ and $\mathbf{Z}_{\text{gene}}$ using element-wise multiplication:

$$\mathbf{Z} = \mathbf{Z}_{\text{meta}} \odot \mathbf{Z}_{\text{gene}}, \quad (4.28)$$

where $\mathbf{Z}_{\text{meta}}$ and $\mathbf{Z}_{\text{gene}}$ are the learned representations of the metabolite and gene sequences, respectively, both residing in the same D -dimensional feature space. The resulting vector $\mathbf{Z} \in \mathbb{R}^D$ combines gene and metabolite information by multiplying the corresponding entries of $\mathbf{Z}_{\text{meta}}$ and $\mathbf{Z}_{\text{gene}}$ at each dimension. The element-wise multiplication serves to emphasize the dimensions where both representations align, while reducing the impact of less significant features. The final vector $\mathbf{Z}$ serves as a compact, interaction-based representation that incorporates both individual and combined features from the gene and metabolite sequences.

A supervised predictor uses $\mathbf{Z}$ as input and outputs the gene deletion status as a binary classification:

$$\hat{y} = \sigma(\text{MLP}(\mathbf{Z})), \quad (4.29)$$

where $\sigma(\cdot)$ is the sigmoid activation function.

The loss function for the deletion status classification task is the binary cross-entropy loss:

$$\mathcal{L}_{\text{pred}} = - \sum_i \left(y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i) \right), \quad (4.30)$$

where y_i is the true label and $\hat{y}_i$ is the predicted probability.

The total loss is:

$$\mathcal{L}_{\text{DeepGDeI}} = \mathcal{L}_{\text{meta}} + \mathcal{L}_{\text{gene}} + \mathcal{L}_{\text{pred}}. \quad (4.31)$$

4.3.3 Implementation Details

Device and Optimization

All neural network models were trained within the proposed framework on an NVIDIA A100-PCIE-40GB GPU with a Xeon Gold 6258R CPU. Model parameters were optimized for 100 epochs using the Adam optimizer with a learning rate of 1×10^{-3} .

Data Splitting and Balancing

We employed a 10-fold stratified cross-validation scheme, repeated five times with different random seeds. In each fold, approximately 10% of metabolites were reserved for testing, while the remainder formed the training set. From the training data, 10% of metabolites were randomly selected as a validation set.

Table 4.1: Grid search for hyperparameter selection. The optimal values are highlighted in underline.

Hyperparameter	Search Space
Embedding dimension ($\mathbb{R}^d$)	{16, <u>32</u> , 64, 128}
LSTM hidden state	{16, 32, <u>64</u> , 128}
Embedding dimension ($\mathbb{R}^D$)	{16, 32, <u>64</u> , 128}
Dropout rate	{ <u>0</u> , 0.1, 0.2}
Batch size	{ <u>8</u> , 16, 32, 64}

To ensure balanced proportions of deleted and non-deleted genes across training, validation, and test sets, we applied a gene-wise balancing procedure: for genes with both deletion states, equal samples were randomly drawn from each state; otherwise, all instances were retained. Next, global class balancing was performed by downsampling the majority class to match the minority, yielding a fully balanced training set. Due to metabolite number constraints, final sample sizes may only approximate the intended ratios. Performance metrics are reported as mean and standard deviation across all folds.

Hyperparameters

The hyperparameters of the neural network models were determined via grid search to identify the optimal configuration, as summarized in Table 4.1. The selected values were adopted as the default settings for the proposed framework.

Computational Complexity and Runtime

The proposed framework adopts an LSTM-based autoencoder as the backbone neural network model. The computational complexity of a single LSTM layer is approximately $\mathcal{O}(L \cdot d \cdot D)$, where L is the sequence length, d is the input embedding size, and D is the number of hidden units. The proposed framework incorporates two LSTM models in both Meta-M and Gene-M. Consequently, the per-step framework complexity is dominated by the terms $\mathcal{O}(c \cdot L \cdot d \cdot D)$, where the factor c arises from the LSTM layer size in the encoder and decoder.

For model training with a batch size of B , the overall computational complexity for parallel processing the input gene sequences of length L_g and metabolite sequences of length L_m is $\mathcal{O}(B \cdot (L_g + L_m) \cdot c \cdot d \cdot D)$. The sequence lengths L_g and L_m are the dominant factors influencing the time complexity, as they are fixed

by the input and directly affect the per-step computations. The actual time cost also depends on setting the number of hidden units, embedding size, and layer size, as all these hyperparameters of the neural network contribute to the overall computational demand. Empirically, the framework, following the optimal values in Table 4.1, demonstrates efficient training, completing 100 epochs in approximately 10 hours under our experimental conditions.

4.4 Toy Model Illustration

To illustrate DeepGDel and the BPM baseline, we apply them to Toy Model 2 from Section 2.5.2 with a new target metabolite.

4.4.1 DeepGDel Application

To apply DeepGDel to the toy model:

- Metabolites are represented via SMILES notation (Section 2.9).
- Genes are represented via amino acid sequences (Section 2.9).
- For the Type 4 gene deletion strategies from Chapter 2 (Toy Model 2, Table 2.3) that achieve growth-coupled production for m_6 , we set labels 0 for deleted genes and 1 for retained genes.

Training proceeds in two stages: first, Meta-M and Gene-M learn latent representations from SMILES and amino acid sequences through unsupervised autoencoder training; then, Pred-M integrates these representations to perform binary classification. The predicted deletion statuses for genes and metabolites align with the ground-truth strategies, confirming that the framework successfully learns predictive patterns from the toy model data.

4.4.2 BPM Baseline Illustration

We also apply the BPM baseline method to Toy Model 2 with a new target metabolite m_1 to demonstrate its execution. We add an auxiliary reaction r_{10} to simulate the secretion of m_1 .

Step 1: Extract Training Data. In this example, the training dataset contains only one maximal gene deletion strategy $D_{\max} = \{g_3, g_4\}$ (Type 4 from Table 2.3) for m_6 .

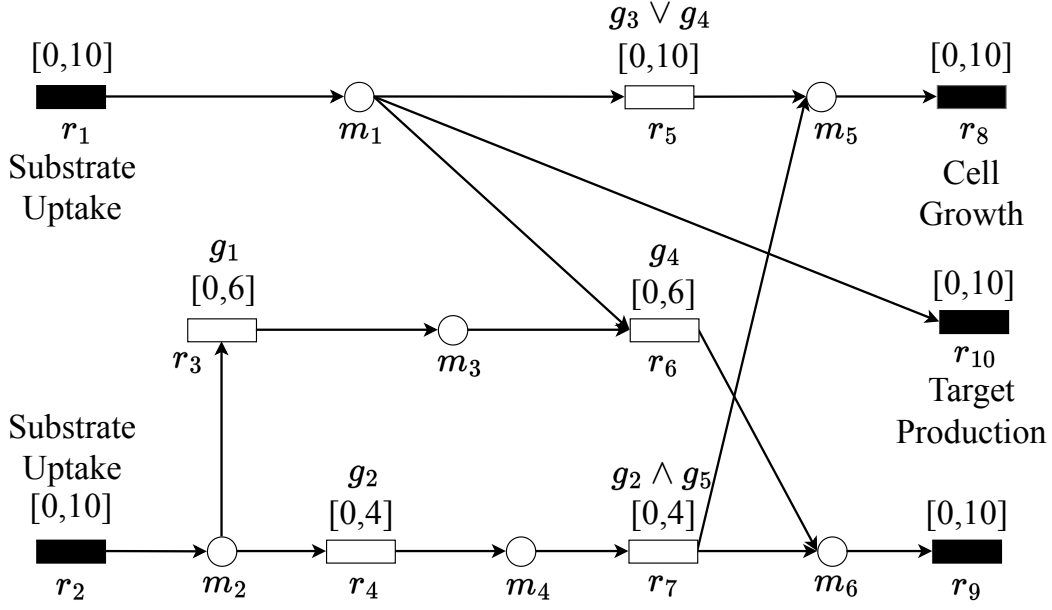


Figure 4.2: Toy model with new target metabolite m_1 for DeepGDeI illustration. An auxiliary reaction r_{10} was added to simulate the secretion of m_1 .

Step 2: Compute Occurrence Counts. For each gene g_i in the toy model, referring to $D_{\max}$, the counts are derived as follows:

$$\begin{aligned} N_0(g_i, m_6) &= 1, \quad N_1(g_i, m_6) = 0 \quad \text{for } g_i \in \{g_3, g_4\}, \\ N_0(g_i, m_6) &= 0, \quad N_1(g_i, m_6) = 1 \quad \text{for } g_i \in \{g_1, g_2, g_5\}. \end{aligned}$$

Step 3: Calculate Non-Deleted Proportions. The proportion of non-deleted occurrences $q_1(g_i)$ for each gene is calculated as:

$$q_1(g_i) = \frac{N_1(g_i)}{N_0(g_i) + N_1(g_i)} = \begin{cases} 0, & i = 3, 4 \\ 1, & i = 1, 2, 5 \end{cases}$$

Step 4: Apply Threshold to Predict States. Using the threshold $\tau = 0.5$, the state of each gene g_i is predicted as:

$$\begin{aligned} g_1 : q_1(g_1) &= 1.0 > 0.5 \Rightarrow \hat{y}(g_1) = 1, \\ g_2 : q_1(g_2) &= 1.0 > 0.5 \Rightarrow \hat{y}(g_2) = 1, \\ g_3 : q_1(g_3) &= 0.0 < 0.5 \Rightarrow \hat{y}(g_3) = 0, \\ g_4 : q_1(g_4) &= 0.0 < 0.5 \Rightarrow \hat{y}(g_4) = 0, \\ g_5 : q_1(g_5) &= 1.0 > 0.5 \Rightarrow \hat{y}(g_5) = 1. \end{aligned}$$

Table 4.2: Summary of constructed datasets for three metabolic models.

Dataset component	e_coli_core	iMM904	iML1515
#Deletion strategies	41	105	443
#Genes per strategy	137	905	1516
#SMILES terms	41	99	425
#Dummy SMILES	0	6	18
#AA sequences	136	889	1513
#Dummy AA sequences	1	16	3
#Balanced train samples	4932	85070	603368
#Balanced test samples	685	9955	68220

As a result, the BPM method predicts the gene deletion strategy for m_1 as: $\mathbf{y}_{\text{pred}} = [1, 1, 0, 0, 1]$. This indicates that g_3 and g_4 are deleted, while g_1 , g_2 , and g_5 remain non-deleted.

4.5 Computational Experiments

This section describes models, data, baselines, metrics, and results.

4.5.1 Experimental Setup

We use three models: *e_coli_core* [47], *iMM904* [48], *iML1515* [49]. Data sources: MetNetComp (strategies), MetaNetX (SMILES), KEGG (amino acid sequences). Splits follow Section 4.3.5; class balancing applied.

Data Balancing

DeepGDel’s training data exhibits severe class imbalance: non-deleted genes (label 1) vastly outnumber deleted genes (label 0). To address this, we generate balanced gene-metabolite pairs for training.

For each gene deletion strategy $\mathbf{y} \in DS_{\text{known}}$ corresponding to metabolite m , we create tuples (g_i, m, y_i) for all genes $g_i \in G$, where $y_i \in \{0, 1\}$ indicates whether gene g_i is deleted (0) or retained (1) in the strategy. We then balance the dataset by sampling equal numbers of deleted and non-deleted instances.

The balanced dataset is split into training/validation (80%) and test (20%) sets, with class balance maintained in each split. As shown in Table 4.2, this balancing process results in substantially increased sample counts: for example, *e_coli_core*

has 4,932 balanced training samples from 41 deletion strategies, and *iML1515* has 603,368 balanced training samples from 443 strategies.

This balanced sampling ensures that the model learns meaningful patterns from both deletion and retention cases rather than being biased toward predicting all genes as non-deleted.

Baseline Methods

We compared DeepGDeI against two baseline methods:

BPM (Baseline Prediction Method): A proportion-based method that extracts training data from known maximal gene deletion strategies and computes occurrence counts. For each gene g_i , BPM counts occurrences across all metabolites in the training set:

$$N_0(g_i) = \sum_{m \in M_{\text{train}}} \mathbb{I}[g_i \text{ deleted in strategy for } m], \quad (4.32)$$

$$N_1(g_i) = \sum_{m \in M_{\text{train}}} \mathbb{I}[g_i \text{ retained in strategy for } m], \quad (4.33)$$

$$q_1(g_i) = \frac{N_1(g_i)}{N_0(g_i) + N_1(g_i)}, \quad (4.34)$$

where M_{train} is the set of metabolites in the training set and $\mathbb{I}[\cdot]$ is the indicator function. The method then applies a threshold $\tau = 0.5$ to predict deletion status: $\hat{y}(g_i) = 1$ if $q_1(g_i) > 0.5$, otherwise $\hat{y}(g_i) = 0$.

DNN (Deep Neural Network): A deep learning-based baseline adopting a simplified design [60]. The DNN takes as input the same tokenized and padded sequences of the gene $\mathbf{T}_g$ and metabolite $\mathbf{T}_m$ as used in DeepGDeI. Unlike the modular architecture of DeepGDeI, this baseline:

$$\mathbf{X}_g = \text{Embedding}(\mathbf{T}_g), \quad (4.35)$$

$$\mathbf{X}_m = \text{Embedding}(\mathbf{T}_m), \quad (4.36)$$

$$\mathbf{X} = [\mathbf{X}_g; \mathbf{X}_m], \quad (4.37)$$

$$\mathbf{h}_1 = \text{ReLU}(\mathbf{W}_1 \mathbf{X} + \mathbf{b}_1), \quad (4.38)$$

$$\hat{y} = \sigma(\mathbf{W}_2 \mathbf{h}_1 + \mathbf{b}_2), \quad (4.39)$$

where $\hat{y} = P(y = 1 \mid \mathbf{T}_g, \mathbf{T}_m; \theta)$, $y \in \{0, 1\}$ denotes the ground-truth label, and $\theta = \{\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}_1, \mathbf{b}_2\}$ represents learnable parameters. The DNN is trained using binary cross-entropy loss with the same setup as DeepGDeI.

Evaluation Metrics

We evaluate model performance using four metrics: overall accuracy, macro-averaged precision, recall, and F1 score, along with macro-averaged AUC.

Overall Accuracy: Accuracy measures the proportion of correctly classified instances out of total instances. Overall accuracy represents the proportion of genes correctly classified as deleted or non-deleted across all metabolites:

$$\text{Overall Accuracy} = \frac{\sum_{m=1}^{|M|} \sum_{i=0}^1 (TP_{m_i} + TN_{m_i})}{|M| \times |G|}, \quad (4.40)$$

where TP and TN denote true positives and true negatives.

Macro-Averaged Precision: Precision measures the proportion of true positive predictions out of all positive predictions for each class. Macro-averaged precision is the average of precision values across both classes:

$$\text{Macro-Averaged Precision} = \frac{1}{|M|} \sum_{m=1}^{|M|} \frac{1}{2} \sum_{i=0}^1 \frac{TP_{m_i}}{TP_{m_i} + FP_{m_i}}. \quad (4.41)$$

Macro-Averaged Recall: Recall (sensitivity) measures the proportion of true positives out of all actual positives for each class. Macro-averaged recall is the average across both classes:

$$\text{Macro-Averaged Recall} = \frac{1}{|M|} \sum_{m=1}^{|M|} \frac{1}{2} \sum_{i=0}^1 \frac{TP_{m_i}}{TP_{m_i} + FN_{m_i}}. \quad (4.42)$$

Macro-Averaged F1 Score: F1 is the harmonic mean of precision and recall for each class. Macro-averaged F1 is the average across both classes:

$$\text{Macro-Averaged F1 Score} = \frac{1}{|M|} \sum_{m=1}^{|M|} \frac{1}{2} \sum_{i=0}^1 2 \cdot \frac{\text{Prec}_{m_i} \cdot \text{Recall}_{m_i}}{\text{Prec}_{m_i} + \text{Recall}_{m_i}}. \quad (4.43)$$

Macro-Averaged AUC: Area under the ROC curve, averaged across classes.

All metrics are reported as mean $\pm$ std over folds.

4.5.2 Results and Analysis

Performance Comparison with Baselines

The results of performance comparisons with baseline methods across the three metabolic models under study are summarized in Table 4.3.

For *e_coli_core*: Table 4.3 summarizes the performance comparison between the proposed framework DeepGDeI and baseline methods in the context of targeting GCP-possible metabolites of *e_coli_core* metabolic model. Regarding overall accuracy, DeepGDeI achieved the highest score of 71.90% across all tested target metabolites. In comparison, the BPM and DNN baselines achieved 57.21% and 62.68%, respectively. Regarding the macro-averaged precision, recall, F1 score, and AUC, DeepGDeI achieved the highest performance with 71.72%, 72.15%, 70.14%, and 71.88%, respectively. In comparison, the BPM and DNN baselines showed lower and less balanced results: BPM achieved 56.68%, 71.04%, 61.35%, and 56.74%, while DNN achieved 64.35%, 64.78%, 60.19%, and 62.04%, respectively.

For *iMM904*: Table 4.3 summarizes the performance comparison between the proposed framework DeepGDeI and baseline methods in the context of targeting GCP-possible metabolites of *iMM904* metabolic model. Regarding overall accuracy, DeepGDeI achieved the highest score of 79.78% across all tested target metabolites. In comparison, the BPM and DNN baselines achieved 57.27% and 69.98%, respectively. Regarding the macro-averaged precision, recall, F1 score, and AUC, DeepGDeI achieved the highest performance with 78.29%, 77.08%, 77.25%, and 79.45%, respectively. In comparison, the BPM and DNN baselines showed lower and less balanced results: BPM achieved 54.38%, 73.29%, 60.00%, and 53.06%, while DNN achieved 72.48%, 67.84%, 64.20%, and 67.17%, respectively.

For *iML1515*: Table 4.3 summarizes the performance comparison between the proposed framework DeepGDeI and baseline methods in the context of targeting GCP-possible metabolites of *iML1515* metabolic model. Regarding overall accuracy, DeepGDeI achieved the highest score of 67.24% across all tested target metabolites. In comparison, the BPM and DNN baselines achieved 54.21% and 60.44%, respectively. Regarding the macro-averaged precision, recall, F1 score, and AUC, DeepGDeI achieved the highest performance with 69.59%, 63.76%, 62.59%, and 66.45%, respectively. In comparison, the BPM and DNN baselines showed lower and less balanced results: BPM achieved 54.04%, 60.03%, 55.15%, and 51.77%, while DNN achieved 63.96%, 62.09%, 58.11%, and 59.70%, respectively.

Ablation Study

The results of ablation tests on input data type across the three metabolic models under study are summarized in Table 4.4.

Table 4.3: Performance comparison between DeepGDeI and baseline methods. Best results in bold. M-Avg. = Macro-averaged.

Metric	e_coli_core			iMM904			iML1515		
	BPM	DNN	DeepGDeI	BPM	DNN	DeepGDeI	BPM	DNN	DeepGDeI
Overall Accuracy (%)	57.21	62.68	71.90	57.27	69.98	79.78	54.21	60.44	67.24
M-Avg. Precision (%)	56.68	64.35	71.72	54.38	72.48	78.29	54.04	63.96	69.59
M-Avg. Recall (%)	71.04	64.78	72.15	73.29	67.84	77.08	60.03	62.09	63.76
M-Avg. F1 Score (%)	61.35	60.19	70.14	60.00	64.20	77.25	55.15	58.11	62.59
M-Avg. AUC (%)	56.74	62.04	71.88	53.06	67.17	79.45	51.77	59.70	66.45

For *e_coli_core*: Table 4.4 summarizes the performance comparison between two ablation variants of DeepGDeI, i.e., with only input SMILES data of metabolites or AA sequence of genes, in the context of targeting GCP-possible metabolites of the *e_coli_core* metabolic model. When utilizing only SMILES data of metabolites as input, the overall accuracy achieved is 56.05%. In comparison, utilizing only AA sequence data of genes as input achieved a better overall accuracy of 62.14%. Regarding the macro-averaged precision, recall, F1 score, and AUC, the variant with only input of metabolite SMILES data achieved results of 52.59%, 51.76%, 48.65%, and 51.64% respectively. In comparison, the variant with only input of gene AA sequence data showed less balanced performance with 70.75%, 41.38%, 52.22%, and 61.51% respectively.

For *iMM904*: Table 4.4 summarizes the performance comparison between two ablation variants of DeepGDeI, in the context of targeting GCP-possible metabolites of the *iMM904* metabolic model. When utilizing only SMILES data of metabolites as input, the overall accuracy achieved is 64.86%. In comparison, utilizing only AA sequence data of genes as input achieved a better overall accuracy of 72.97%. Regarding the Macro-averaged precision, recall, F1 score, and AUC, the variant with only input of metabolite SMILES data achieved results of 73.06%, 65.64%, 62.10%, and 62.47% respectively. In comparison, the variant with only input of gene AA sequence data showed better performance with 78.23%, 69.49%, 69.17%, and 71.88% respectively.

For *iML1515*: Table 4.4 summarizes the performance comparison between two ablation variants of DeepGDeI, in the context of targeting GCP-possible metabolites of the *iML1515* metabolic model. When utilizing only SMILES data of metabolites as input, the overall accuracy achieved is 54.05%. In comparison, utilizing only AA sequence data of genes as input achieved a better overall accuracy of 61.16%. Regarding the macro-averaged precision, recall, F1 score, and AUC, the variant

Table 4.4: Ablation comparison: Met-only vs. Gene-only inputs. Best results in bold.

Metric	e_coli_core		iMM904		iML1515	
	Met-only	Gene-only	Met-only	Gene-only	Met-only	Gene-only
Overall Accuracy (%)	56.05	62.14	64.86	72.97	54.05	61.16
M-Avg. Precision (%)	52.59	70.75	73.06	78.23	57.11	66.81
M-Avg. Recall (%)	51.76	41.38	65.64	69.49	54.82	61.40
M-Avg. F1 Score (%)	48.65	52.22	62.10	69.17	50.43	58.49
M-Avg. AUC (%)	51.64	61.51	62.47	71.88	53.28	60.85

with only input of metabolite SMILES data achieved results of 57.11%, 54.82%, 50.43%, and 53.28% respectively. In comparison, the variant with only input of gene AA sequence data showed better performance with 66.81%, 61.40%, 58.49%, and 60.85% respectively.

4.6 Discussion

This chapter presented DeepGDeI, a deep learning framework that reformulates gene deletion strategy design as a supervised prediction problem, departing fundamentally from optimization-based approaches. The experimental results demonstrate the effectiveness of this data-driven paradigm: DeepGDeI achieves substantial accuracy improvements of 14.69%, 22.52%, and 13.03% over baselines across *e_coli_core*, *iMM904*, and *iML1515*, respectively, while maintaining balanced precision and recall across all models. These improvements indicate that the learned representations capture meaningful patterns in gene-metabolite relationships that simple heuristics or naive neural networks fail to extract.

The ablation studies provide important insights into the complementary nature of metabolite and gene representations. When using only SMILES-based metabolite features, performance drops substantially (e.g., 71.90% to 56.05% in *e_coli_core*), demonstrating that metabolite chemical structure alone is insufficient. Similarly, gene-only features achieve better but still suboptimal performance (62.14% in *e_coli_core*), indicating that gene sequence information provides stronger but incomplete signal. The full framework’s superior performance across all models confirms that both modalities are necessary: metabolite characteristics inform which genes are relevant for production, while gene sequences encode functional constraints that guide deletion decisions.

A key innovation of DeepGDel is its status as the first supervised machine learning formulation for gene deletion strategy prediction. By learning from existing successful strategies, the framework avoids the explicit network modeling and iterative optimization required by traditional methods, enabling millisecond-scale predictions that scale to large metabolic models. The LSTM-based autoencoder architecture captures sequential patterns in both SMILES and amino acid sequences, learning representations that encode structural and functional information relevant for deletion decisions. This modular design, with separate unsupervised representation learning followed by supervised classification, enables the model to leverage large amounts of unlabeled sequence data through autoencoder pretraining while focusing the supervised learning on the specific prediction task.

However, several limitations deserve acknowledgment. The framework is restricted to sequential representations and lacks explicit modeling of network topology, which may limit its ability to capture pathway-level interactions that are crucial for understanding metabolic coupling. The learned representations are essentially black boxes: while the autoencoders successfully encode relevant information, we cannot articulate what specific biological properties they capture or why particular deletion predictions are made. Additionally, the framework requires substantial labeled data from sources like MetNetComp, creating a dependency that may limit applicability to emerging organisms or newly characterized metabolites. The LSTM architecture represents a conservative choice that provides strong baseline performance but may be superseded by more advanced models such as transformer-based architectures [61] in future work.

A fundamental limitation of DeepGDel is its exclusive reliance on sequential representations—amino acid sequences for genes and SMILES strings for metabolites—which cannot capture the topological relationships among metabolites in metabolic networks. Metabolic systems are inherently networked, with pathway-level coupling and structural dependencies that determine how flux redistribution following gene deletions affects growth-coupled production. Chapter 5 addresses this limitation by extending DeepGDel with graph-based learning, integrating metabolic network topology into the prediction framework while maintaining the data-driven paradigm.

Chapter 5

GraphGDel

This chapter presents the third contribution of this dissertation: a graph-enhanced framework for predicting gene deletion strategies that extends DeepGDel by incorporating network structure. We first motivate the need for graph-based learning, present a systematic pipeline for constructing metabolic graphs, introduce the GraphGDel framework, illustrate its application, and provide experimental validation.

5.1 Introduction

DeepGDel [62] (Chapter 4) successfully demonstrated that deep learning approaches can accelerate gene deletion strategy design by learning from existing successful strategies, achieving substantial accuracy improvements through sequence-based representation learning. However, this framework focuses exclusively on sequential data—amino acid sequences for genes and SMILES strings for metabolites—while metabolic systems are fundamentally networked entities with complex topological relationships. Metabolites do not act in isolation; they participate in multiple interconnected pathways where flux redistribution following gene deletions depends critically on the structural coupling between metabolic components. This network topology is essential for understanding how growth and

production become coupled, and ignoring these structural relationships may limit the predictive power of any gene deletion strategy framework.

Graph representations provide a natural and powerful framework for capturing these structural interactions. Unlike sequential encodings, graphs explicitly model entities as nodes and their relationships as edges, enabling algorithms to leverage topological patterns such as path connectivity, clustering, and centrality. Recent advances in graph neural networks (GNNs) have demonstrated remarkable success on biological data, extracting meaningful structural patterns for tasks ranging from protein-protein interaction prediction to molecular property estimation. By applying graph-based learning to metabolic networks, we can potentially capture pathway-level constraints and coupling relationships that sequential representations alone cannot access.

Integrating graph learning into gene deletion strategy prediction offers clear potential benefits but also presents significant technical challenges. Unlike manually curated pathway databases like KEGG, which provide explicit graphical representations with nodes and edges, genome-scale metabolic models (GEMs) are fundamentally algebraic structures: stoichiometric matrices encode mass balance constraints, Boolean GPR rules link genes to reactions, and flux bounds define reaction capacity limits. This algebraic formulation does not immediately yield an interpretable graph with nodes representing biological entities and edges encoding their interactions, creating a representational gap that must be bridged systematically. We therefore require a principled pipeline to construct biologically meaningful graphs from these algebraic formulations while preserving the essential structural logic of the underlying metabolic network.

A second fundamental challenge arises from the presence of currency metabolites that participate ubiquitously in metabolic reactions. Metabolites such as ATP, NADH, NADPH, water, and protons appear as substrates or products in hundreds of reactions across diverse pathways, creating densely connected hub nodes that can fundamentally distort the network's topological properties. For example, ATP is produced or consumed in virtually all energy-coupling reactions, creating edges to nearly every metabolic pathway in the network. These highly connected hubs obscure pathway-specific structure, create spurious shortest paths between unrelated metabolites, and introduce substantial topological noise that may confound graph-based learning algorithms. While curated databases like KEGG and BiGG manually collapse or remove these currency metabolites to improve visual clarity

and pathway interpretation, the systematic impact of these topological artifacts on graph-based learning remains underexplored.

This chapter introduces GraphGDel (Graph-based Gene Deletion framework), a novel extension of DeepGDel that incorporates metabolite graph modeling to address these challenges. GraphGDel makes three key contributions: (1) a systematic four-step graph construction pipeline that transforms algebraic GEM representations into biologically meaningful metabolite graphs, handling both the representational gap and the currency metabolite problem through principled refinement; (2) a new Graph-M module that uses graph convolutional networks (GCNs) to learn structure-aware metabolite representations, complementing the existing sequential features from Gene-M and Meta-M; (3) empirical validation demonstrating that this is the first successful application of graph learning to gene deletion prediction, with consistent performance improvements of 14.04%, 16.26%, and 13.18% over DeepGDel across three metabolic models of varying scales.

The GraphGDel framework extends DeepGDel’s modular architecture with four coordinated components: Gene-M and Meta-M provide sequential encodings of gene and metabolite characteristics as established in Chapter 4; Graph-M introduces graph-based metabolite representation learning that captures network topology through GCNs operating on the constructed metabolite graphs; Pred-M integrates these complementary representations for final deletion status prediction. The graph construction pipeline produces refined metabolite graphs that preserve essential metabolic structure while eliminating confounding topological noise: initial extraction creates a tripartite graph from GEM components; projection yields metabolite-level connectivity; degree-based filtering removes extreme hubs; and domain knowledge refinement eliminates canonical currency metabolites. Ablation studies validate that each refinement step contributes to improved prediction performance, confirming that both topological simplification and biological interpretation are necessary for effective graph-based learning.

5.2 Graph Construction Pipeline

We propose a systematic pipeline to construct biologically meaningful metabolite graphs from algebraic genome-scale metabolic model representations. The process begins by extracting core biochemical entities and their interactions, followed by two refinement steps that incorporate both graph-structural properties and domain-specific metabolite knowledge. The resulting graph preserves essential metabolic

structure while reducing confounding topological noise, thereby enabling more interpretable and effective downstream graph-based learning.

The construction procedure consists of four sequential steps:

5.2.1 Step 1: Temporary Tripartite Graph

We begin with a genome-scale metabolic model as input. As the metabolic model lacks an explicit topological form, we draft a temporary graph $\mathcal{G}_{\text{temp}}$ from its components. We construct $\mathcal{G}_{\text{temp}}$ as a tripartite heterogeneous graph consisting of three node types: metabolites, reactions, and genes. Directed edges are defined from metabolites to reactions (for substrates), from reactions to metabolites (for products), and from genes to reactions (based on GPR rules). This graph encodes the complete structural and regulatory logic of the metabolic model.

5.2.2 Step 2: Metabolite Graph Projection

To focus on metabolite-level connectivity, we remove gene nodes and treat each reaction node in $\mathcal{G}_{\text{temp}}$ as a hyperedge connecting all its associated metabolites. We then project this hypergraph into a pairwise undirected graph.

Formally, for each reaction r_j :

- Let $Sub_j = \{m_i | S_{ij} < 0\}$ denote the set of substrate metabolites (consumed by reaction r_j)
- Let $Prod_j = \{m_i | S_{ij} > 0\}$ denote the set of product metabolites (produced by reaction r_j)
- Add undirected edge $(m_i, m_{i'})$ for all $m_i \in Sub_j$ and $m_{i'} \in Prod_j$

This projection converts each hyperedge (reaction) into a complete bipartite graph connecting all substrates to all products, effectively linking each reactant to each product within the same reaction. The resulting metabolite graph, denoted as $\mathcal{G}_0$, contains only metabolite nodes, with edges representing biochemical transformations.

5.2.3 Step 3: Graph Attribute-Based Node Filtering

Highly connected hub-like metabolites can distort the network's topological structure by disproportionately influencing properties such as shortest paths and information flow. To reduce this topological noise, we identify and remove from $\mathcal{G}_0$ the ten

metabolites with the highest degree, resulting in a trimmed graph $\mathcal{G}_1$. This threshold was selected to remove the most extreme topological outliers while preserving the majority of metabolic structure.

5.2.4 Step 4: Knowledge-Based Node Refinement

We apply a two-step refinement process to enhance biological interpretability, resulting in the final graph $\mathcal{G}_{\text{final}}$:

(1) Remove canonical currency metabolites.

Currency metabolites such as ATP, NADH, CoA, H₂O, and others participate in many biochemical reactions across the metabolic network, creating highly connected hub nodes that dominate graph topology [63]. Their high connectivity obscures the modular organization of metabolic pathways and adds substantial topological noise to graph-based learning. Although such entities are commonly collapsed or excluded in databases like KEGG and BiGG [64] to improve visual clarity, their impact on graph-based learning remains underexplored. Removing these metabolites reduces this topological noise while preserving the biochemical transformation structure essential for understanding metabolic pathways. We exclude known currency metabolites based on standardized metabolite identifiers from the MetaNetX database [65], which provides consistent mappings across organisms. This database-centric approach enables systematic currency metabolite identification and removal: ["atp", "adp", "amp", "nad", "nadh", "nadp", "nadph", "coa", "h", "h2o", "pi", "ppi", "co2", "nh4", "o2", "glutathione", "na1", "k", "mg2", "ca2"].

(2) Selectively restore removed metabolites. Any metabolite removed in previous steps that corresponds to a designated production target (e.g., a growth-coupled product) or a non-currency, high-degree metabolite is selectively reintroduced to maintain the functional integrity of the $\mathcal{G}_{\text{final}}$.

As a result, the proposed graph construction pipeline produces a representation that preserves key functional structures while minimizing the confounding influence of topological hubs.

5.3 GraphGDel Framework

GraphGDel uses the four modules below.

Algorithm 6 Constructing Metabolite Graph from a Constraint-Based Metabolic Model

Input: $\mathcal{C} = \{M, R, S, G, F\}$ where:

- $M = \{m_1, \dots, m_{|M|}\}$: metabolites (including m_{target})
- $R = \{r_1, \dots, r_{|R|}\}$: reactions
- $S \in \mathbb{R}^{|M| \times |R|}$: stoichiometric matrix, where S_{ij} is the coefficient of m_i in r_j
- $G = \{g_1, \dots, g_{|G|}\}$: genes
- $F = \{f_1, \dots, f_{|R|}\}$: GPR Boolean functions mapping subsets of G to reactions

Currency Metabolite Set: $\mathcal{U} = \{\text{atp, adp, amp, nad, nadh, nadp, nadph, coa, h, h2o, pi, ppi, co2, nh4, o2, glutathione, na1, k, mg2, ca2}\}$.

Step 1: Construct Temporary Graph

Define node set: $V_{\text{temp}} = M \cup R \cup G$

Define directed edge set E_{temp} :

- Add (m_i, r_j) if $S_{ij} < 0$ (substrate)
- Add (r_j, m_i) if $S_{ij} > 0$ (product)
- Add (g_k, r_j) if g_k appears in f_j (GPR rule)

Result: $\mathcal{G}_{\text{temp}} = (V_{\text{temp}}, E_{\text{temp}})$

Step 2: Induce Metabolite Graph $\mathcal{G}_0$

Project each reaction into undirected metabolite pairs:

- For each reaction r_j , for every substrate m_i ($S_{ij} < 0$) and every product $m_{i'}$ ($S_{i'j} > 0$), add edge $(m_i, m_{i'})$.

Result: $\mathcal{G}_0 = (V_0 = M, E_0)$

Step 3: Graph Attribute-Based Node Filtering

Compute degrees: $\deg(m) = |\{v \mid (m, v) \in E_0, m, v \in V_0\}|$

Select top- k : $H_k = \text{Top-}k \text{ metabolites by degree}$

Remove: $V_1 = V_0 \setminus H_k$, $E_1 = \{(m_i, m_j) \in E_0 \mid m_i, m_j \in V_1\}$

Result: $\mathcal{G}_1 = (V_1, E_1)$

Step 4: Knowledge-Based Node Refinement

(1) Remove currency metabolites: $V_2 = V_1 \setminus \mathcal{U}$, $E_2 = \{(m_i, m_j) \in E_1 \mid m_i, m_j \in V_2\}$

(2) Restore removed metabolites: $T = \{m \in H_k \mid m = m_{\text{target}} \text{ or } m \notin \mathcal{U}\}$

$V_3 = V_2 \cup T$, $E_T = \{(m_i, m_j) \in E_0 \mid (m_i \in T \text{ or } m_j \in T) \text{ and } m_i, m_j \in V_3\}$

$E_3 = E_2 \cup E_T$

Output: $\mathcal{G}_{\text{final}} = (V_3, E_3)$

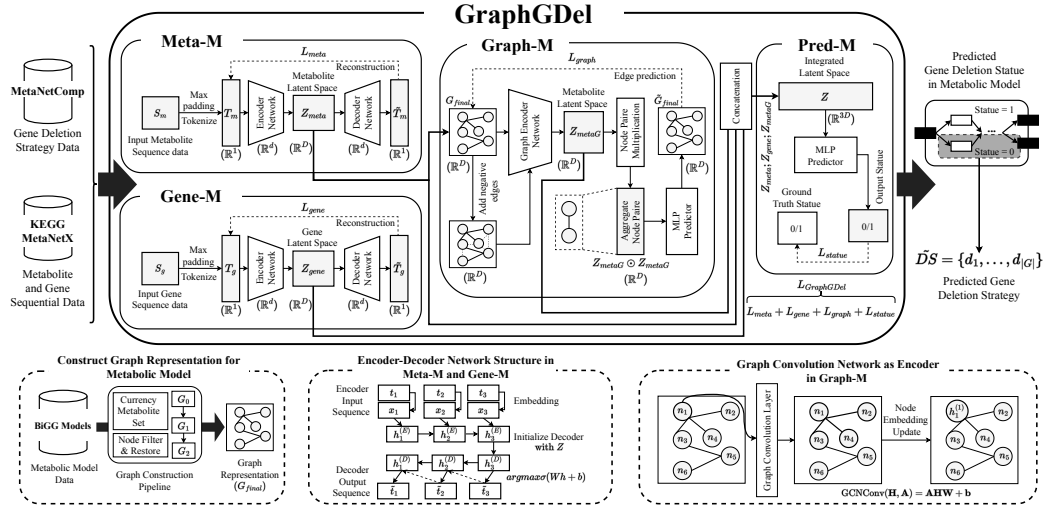


Figure 5.1: System overview of the GraphGDel framework. The framework comprises four neural network-based modules: (1) **Gene-M** and (2) **Meta-M**, which learn latent representations from sequential data; (3) **Graph-M**, which learns structure-aware metabolite representations from the constructed graph; and (4) **Pred-M**, which integrates all representations for gene deletion prediction.

5.3.1 Framework Overview

GraphGDel extends DeepGDel by adding Graph-M, a GNN-based module that learns structure-aware metabolite representations. We compute metabolite features from the SMILES-based Meta-M and use them as node features in Graph-M. The GNN aggregates neighbor information to produce Z_{metaG} ; Pred-M then integrates Z_{gene} (Gene-M), Z_{meta} (Meta-M), and Z_{metaG} for prediction.

5.3.2 Gene-M and Meta-M

GraphGDel employs the same sequence representation modules as DeepGDel (Chapter 4, Sections 4.3.2–4.3.2), which we briefly summarize here.

Meta-M: Metabolite Representation Learning. The Meta-M module transforms SMILES sequences into dense feature vectors using an LSTM-based autoencoder architecture [66]. Given a SMILES sequence S_m of length L_m , the module performs character-level tokenization to obtain $T_m = [t_1, t_2, \dots, t_{L_m}]$, where each t_j is a token index in the vocabulary. The tokenized sequence is embedded into a continuous vector space:

$$\mathbf{X}_m = \text{Embedding}(\mathbf{T}_m), \quad (5.1)$$

and then processed through an LSTM encoder:

$$\mathbf{h}_l = \text{LSTM}(\mathbf{x}_l, \mathbf{h}_{l-1}, \mathbf{c}_{l-1}), \quad (5.2)$$

where $\mathbf{h}_l \in \mathbb{R}^D$ is the hidden state and $\mathbf{c}_l$ is the cell state. After applying layer normalization, the final metabolite representation is obtained via average pooling:

$$\mathbf{Z}_{\text{meta}} = \frac{1}{L_m} \sum_{l=1}^{L_m} \mathbf{h}_l^{\text{norm}}. \quad (5.3)$$

The decoder reconstructs the SMILES sequence, and training minimizes the reconstruction loss:

$$\mathcal{L}_{\text{meta}} = \frac{1}{L_m} \sum_{l=1}^{L_m} \text{CrossEntropy}(\tilde{\mathbf{t}}_l, \mathbf{t}_l), \quad (5.4)$$

where $\mathbf{t}_l$ and $\tilde{\mathbf{t}}_l$ are ground-truth and predicted tokens.

Gene-M: Gene Representation Learning. The Gene-M module employs an identical architecture to Meta-M but processes amino acid sequences instead of SMILES strings. Given a gene sequence $\mathbf{S}_g$ of length L_g , the module follows the same processing pipeline:

$$\mathbf{X}_g = \text{Embedding}(\mathbf{T}_g), \quad (5.5)$$

$$\mathbf{h}_l = \text{LSTM}(\mathbf{x}_l, \mathbf{h}_{l-1}, \mathbf{c}_{l-1}), \quad (5.6)$$

$$\mathbf{h}_l^{\text{norm}} = \text{LayerNorm}(\mathbf{h}_l), \quad (5.7)$$

$$\mathbf{Z}_{\text{gene}} = \frac{1}{L_g} \sum_{l=1}^{L_g} \mathbf{h}_l^{\text{norm}}, \quad (5.8)$$

$$\tilde{\mathbf{T}}_g = \text{Decoder}(\mathbf{Z}_{\text{gene}}), \quad (5.9)$$

with training objective:

$$\mathcal{L}_{\text{gene}} = \frac{1}{L_g} \sum_{l=1}^{L_g} \text{CrossEntropy}(\tilde{\mathbf{t}}_l, \mathbf{t}_l). \quad (5.10)$$

This parallel design ensures balanced feature extraction from both biological entities.

5.3.3 Graph-M: Graph Representation Module

To capture topological relationships within the metabolic network, Graph-M employs Graph Convolutional Networks (GCN) to refine metabolite representations

learned by Meta-M. This module takes sequence-based metabolite features from Meta-M as initial node embeddings and enhances them by incorporating graph connectivity patterns.

Given a metabolic graph $\mathcal{G} = (V, E)$, we initialize node features using Meta-M representations:

$$\mathbf{X} = [\mathbf{Z}_{\text{meta}}^{(1)}, \mathbf{Z}_{\text{meta}}^{(2)}, \dots, \mathbf{Z}_{\text{meta}}^{(|V|)}]^T, \quad (5.11)$$

where each row $\mathbf{X}_i$ corresponds to the Meta-M learned representation of metabolite v_i .

We add learnable positional embeddings:

$$\mathbf{X}_{\text{enhanced}} = \mathbf{X} + \mathbf{P}, \quad (5.12)$$

where $\mathbf{P} \in \mathbb{R}^{|V| \times d_{\text{pos}}}$ represents positional embeddings.

Enhanced features are processed through two graph convolutional layers:

$$\mathbf{H}^{(1)} = \text{ReLU}(\text{GCNConv}(\mathbf{X}_{\text{enhanced}}, \mathbf{A})), \quad (5.13)$$

$$\mathbf{H}^{(2)} = \text{GCNConv}(\mathbf{H}^{(1)}, \mathbf{A}), \quad (5.14)$$

where $\mathbf{A}$ is the adjacency matrix, and:

$$\text{GCNConv}(\mathbf{H}, \mathbf{A}) = \mathbf{A}\mathbf{H}\mathbf{W} + \mathbf{b}, \quad (5.15)$$

with learnable $\mathbf{W}$ and $\mathbf{b}$.

The final graph-refined embeddings are:

$$\mathbf{Z}_{\text{metaG}} = \mathbf{H}^{(2)}. \quad (5.16)$$

To train Graph-M, we use a link prediction task. For node pair (i, j) :

$$s_{ij} = \sigma(\mathbf{W}_{\text{link}}[\mathbf{Z}_{\text{metaG}}^{(i)} \parallel \mathbf{Z}_{\text{metaG}}^{(j)}] + \mathbf{b}_{\text{link}}), \quad (5.17)$$

where $\parallel$ denotes concatenation.

The training objective is:

$$\mathcal{L}_{\text{graph}} = - \sum_{(i,j) \in E} \log(s_{ij}) - \sum_{(i,j) \notin E} \log(1 - s_{ij}), \quad (5.18)$$

where the first term corresponds to positive edges and the second to negative edges.

5.3.4 Pred-M: Prediction Module

Pred-M integrates learned representations from all three upstream modules. For a specific gene-metabolite pair, we select the corresponding metabolite node from the graph embeddings:

$$\mathbf{Z}_{\text{metaG}}^{(m)} = \mathbf{Z}_{\text{metaG}}[m], \quad (5.19)$$

where m is the index of the target metabolite in the graph.

The three representations are concatenated to form a combined feature vector:

$$\mathbf{Z} = [\mathbf{Z}_{\text{meta}} \parallel \mathbf{Z}_{\text{gene}} \parallel \mathbf{Z}_{\text{metaG}}^{(m)}], \quad (5.20)$$

where $\parallel$ denotes concatenation, resulting in a feature vector of dimension $3D$.

The combined representation is processed through a fully connected layer followed by sigmoid activation:

$$\hat{y} = \sigma(\mathbf{W}_{\text{pred}}\mathbf{Z} + \mathbf{b}_{\text{pred}}), \quad (5.21)$$

where $\mathbf{W}_{\text{pred}} \in \mathbb{R}^{1 \times 3D}$ and $\mathbf{b}_{\text{pred}} \in \mathbb{R}$ are learnable parameters.

The output $\hat{y} \in [0, 1]$ represents the probability that the gene should be deleted (when $\hat{y}$ is close to 0) or retained (when $\hat{y}$ is close to 1).

The training objective for the prediction task is the binary cross-entropy loss:

$$\mathcal{L}_{\text{pred}} = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y}), \quad (5.22)$$

where $y \in \{0, 1\}$ is the ground-truth label.

The overall training objective combines all four loss components:

$$\mathcal{L}_{\text{GraphGDeI}} = \mathcal{L}_{\text{meta}} + \mathcal{L}_{\text{gene}} + \mathcal{L}_{\text{graph}} + \mathcal{L}_{\text{pred}}. \quad (5.23)$$

This multi-task learning approach ensures each module learns meaningful representations while contributing to the final prediction. The framework is trained end-to-end, allowing representations from each module to be optimized jointly.

5.3.5 Implementation Details

All neural network models were trained on an NVIDIA A100-PCIE-40GB GPU with a Xeon Gold 6258R CPU. Model parameters were optimized for 100 epochs using the Adam optimizer with a learning rate of 1×10^{-3} .

Data partitioning: 90% for training/validation and 10% as an independent test set. Within the training/validation portion, we applied 10-fold stratified splitting. After selecting the best configuration from hyperparameter search, we retrained on the entire training/validation set and evaluated on the test set.

Computational complexity is dominated by three components: LSTM autoencoders, GCNs, and the prediction module. For LSTM autoencoders, complexity is $\mathcal{O}(L \cdot d \cdot D)$ per layer. The Graph-M module has complexity $\mathcal{O}(|E| \cdot D)$ per GCN layer. Overall complexity for batch size B : $\mathcal{O}(B \cdot (L_g + L_m) \cdot d \cdot D + B \cdot |E| \cdot D + B \cdot 3D)$. Empirically, training completes 100 epochs in approximately 10 hours under our experimental conditions.

5.4 Computational Experiments

We evaluate GraphGDeI using three experiments: baseline comparison, module ablation, and graph representation comparison.

5.4.1 Experimental Setup

GraphGDeI builds upon the experimental design established in DeepGDeI (Chapter 4, Section 4.5.1), with the addition of graph representations constructed via the pipeline described in Section 5.2. We evaluate the framework using three metabolic models of varying scales: *e_coli_core* [47], *imm904* [48], and *iml1515* [49], which represent simplified bacterial core metabolism, comprehensive yeast modeling, and detailed genome-scale bacterial modeling, respectively. These models span different scales and organism types, enabling validation of the framework’s generalizability across diverse metabolic contexts.

For each model, we construct datasets integrating four data modalities: (1) gene deletion strategies from MetNetComp, providing ground-truth labels for training and evaluation; (2) SMILES sequences for metabolites from MetaNetX, enabling chemical structure-aware representation; (3) amino acid sequences for genes from KEGG, capturing protein-level information; and (4) metabolite graphs constructed via Algorithm 6, encoding topological relationships between metabolites. The dataset characteristics and sample counts are provided in Table 4.2 from Chapter 4, with GraphGDeI utilizing the same balanced training and test splits to ensure fair comparison with DeepGDeI.

The experimental protocol follows a rigorous cross-validation design: we employ 10-fold stratified cross-validation with 5 random seeds, ensuring that each metabolite appears exclusively in either training or test sets across all folds. This design accounts for metabolite-specific variation while providing robust statistical estimates. Severe class imbalance in the raw data—where non-deleted genes vastly outnumber deleted genes—is addressed through balanced sampling during training, as detailed in Chapter 4 (Section 4.5.1). This balancing ensures that the model learns meaningful patterns from both deletion and retention cases rather than defaulting to predicting all genes as non-deleted.

Baseline Methods

To evaluate the effectiveness of GraphGDeI, we compare it against two baseline methods that represent different levels of architectural complexity. The first baseline (DNN) tests whether the modular design with specialized representation learning provides benefits over naive end-to-end classification. The second baseline (DeepGDeI) isolates the contribution of graph-based learning by comparing against the sequential-only predecessor.

DNN (Deep Neural Network): A deep learning-based baseline. The DNN takes as input the same tokenized and padded sequences of the gene $\mathbf{T}_g$ and metabolite $\mathbf{T}_m$ as used in our proposed framework. This baseline adopts a simplified design:

$$\mathbf{X}_g = \text{Embedding}(\mathbf{T}_g), \quad (5.24)$$

$$\mathbf{X}_m = \text{Embedding}(\mathbf{T}_m), \quad (5.25)$$

$$\mathbf{X} = [\mathbf{X}_g; \mathbf{X}_m], \quad (5.26)$$

$$\mathbf{h}_1 = \text{ReLU}(\mathbf{W}_1\mathbf{X} + \mathbf{b}_1), \quad (5.27)$$

$$\hat{y} = \sigma(\mathbf{W}_2\mathbf{h}_1 + \mathbf{b}_2), \quad (5.28)$$

where $\hat{y} = P(y = 1 \mid \mathbf{T}_g, \mathbf{T}_m; \theta)$, $y \in \{0, 1\}$ is the ground-truth label, and $\theta = \{\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}_1, \mathbf{b}_2\}$ represents learnable parameters.

DeepGDeI: Introduced in Chapter 4, this framework extends simple DNN approaches by incorporating sequential learning through LSTM-based autoencoders for both gene and metabolite representations. DeepGDeI learns metabolite and gene

representations from SMILES and AA sequences using an encoder-decoder architecture:

$$\mathbf{Z}_{\text{meta}} = \text{LSTM-Encoder}(\mathbf{T}_m), \quad (5.29)$$

$$\mathbf{Z}_{\text{gene}} = \text{LSTM-Encoder}(\mathbf{T}_g), \quad (5.30)$$

where $\mathbf{Z}_{\text{meta}}$ and $\mathbf{Z}_{\text{gene}}$ are the learned latent representations for metabolites and genes, respectively. These representations are integrated via element-wise multiplication: $\mathbf{Z} = \mathbf{Z}_{\text{meta}} \odot \mathbf{Z}_{\text{gene}}$, and passed through a fully connected layer for final prediction.

Evaluation Metrics

We evaluate model performance using four metrics: overall accuracy, macro-averaged precision, recall, and F1 score, along with macro-averaged AUC. Definitions follow those in Chapter 4 (Section 4.5.1). All metrics are reported as mean $\pm$ std over folds.

5.4.2 Results and Analysis

Performance Comparison with Baselines

The results of performance evaluations across the three metabolic models under study are summarized in Table 5.1.

Across all metabolic models, GraphGDel consistently outperformed the baseline methods in overall accuracy, macro-averaged precision, recall, F1 score, and AUC. Specifically, the overall accuracy of GraphGDel was 77.35%, 85.54%, and 73.22% for *e_coli_core*, *iMM904*, and *iML1515*, respectively, representing substantial improvements over DNN (63.31%, 69.28%, 60.04%) and DeepGdel (71.18%, 80.58%, 67.91%). This demonstrates the robustness of GraphGDel across models of varying scales.

Module Ablation

The results of ablation studies on learning modules across the three metabolic models under study are summarized in Table 5.2.

To assess the contributions of the learning modules, we evaluated two variants of GraphGDel: Ge-SMILES (using only Gene-M and Meta-M) and Ge-Graph (using only Gene-M and Graph-M). Across all models, relying solely on SMILES-based

Table 5.1: Performance comparison: GraphGDeI vs DeepGDeI vs DNN. Best in bold. M-Avg. = Macro-averaged.

Metric	e_coli_core			iMM904			iML1515		
	DNN	DeepGDeI	GraphGDeI	DNN	DeepGDeI	GraphGDeI	DNN	DeepGDeI	GraphGDeI
Overall Accuracy (%)	63.31	71.18	77.35	69.28	80.58	85.54	60.04	67.91	73.22
M-Avg. Precision (%)	64.99	72.44	75.99	72.07	77.51	83.19	64.60	68.90	74.67
M-Avg. Recall (%)	64.13	72.87	75.38	68.52	77.85	82.91	61.80	64.40	72.51
M-Avg. F1 Score (%)	59.59	70.85	74.45	64.84	76.48	81.87	58.69	63.22	72.31
M-Avg. AUC (%)	62.66	71.16	76.56	66.50	80.24	84.24	60.30	65.78	73.17

Table 5.2: Ablation study: Ge-SMILES vs Ge-Graph vs GraphGDeI. Best in bold.

Metric	e_coli_core		iMM904		iML1515	
	Ge-SMILES	Ge-Graph	Ge-SMILES	Ge-Graph	Ge-SMILES	Ge-Graph
Overall Accuracy (%)	70.34	74.33	78.76	83.33	67.11	70.35
M-Avg. Precision (%)	70.34	72.72	78.34	81.74	68.45	71.49
M-Avg. Recall (%)	71.95	72.31	77.29	81.36	63.23	69.30
M-Avg. F1 Score (%)	70.23	71.24	77.44	80.93	62.13	69.45
M-Avg. AUC (%)	70.37	73.47	78.49	82.42	68.35	70.44

or graph-based metabolite features in combination with gene features consistently resulted in reduced performance. For example, in *e_coli_core*, the overall accuracy dropped from 77.35% (full model) to 70.34% (Ge-SMILES) and 74.33% (Ge-Graph). These results highlight that integrating both sequential and graph-based metabolite representations is critical for optimal performance.

Graph Representation Evaluation

The results of graph representation evaluations across the three metabolic models under study are summarized in Table 5.3.

We further examined the impact of different graph representations generated at early stages of the pipeline ($\mathcal{G}_0$ and $\mathcal{G}_1$). Across all metabolic models, using the early-stage representations resulted in decreased performance compared to the proposed graph representation. For instance, in *iMM904*, the overall accuracy was 82.02% for $\mathcal{G}_0$, 84.70% for $\mathcal{G}_1$, and 85.54% for the proposed representation. This progressive improvement validates the necessity of each refinement step in the graph construction pipeline, demonstrating that both degree-based filtering and currency metabolite removal contribute essential topological simplifications.

Overall, the experimental results confirm that GraphGDeI effectively integrates sequential and graph-based representations, leveraging the constructed graph to enhance gene deletion strategy prediction. The improvements are consistent across

Table 5.3: Performance with different graph representations. Best in bold.

Metric	e_coli_core			iMM904			iML1515		
	$\mathcal{G}_0$	$\mathcal{G}_1$	Full	$\mathcal{G}_0$	$\mathcal{G}_1$	Full	$\mathcal{G}_0$	$\mathcal{G}_1$	Full
Overall Accuracy (%)	73.25	76.20	77.35	82.02	84.70	85.54	70.15	72.80	73.22
M-Avg. Precision (%)	73.10	75.85	75.99	80.33	83.05	83.19	70.88	73.92	74.67
M-Avg. Recall (%)	73.55	75.45	75.38	81.01	83.10	82.91	68.42	72.51	72.19
M-Avg. F1 Score (%)	72.67	74.40	74.45	80.57	81.75	81.87	68.09	72.20	72.31
M-Avg. AUC (%)	73.12	76.10	76.56	81.55	83.70	84.24	69.67	72.85	73.17

models of different sizes, indicating the generalizability and robustness of the framework.

5.5 Discussion

This chapter introduced GraphGDel, a framework that extends DeepGDel by incorporating graph-based metabolite modeling to capture network topology that is essential for understanding metabolic coupling. The experimental results demonstrate the effectiveness of this approach, with GraphGDel consistently outperforming both DNN and DeepGDel baselines across all three metabolic models. The improvements of 6.17%, 4.96%, and 5.31% over DeepGDel across *e_coli_core*, *iMM904*, and *iML1515*, respectively, indicate that structural information provides meaningful additional signal beyond sequential representations alone.

The ablation studies reveal a key insight: sequential and graph-based features are complementary rather than redundant. Removing either SMILES-based metabolite features or graph-based features leads to consistent performance degradation across all models, demonstrating that each modality captures distinct aspects of metabolite characteristics. Furthermore, the graph representation evaluation experiments validate the design choices in our construction pipeline. The progressive performance improvement from $\mathcal{G}_0$ to $\mathcal{G}_1$ to the final representation indicates that both degree-based filtering and currency metabolite removal contribute essential topological simplifications, removing confounding connectivity while preserving biologically meaningful structure.

The incorporation of the Graph-M module represents a fundamental departure from previous approaches that relied solely on sequential data. By leveraging metabolic network topology, GraphGDel captures relationships between metabolites that are not apparent from their sequential representations alone. This structural awareness is particularly valuable for understanding how metabolites interact across

pathways, which is essential for predicting gene deletion strategies that achieve growth-coupled production. However, the current pipeline relies on hand-crafted refinement rules for graph construction, which may not generalize optimally to all metabolic contexts. Additionally, the use of LSTM autoencoders and Graph Convolutional Networks represents a conservative architectural choice chosen for framework feasibility and benchmarking. The modular design of GraphGDeI enables future integration of more advanced architectures, such as transformer-based models for sequential data or heterogeneous GNNs for multi-entity graphs, potentially leading to further performance gains.

The interpretability of graph embeddings remains limited, as in many deep learning approaches. While the graph construction pipeline offers some transparency through its explicit filtering steps, the learned representations in Graph-M are not easily explainable in biological terms. This interpretability gap represents a practical limitation for applications where mechanistic understanding is as important as predictive accuracy.

Chapter 6

Conclusion and Future Directions

This chapter synthesizes the contributions of this dissertation, reflecting critically on the methodological evolution from optimization to learning, comparing the three frameworks across multiple dimensions, discussing biological and computational insights gained during this research, honestly assessing limitations, and outlining realistic future research directions.

6.1 Summary of Contributions

This dissertation presented a progression of three computational frameworks for designing gene deletion strategies to achieve growth-coupled production in genome-scale metabolic models, representing a gradual shift from optimization-based approaches toward data-driven methodologies.

Chapter 3: DBgDel introduced the integration of database knowledge into gene deletion calculation algorithms, to our knowledge the first attempt to bridge computational gene deletion databases with downstream optimization. By extracting core metabolic components from MetNetComp and constraining the search space, DBgDel achieved a 6.1-fold computational acceleration on average while maintaining competitive success rates (24/40, 81/700, 507/991) across three genome-scale

models. This work demonstrated that database-derived prior knowledge can be systematically incorporated to improve computational efficiency without substantial quality trade-offs.

Chapter 4: DeepGDel reformulated gene deletion strategy design as a supervised machine learning prediction task, proposing what we believe is the first data-driven framework for this problem. DeepGDel jointly learned from SMILES and amino acid sequences using LSTM-based autoencoders, achieving accuracy improvements of 14.69%, 22.52%, and 13.03% over baseline methods. This framework demonstrated the feasibility of fully automated prediction without explicit network simulation or optimization, marking a conceptual shift from calculation to prediction.

Chapter 5: GraphGDel extended DeepGDel by incorporating metabolic network topology through graph representations. A four-step pipeline constructed biologically meaningful metabolite graphs from algebraic GEM formulations, addressing currency metabolite challenges. GraphGDel integrated sequential and graph-based learning using GCNs, achieving the best overall performance among all evaluated methods, with further gains of 2-5% over DeepGDel on large-scale models.

6.2 Methodological Evolution: From Optimization to Learning

The progression from DBgDel through DeepGDel to GraphGDel reflects not merely incremental improvements, but a fundamental evolution in how we conceptualize and approach gene deletion strategy design. This evolution reveals important insights about the nature of the problem itself.

The Optimization-to-Learning Transition

The most significant transition occurred between DBgDel and DeepGDel, marking a shift from enhancing existing optimization algorithms to creating entirely new prediction paradigms. DBgDel demonstrated that database knowledge can accelerate optimization, but it remained fundamentally tied to computational constraints: it still required MILP-based optimization, hand-crafted extraction rules, and explicit network simulation. DeepGDel broke from this paradigm by treating gene deletion strategy design as a pure prediction problem, where the goal is not to “calculate optimal deletions” but to “learn deletion patterns from data.”

This transition was motivated by several observations. First, the MetNetComp database contains over 85,000 strategies, suggesting that there may be learnable patterns in how genes are deleted across different metabolites. Second, existing

optimization-based approaches, while mathematically rigorous, are computationally expensive and scale poorly to genome-scale models. Third, the success of deep learning in related domains involving sequential biological data (protein sequences, molecular structures) suggested that similar approaches might be applicable here.

The results confirmed this intuition, but with important caveats. While DeepGDel achieved substantial improvements over baseline prediction methods, it remains unclear whether these improvements translate to “better” gene deletion strategies in a biological sense, as evaluation was performed computationally against existing databases rather than through wet-lab validation. The black-box nature of deep learning models also introduces interpretability challenges that did not exist in optimization-based approaches.

Adding Structural Awareness

The progression from DeepGDel to GraphGDel addressed a key limitation: the former operated on sequential representations only, ignoring the inherently networked structure of metabolic systems. GraphGDel’s contribution lies not just in performance improvements, but in demonstrating that metabolic network topology contains information that complements sequential patterns.

However, incorporating graph structure introduced its own challenges. Genome-scale metabolic models are not naturally graph-structured; they are represented algebraically through stoichiometric matrices, GPR rules, and flux bounds. Constructing biologically meaningful graphs required careful refinement to handle currency metabolites and topological noise, problems that remain fundamentally unsolved in general. The four-step pipeline we propose is one approach, but it relies on hand-crafted rules and domain knowledge that may not generalize to all contexts.

Insights About the Problem

Through these three frameworks, we gained insights about gene deletion strategy design that extend beyond the specific methods we developed.

First, we learned that this problem is amenable to both optimization and learning approaches, but they serve different purposes. DBgDel’s optimization-based approach provides mathematically provable properties and guarantees, while DeepGDel and GraphGDel provide speed and generalizability from data. Neither is strictly “better”—their appropriateness depends on context, data availability, and computational constraints.

Second, we learned that the choice of data representation matters critically. Sequential representations (SMILES, amino acids) capture intrinsic molecular properties, while graph representations capture relational properties. Both are valuable, and combining them yields superior performance. However, we also discovered that not all graph representations are created equal: naive graph projections from metabolic networks produce dense, noisy structures that hinder learning, while carefully refined graphs enable effective learning but require substantial engineering effort.

Third, we learned that modular architecture design matters for both interpretability and extensibility. The modular design of DeepGDeI and GraphGDeI allowed us to systematically evaluate contributions of different data modalities through ablation studies. This design also facilitates future improvements: individual modules can be replaced or enhanced without redesigning the entire framework.

6.3 Limitations and Open Challenges

Honest assessment of limitations is crucial for understanding the scope and significance of this work. The three frameworks share several fundamental constraints while each facing distinct challenges.

Data Dependency and Generalization

All frameworks depend critically on database quality and coverage, but in different ways. DBgDeI requires comprehensive strategy coverage to extract useful constraints, failing when database patterns are sparse or exhibit high diversity (witness the success rate drop from 60.0% in *e_coli_core* to 11.6% in *iMM904*). DeepGDeI and GraphGDeI face a “cold start” problem: applying them to new organisms requires substantial training data that may not exist. This data dependency limits generalizability to emerging model organisms or newly characterized metabolites. Additionally, all methods are subject to biases and errors in their training or reference data. If strategy data contain systematic errors or suboptimal computations, both optimization-enhanced and learning-based approaches will perpetuate these limitations.

Static Representations and Simplified Problem Formulation

All frameworks operate on static metabolic representations and ignore dynamic behavior, regulation, and context-dependent gene expression. This may limit applicability to realistic strain design scenarios where temporal dynamics and regulatory responses matter. Moreover, our problem formulation assumes growth-coupled

production is binary, ignoring that it may exist on a continuous spectrum. We also abstract away critical real-world considerations: production rate, medium complexity, pathway stability, multi-product requirements, and regulatory effects. These simplifications make evaluation tractable but limit practical relevance.

Evaluation and Validation Gaps

Finally, evaluation is fundamentally incomplete across all frameworks. Computational validation, which uses success rates or prediction accuracies against existing databases, is necessary but insufficient. The ultimate validation requires implementing predicted strategies in wet-lab experiments and measuring growth-coupled production, which remains largely absent. We cannot know whether our methods produce strategies that work in practice without experimental validation, yet such validation poses substantial challenges: strategies may involve dozens or hundreds of genes, making comprehensive testing infeasible.

6.4 Future Directions

Based on the contributions, insights, and limitations identified throughout this dissertation, several promising directions for future research emerge. These directions range from immediate extensions to longer-term paradigm shifts, each grounded in specific gaps identified during this research.

Architectural Improvements and Model Selection

The current frameworks employ relatively classical deep learning architectures (LSTM autoencoders, GCNs) as default choices, selected to demonstrate feasibility and benchmark performance. However, recent advances in transformer-based architectures and specialized protein/molecular language models (ESM [67], ProtBERT [68], ChemBERTa [69]) have shown superior performance in representation learning tasks. A natural next step is to systematically benchmark these architectures within the modular framework we established. Key questions include: Do the performance improvements justify the increased computational cost? Do they generalize better across organisms?

Similarly, more sophisticated graph neural network architectures—such as Graph Attention Networks (GATs), Graph Transformers, or Heterogeneous GNNs—could potentially capture more nuanced metabolite interactions. However, we should also consider whether the added complexity is necessary. Our ablation studies showed that simple graph representations already provide substantial improvements; more complex models may not offer commensurate gains.

Addressing the Cold Start Problem

A critical limitation of learning-based approaches is their dependence on training data, which creates a “cold start” problem when applying methods to new organisms or models. Several promising directions could address this challenge.

First, transfer learning across organisms could enable pre-training representations on well-characterized models (e.g., *E. coli*, *S. cerevisiae*) and fine-tuning for species-specific prediction. This would be particularly valuable for emerging model organisms with limited gene deletion strategy data. However, the extent to which metabolic deletion patterns transfer across species remains an open empirical question.

Second, semi-supervised or few-shot learning approaches could leverage unlabeled metabolic data (networks, sequences) alongside limited labeled deletion strategies. This could reduce the data requirements for training effective models.

Third, active learning strategies could guide experimental data collection to maximize information gain for model improvement. By identifying high-uncertainty predictions, these strategies could prioritize which gene deletion experiments would most efficiently improve model performance.

Toward Interpretable and Explainable Models

The black-box nature of deep learning models limits their trust and adoption in biological applications. Addressing this requires both technical development and methodological innovation.

Technical approaches include integrating attention mechanisms to visualize which sequence positions or metabolites drive predictions, computing SHAP values to quantify feature contributions, and generating gradient-based saliency maps to identify critical pathway interactions. However, these techniques provide only approximate explanations and may not reveal true mechanistic understanding.

A more fundamental direction is developing inherently interpretable architectures. For example, graph neural networks that enforce biologically-motivated constraints (e.g., pathway modularity, reaction stoichiometry) could produce predictions that align with established metabolic principles. Similarly, hybrid models that combine learned representations with symbolic reasoning could provide explanations grounded in biochemical knowledge.

Graph Construction and Refinement

The current graph construction pipeline addresses currency metabolite challenges through hand-crafted rules, but optimal refinement strategies remain unclear

and may be context-dependent. Several directions could improve this.

First, learning-based refinement could optimize graph construction parameters jointly with downstream prediction objectives, rather than relying on heuristics. This could discover graphs that are more informative for prediction tasks.

Second, heterogeneous graphs incorporating multiple node types (metabolites, reactions, genes) and edge types (biochemical transformations, GPR rules, regulatory interactions) could capture richer relational information. However, designing effective message-passing schemes for such heterogeneous structures remains challenging.

Third, multi-scale graph representations could simultaneously capture both local pathway structures and global metabolic organization, addressing the observation that different aspects of metabolism operate at different scales.

Beyond Single Metabolites: Multi-Objective and Sequential Design

Current frameworks focus on designing gene deletion strategies for individual target metabolites. However, industrial biotechnology often requires multi-product strains or sequential production of multiple compounds. Extending our methods to handle these scenarios requires both methodological innovation and new problem formulations.

For multi-objective scenarios, we could incorporate Pareto-front analysis into prediction pipelines, enabling users to explore trade-offs among competing objectives (production rate, growth rate, medium complexity, pathway stability). This would require extending prediction models to output not just binary deletion decisions, but quantitative predictions of metabolic outcomes.

For sequential production scenarios, we could develop temporal models that predict how gene deletions interact over time, accounting for dynamic metabolic shifts and regulatory responses.

Wet-Lab Validation and Experimental Integration

Perhaps the most critical future direction is establishing robust validation pipelines that bridge computational predictions and wet-lab experiments. All three frameworks were evaluated computationally, but ultimate validation requires demonstrating that predicted strategies actually work in practice.

This requires not just performing experiments, but designing them strategically. Active learning approaches could identify high-uncertainty predictions for experimental prioritization. Additionally, failed predictions should inform model refinement, creating feedback loops between computation and experimentation. This

“design-build-test-learn” cycle could substantially improve model robustness.

However, we should also be realistic about validation scope. Gene deletion strategies may involve dozens or hundreds of genes, making complete experimental validation infeasible. Prioritization strategies that identify the most critical or most testable components could make validation more tractable.

6.5 Concluding Remarks

This dissertation has examined the transition from optimization-based to data-driven approaches for designing gene deletion strategies in genome-scale metabolic models. Through three frameworks—DBgDel, DeepGDel, and GraphGDel—we showed that this transition is both feasible and productive, yielding substantial improvements in computational efficiency and predictive performance.

At the same time, this work highlights several fundamental limitations and tensions that remain unresolved. The trade-off between interpretability and performance, the “cold start” problem in learning-based methods, the difficulty of constructing biologically meaningful graph representations, and the persistent gap between computational and experimental validation all represent ongoing challenges.

These challenges do not diminish the contributions of this dissertation, but they do temper any claim of a “revolutionary” advance. We have not solved the gene deletion strategy design problem. Rather, we have demonstrated that data-driven approaches can offer meaningful improvements over existing methods within well-defined contexts, and we have clarified many of the obstacles that remain.

Looking forward, the most promising direction is not to continue adding complexity to our frameworks, but to deepen our understanding of the problem itself. What makes a metabolite amenable to growth-coupled production? What properties of metabolic networks determine whether gene deletions succeed or fail? How do network structure, sequence characteristics, and deletion patterns interact?

Addressing these questions will require not only methodological innovation but also biological insight. The frameworks developed here provide tools for generating hypotheses and making predictions, but ultimately, understanding gene deletion strategy design will depend on integrating computational methods with experimental biology to uncover underlying principles.

As databases expand, models improve, and experimental validation becomes more systematic, the foundations laid in this dissertation may support more sophisticated approaches to computational strain design. Moving forward will require

honest assessment of limitations, careful evaluation of trade-offs, and integration of computational, biological, and engineering perspectives.

We hope that this dissertation contributes not only specific methods, but also a framework for thinking critically about the role of data-driven approaches in biological engineering.

References

- [1] I. Otero-Muras and P. Carbonell, “Automated engineering of synthetic metabolic pathways for efficient biomanufacturing,” *Metabolic Engineering*, vol. 63, pp. 61–80, 2021.
- [2] B. García-Jiménez, J. Torres-Bacete, and J. Nogales, “Metabolic modelling approaches for describing and engineering microbial communities,” *Computational and Structural Biotechnology Journal*, vol. 19, pp. 226–246, 2021.
- [3] C. J. Foster, L. Wang, H. V. Dinh, P. F. Suthers, and C. D. Maranas, “Building kinetic models for metabolic engineering,” *Current Opinion in Biotechnology*, vol. 67, pp. 35–41, 2021.
- [4] Y. Toya and H. Shimizu, “Flux analysis and metabolomics for systematic metabolic engineering of microorganisms,” *Biotechnology advances*, vol. 31, no. 6, pp. 818–826, 2013.
- [5] P. Pharkya and C. D. Maranas, “An optimization framework for identifying reaction activation/inhibition or elimination candidates for overproduction in microbial systems,” *Metabolic engineering*, vol. 8, no. 1, pp. 1–13, 2006.
- [6] J. D. Orth, I. Thiele, and B. Ø. Palsson, “What is flux balance analysis?,” *Nature biotechnology*, vol. 28, no. 3, pp. 245–248, 2010.
- [7] E. Simeonidis and N. D. Price, “Genome-scale modeling for metabolic engineering,” *Journal of Industrial Microbiology and Biotechnology*, vol. 42, no. 3, pp. 327–338, 2015.
- [8] D. Machado, M. J. Herrgård, and I. Rocha, “Stoichiometric representation of gene–protein–reaction associations leverages constraint-based analysis from reaction to gene-level phenotype prediction,” *PLoS computational biology*, vol. 12, no. 10, p. e1005140, 2016.
- [9] A. P. Burgard, P. Pharkya, and C. D. Maranas, “Optknock: a bilevel programming framework for identifying gene knockout strategies for microbial strain

- optimization,” *Biotechnology and bioengineering*, vol. 84, no. 6, pp. 647–657, 2003.
- [10] P. Pharkya, A. P. Burgard, and C. D. Maranas, “Optstrain: a computational framework for redesign of microbial production systems,” *Genome research*, vol. 14, no. 11, pp. 2367–2376, 2004.
- [11] P. Schneider, R. Mahadevan, and S. Klamt, “Systematizing the different notions of growth-coupled product synthesis and a single framework for computing corresponding strain designs,” *Biotechnology Journal*, vol. 16, no. 12, p. 2100236, 2021.
- [12] T. B. Alter and B. E. Ebert, “Determination of growth-coupling strategies and their underlying principles,” *BMC bioinformatics*, vol. 20, pp. 1–17, 2019.
- [13] V. Vieira, P. Maia, M. Rocha, and I. Rocha, “Comparison of pathway analysis and constraint-based methods for cell factory design,” *BMC bioinformatics*, vol. 20, pp. 1–15, 2019.
- [14] S. Ranganathan, P. F. Suthers, and C. D. Maranas, “Optforce: an optimization procedure for identifying all genetic manipulations leading to targeted over-productions,” *PLoS computational biology*, vol. 6, no. 4, p. e1000744, 2010.
- [15] L. Yang, W. R. Cluett, and R. Mahadevan, “Emilio: a fast algorithm for genome-scale strain design,” *Metabolic engineering*, vol. 13, no. 3, pp. 272–281, 2011.
- [16] G. Rockwell, N. J. Guido, and G. M. Church, “Redirector: designing cell factories by reconstructing the metabolic objective,” *PLoS computational biology*, vol. 9, no. 1, p. e1002882, 2013.
- [17] D. Egen and D. S. Lun, “Truncated branch and bound achieves efficient constraint-based genetic design,” *Bioinformatics*, vol. 28, no. 12, pp. 1619–1623, 2012.
- [18] N. E. Lewis, K. K. Hixson, T. M. Conrad, J. A. Lerman, P. Charusanti, A. D. Polpitiya, J. N. Adkins, G. Schramm, S. O. Purvine, D. Lopez-Ferrer, *et al.*, “Omic data from evolved e. coli are consistent with computed optimal growth from genome-scale models,” *Molecular systems biology*, vol. 6, no. 1, p. 390, 2010.

-
- [19] D. Gu, C. Zhang, S. Zhou, L. Wei, and Q. Hua, “Idealknock: a framework for efficiently identifying knockout strategies leading to targeted overproduction,” *Computational biology and chemistry*, vol. 61, pp. 229–237, 2016.
- [20] S. Ohno, H. Shimizu, and C. Furusawa, “Fastpros: screening of reaction knockout strategies for metabolic engineering,” *Bioinformatics*, vol. 30, no. 7, pp. 981–987, 2014.
- [21] T. Tamura, “Grid-based computational methods for the design of constraint-based parsimonious chemical reaction networks to simulate metabolite production: Gridprod,” *BMC bioinformatics*, vol. 19, pp. 1–9, 2018.
- [22] A. von Kamp and S. Klamt, “Growth-coupled overproduction is feasible for almost all metabolites in five major production organisms,” *Nature communications*, vol. 8, no. 1, p. 15956, 2017.
- [23] T. Tamura, A. Muto-Fujita, Y. Tohsato, and T. Kosaka, “Gene deletion algorithms for minimum reaction network design by mixed-integer linear programming for metabolite production in constraint-based models: gdel_minrn,” *Journal of Computational Biology*, vol. 30, no. 5, pp. 553–568, 2023.
- [24] C. T. Trinh, P. Unrean, and F. Srienc, “Minimal escherichia coli cell for the most efficient production of ethanol from hexoses and pentoses,” *Applied and environmental microbiology*, vol. 74, no. 12, pp. 3634–3643, 2008.
- [25] P. Schneider, A. von Kamp, and S. Klamt, “An extended and generalized framework for the calculation of metabolic intervention strategies based on minimal cut sets,” *PLoS computational biology*, vol. 16, no. 7, p. e1008110, 2020.
- [26] D. Banerjee, T. Eng, A. K. Lau, Y. Sasaki, B. Wang, Y. Chen, J.-P. Prahl, V. R. Singan, R. A. Herbert, Y. Liu, *et al.*, “Genome-scale metabolic rewiring improves titers rates and yields of the non-native product indigoidine at scale,” *Nature communications*, vol. 11, no. 1, p. 5385, 2020.
- [27] T. Tamura, “Metnetcomp: Database for minimal and maximal gene-deletion strategies for growth-coupled production of genome-scale metabolic networks,” *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2023.

- [28] F. Li, L. Yuan, H. Lu, G. Li, Y. Chen, M. K. Engqvist, E. J. Kerkhoven, and J. Nielsen, “Deep learning-based k cat prediction enables improved enzyme-constrained model reconstruction,” *Nature Catalysis*, vol. 5, no. 8, pp. 662–672, 2022.
- [29] Z. Costello and H. G. Martin, “A machine learning approach to predict metabolic pathway dynamics from time-series multiomics data,” *NPJ systems biology and applications*, vol. 4, no. 1, pp. 1–14, 2018.
- [30] W. Ahmad, E. Simon, S. Chithrananda, G. Grand, and B. Ramsundar, “Chemberta-2: Towards chemical foundation models,” *arXiv preprint arXiv:2209.01712*, 2022.
- [31] A. Madani, B. Krause, E. R. Greene, S. Subramanian, B. P. Mohr, J. M. Holton, J. L. Olmos Jr, C. Xiong, Z. Z. Sun, R. Socher, *et al.*, “Large language models generate functional protein sequences across diverse families,” *Nature biotechnology*, vol. 41, no. 8, pp. 1099–1106, 2023.
- [32] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [33] J. Stelling, S. Klamt, K. Bettenbrock, S. Schuster, and E. D. Gilles, “Metabolic network structure determines key aspects of functionality and regulation,” *Nature*, vol. 420, no. 6912, pp. 190–193, 2002.
- [34] Y.-P. Wang and Q.-Y. Lei, “Metabolite sensing and signaling in cell metabolism,” *Signal transduction and targeted therapy*, vol. 3, no. 1, p. 30, 2018.
- [35] H. Jeong, B. Tombor, R. Albert, Z. N. Oltvai, and A.-L. Barabási, “The large-scale organization of metabolic networks,” *Nature*, vol. 407, no. 6804, pp. 651–654, 2000.
- [36] R. Li, X. Yuan, M. Radfar, P. Marendy, W. Ni, T. J. O’Brien, and P. M. Casillas-Espinosa, “Graph signal processing, graph neural network and graph learning on biological data: a systematic review,” *IEEE Reviews in Biomedical Engineering*, vol. 16, pp. 109–135, 2021.
- [37] G. Muzio, L. O’Bray, and K. Borgwardt, “Biological network analysis with deep learning,” *Briefings in bioinformatics*, vol. 22, no. 2, pp. 1515–1530, 2021.

- [38] S. Jin, X. Zeng, F. Xia, W. Huang, and X. Liu, "Application of deep learning methods in biological networks," *Briefings in bioinformatics*, vol. 22, no. 2, pp. 1902–1917, 2021.
- [39] J. Farkas, "220; ber die theorie der einfachen ungleichungen," *Journal für die reine und angewandte Mathematik*, 1902.
- [40] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [41] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations*, 2017.
- [42] D. Weininger, "Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules," *Journal of chemical information and computer sciences*, vol. 28, no. 1, pp. 31–36, 1988.
- [43] N. H. Tran, X. Zhang, L. Xin, B. Shan, and M. Li, "De novo peptide sequencing by deep learning," *Proceedings of the National Academy of Sciences*, vol. 114, no. 31, pp. 8247–8252, 2017.
- [44] Z.-H. You, Y.-K. Lei, L. Zhu, J. Xia, and B. Wang, "Prediction of protein-protein interactions from amino acid sequences with ensemble extreme learning machines and principal component analysis," *BMC bioinformatics*, vol. 14, no. Suppl 8, p. S10, 2013.
- [45] H. ElAbd, Y. Bromberg, A. Hoarfrost, T. Lenz, A. Franke, and M. Wendorff, "Amino acid encoding for deep learning applications," *BMC bioinformatics*, vol. 21, no. 1, p. 235, 2020.
- [46] M. Kanehisa and S. Goto, "Kegg: kyoto encyclopedia of genes and genomes," *Nucleic acids research*, vol. 28, no. 1, pp. 27–30, 2000.
- [47] J. D. Orth, R. M. Fleming, and B. Ø. Palsson, "Reconstruction and use of microbial metabolic networks: the core escherichia coli metabolic model as an educational guide," *EcoSal plus*, vol. 4, no. 1, pp. 10–1128, 2010.
- [48] M. L. Mo, B. Ø. Palsson, and M. J. Herrgård, "Connecting extracellular metabolomic measurements to intracellular flux states in yeast," *BMC systems biology*, vol. 3, pp. 1–17, 2009.

- [49] J. M. Monk, C. J. Lloyd, E. Brunk, N. Mih, A. Sastry, Z. King, R. Takeuchi, W. Nomura, Z. Zhang, H. Mori, *et al.*, “iml1515, a knowledgebase that computes escherichia coli traits,” *Nature biotechnology*, vol. 35, no. 10, pp. 904–908, 2017.
- [50] D. S. Lun, G. Rockwell, N. J. Guido, M. Baym, J. A. Kelner, B. Berger, J. E. Galagan, and G. M. Church, “Large-scale identification of genetic design strategies using local search,” *molecular systems biology*, vol. 5, no. 1, p. 296, 2009.
- [51] L. Heirendt, S. Arreckx, T. Pfau, S. N. Mendoza, A. Richelle, A. Heinken, H. S. Haraldsdóttir, J. Wachowiak, S. M. Keating, V. Vlasov, *et al.*, “Creation and analysis of biochemical constraint-based models using the cobra toolbox v. 3.0,” *Nature protocols*, vol. 14, no. 3, pp. 639–702, 2019.
- [52] I. Rocha, P. Maia, M. Rocha, and E. C. Ferreira, “Optgene: a framework for in silico metabolic engineering,” 2008.
- [53] S. Klamt, J. Saez-Rodriguez, and E. D. Gilles, “Structural and functional analysis of cellular networks with cellnetanalyzer,” *BMC systems biology*, vol. 1, pp. 1–13, 2007.
- [54] N. Tepper and T. Shlomi, “Predicting metabolic engineering knockout strategies for chemical production: accounting for competing pathways,” *Bioinformatics*, vol. 26, no. 4, pp. 536–543, 2010.
- [55] T. Tamura, “Trimming gene deletion strategies for growth-coupled production in constraint-based metabolic networks: Trimgdel,” *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 20, no. 2, pp. 1540–1549, 2023.
- [56] Z. Yang and T. Tamura, “DBgDel: Database-Enhanced Gene Deletion Framework for Growth-Coupled Production in Genome-Scale Metabolic Models,” *IEEE Transactions on Computational Biology and Bioinformatics*, vol. 22, no. 04, pp. 1415–1427, 2025.
- [57] B. Sattarov, I. I. Baskin, D. Horvath, G. Marcou, E. J. Bjerrum, and A. Varnek, “De novo molecular design by combining deep autoencoder recurrent neural networks with generative topographic mapping,” *Journal of chemical information and modeling*, vol. 59, no. 3, pp. 1182–1196, 2019.

-
- [58] K. Abbasi, P. Razzaghi, A. Poso, M. Amanlou, J. B. Ghasemi, and A. Masoudi-Nejad, “Deepcda: deep cross-domain compound–protein affinity prediction through lstm and convolutional neural networks,” *Bioinformatics*, vol. 36, no. 17, pp. 4633–4642, 2020.
- [59] S. K. Sønderby, C. K. Sønderby, H. Nielsen, and O. Winther, “Convolutional lstm networks for subcellular localization of proteins,” in *Algorithms for Computational Biology: Second International Conference, AlCoB 2015, Mexico City, Mexico, August 4-5, 2015, Proceedings 2*, pp. 68–80, Springer, 2015.
- [60] A. Paszke, “Pytorch: An imperative style, high-performance deep learning library,” *arXiv preprint arXiv:1912.01703*, 2019.
- [61] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” *Advances in neural information processing systems*, vol. 30, 2017.
- [62] Z. Yang and T. Tamura, “DeepGDeI: Deep Learning-Based Gene Deletion Prediction Framework for Growth-Coupled Production in Genome-Scale Metabolic Models,” *IEEE Transactions on Computational Biology and Bioinformatics*, vol. 22, no. 05, pp. 2252–2266, 2025.
- [63] T. Pfeiffer, O. S. Soyer, and S. Bonhoeffer, “The evolution of connectivity in metabolic networks,” *PLoS biology*, vol. 3, no. 7, p. e228, 2005.
- [64] Z. A. King, J. Lu, A. Dräger, P. Miller, S. Federowicz, J. A. Lerman, A. Ebrahim, B. O. Palsson, and N. E. Lewis, “Bigg models: A platform for integrating, standardizing and sharing genome-scale models,” *Nucleic acids research*, vol. 44, no. D1, pp. D515–D522, 2016.
- [65] S. Moretti, V. D. T. Tran, F. Mehl, M. Ibberson, and M. Pagni, “Metanetx/m-nxref: unified namespace for metabolites and biochemical reactions in the context of metabolic models,” *Nucleic acids research*, vol. 49, no. D1, pp. D570–D574, 2021.
- [66] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” *Advances in neural information processing systems*, vol. 27, 2014.

-
- [67] Z. Lin, H. Akin, R. Rao, B. Hie, Z. Zhu, W. Lu, N. Smetanin, R. Verkuil, O. Kabeli, Y. Shmueli, *et al.*, “Evolutionary-scale prediction of atomic-level protein structure with a language model,” *Science*, vol. 379, no. 6637, pp. 1123–1130, 2023.
- [68] N. Brandes, D. Ofer, Y. Peleg, N. Rappoport, and M. Linial, “Proteinbert: a universal deep-learning model of protein sequence and function,” *Bioinformatics*, vol. 38, no. 8, pp. 2102–2110, 2022.
- [69] S. Chithrananda, G. Grand, and B. Ramsundar, “Chemberta: large-scale self-supervised pretraining for molecular property prediction,” *arXiv preprint arXiv:2010.09885*, 2020.

List of Main Publications

- **[Chapter 3]:** Ziwei Yang and Takeyuki Tamura (2025). DBgDel: Database-enhanced Gene Deletion Framework for Growth-Coupled Production in Genome-Scale Metabolic Models. IEEE Transactions on Computational Biology and Bioinformatics, vol.22, no.4(2025), pp.1415-1427.
- **[Chapter 4]:** Ziwei Yang and Takeyuki Tamura (2025). DeepGDel: Deep Learning-based Gene Deletion Prediction Framework for Growth-Coupled Production in Genome-Scale Metabolic Models. IEEE Transactions on Computational Biology and Bioinformatics, vol.22, no.5(2025), pp.2252-2266.
- **[Chapter 5]:** Ziwei Yang and Takeyuki Tamura (2025). GraphGDel: Constructing and Learning Graph Representations of Genome-Scale Metabolic Models for Growth-Coupled Gene Deletion Prediction. arXiv preprint.

Note: This thesis is based on two of the above papers published via © 2025 IEEE. Permission to reuse the contents is governed by IEEE copyright policies:

[https://journals.ieeeauthorcenter.
ieee.org/choose-a-publishing-agreement/
avoid-infringement-upon-ieee-copyright/](https://journals.ieeeauthorcenter.ieee.org/choose-a-publishing-agreement/avoid-infringement-upon-ieee-copyright/)

IEEE Copyright Notice

In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of university or educational entity's products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to: http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink. If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

List of Other Publications

Note: This is a list of selected publications that are not relevant to the thesis.

- Ziwei Yang, Rikuto Kotoge, Xihao Piao, Zheng Chen, Lingwei Zhu, Peng Gao, Yasuko Matsubara, Yasushi Sakurai, Jimeng Sun. MLOmics: Cancer Multi-Omics Database for Machine Learning. *Nature Scientific Data*. 12(2025)913-913.
- Ziwei Yang, Zheng Chen, Xin Liu, Rikuto Kotoge, Peng Chen, Yasuko Matsubara, Yasushi Sakurai, Jimeng Sun. GeSubNet: Gene Interaction Inference for Disease Subtype Network Generation. *The Thirteenth International Conference on Learning Representations (ICLR 2025)*, 67683-67708.
- Ziwei Yang, Zheng Chen, Yasuko Matsubara, Yasushi Sakurai. MoCLIM: Towards Accurate Cancer Subtyping via Multi-omics Contrastive Learning with Omics-inference Modeling. *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (IJCAI 2023)*, 2895-2905.
- Ziwei Yang, Lingwei Zhu, Chen Li, Zheng Chen, Naoki Ono, Md Altaf-Ul-Amin, Shigehiko Kanaya. Hierarchical Categorical Generative Modeling for Multi-omics Cancer Subtyping. *2022 IEEE International Conference on Bioinformatics and Biomedicine (BIBM 2022)*, 1350-1355.
- Ziwei Yang, Lingwei Zhu, Zheng Chen, Ming Huang, Naoaki Ono, MD Altaf-Ul-Amin, Shigehiko Kanaya. Cancer Subtyping via Embedded Unsupervised Learning on Transcriptomics Data. *2022 44th Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC 2022)*, 1113-1116.
- Ziwei Yang, Dong Wang, Zheng Chen, Ming Huang, Naoaki Ono, Md Altaf-Ul-Amin, Shigehiko Kanaya. Exploring Feasibility of Truth-involved Automatic Sleep Staging Combined with Transformer. *2021 IEEE International Conference on Bioinformatics and Biomedicine (BIBM 2021)*, 2920-2923.