UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

ORIENTING BIOCHEMICAL REACTIONS IN METABOLIC PATHWAYS AND NETWORKS

ZEPHYR SERRET VERBIST

Thesis supervisor: GABRIEL VALIENTE FERUGLIO (Department of Computer Science)

Degree: Bachelor Degree in Informatics Engineering (Computing)

Bachelor's thesis

Facultat d'Informàtica de Barcelona (FIB)

Universitat Politècnica de Catalunya (UPC) - BarcelonaTech

Table of contents

1 Introduction and contextualization 4
1.1 Terms and concepts used in this study 4
1.1.1 Hypergraphs 4
1.1.2 Metabolic Pathway 4
1.1.3 Internal and external vertices 5
1.2 Problem definition 6
1.3 Stakeholders 6

2 Justification 6

3 Scope 7
3.1 Objective 7
3.2 Sub-objectives 7
3.3 Identification of Functional and Non-Functional Requirements: 7
3.3.1 Functional requirements 7
3.3.2 Non-Functional Requirements: 8
3.4 Risk assessment 8

4 Methodology: Kanban for Solo Development 8
4.1 Adapting Kanban for Solo Development: 9
4.2 Benefits of Kanban for Solo Development: 9

5 Time planning 10
5.1 Task descriptions 10
5.1.1 T1 - Project Management 10
5.1.2 T2 - Design of the ILP Model 10
5.1.3 T3 - App Development: 11
5.1.4 T4 - Implicit Task - Project Documentation: 11
5.1.5 T5 - Preparation for the oral defence 11
5.2 Task duration estimates 11
5.3 Resources 14
5.3.1 Human Resources: 14
5.3.2 Material Resources: 14
5.4 Risk Management: Obstacles and alternative plans 17
5.4.1 Project Timeline 17
5.4.2 Computational power 17
5.4.3 Integration Challenges 17
5.4.4 Inexperience with ILP: 17

6 Budget 18
6.1 Staff costs 18
6.2 Generic costs 20
6.3 Budget Deviations 20
6.3.1 Contingency 20
6.3.2 Incidental Costs 20
6.4 Management Control 21
6.4.1 Metrics for Budget Adherence: 21
6.4.2 Adaptation for Budget Control 21
6.5 Total Budget 21

7 Sustainability report **21**
7.1 Economic dimenstion 22
7.2 Environmental Dimension 22
7.3 Social Dimension 23

8 Creating an effective AMPL model **23**
8.1 Understanding the Basics of AMPL 24
8.2 Desired inputs/outputs 24
8.3 Explaining our models 25
8.3.1 First prototype 25
8.3.2 Models Developed by the Professor and His Colleagues 27
8.4 Initial Benchmarking 29
8.5 Creating and optimizing our final model 30
8.6 Mathematical proof of the final model 32
8.6.1 Introduction of Predicates and Functions: 32
8.6.2 Obtaining is_internal from has_outgoing and has_incoming 32
8.6.3 Obtaining has_outgoing from the X and Y sets 35
8.6.4 Obtaining has_incoming from the X and Y sets 36
8.6.5 Additional Constraints: forcing internal 38
8.6.6 Additional Constraints: forcing external 38
8.6.7 Additional Constraints: respecting the invertability of hyperedges . . 39
8.6.8 Comparing the mathematical correctness of Model A and Serret's models 39
8.7 Final Benchmark and conclusions 39

9 Database Integration **42**
9.1 Acknowledgment of Attempted Integration using bioservices 42
9.2 REST API Data Mapping: Needs and Retrievals 42
9.3 Equation Discovery: Unveiling Reaction Formulas and Compound IDs Across Varied Endpoints 44

10 Navigating the Design and User Experience of our Python GUI **47**
10.1 Installing and Running the GUI 47
10.2 Pathway Selector 48
10.3 Pathway Solver/Visualizer 51
10.4 Benchmark View 57

List of Figures

1 Example of a directed hypergraph $H = (V,E)$ 4
2 Sphingolipid metabolism. Source: KEGG entry hsa map00600 5
3 Gantt Diagram. Source : own compilation 13
4 The model file of the first prototype 27
5 AMPL Model - Model A 28
6 AMPL Model - Model B 29
7 Serret's DualImply Model (with extra restrictions) 31
8 Strong AMPL rules used to define has_internal$_i$ 34
9 Serret's UniImply model 40
10 Integration attempt using bioservices 42
11 Obtaining compound ids from the human readable equation 46
12 Pathway Selector - User view 48
13 Pathway Selector - Labeled Features 49
14 Show Image - Image corresponding to the entry 'map00010' 50

15 Feature tooltips - tooltip of the show image button 50
16 Warning Message on Updating Dataset 50
17 Solver/Visualizer - User View 51
18 Output for map00908 - Zeatin biosynthesis 52
19 Solver-Visualizer - Saving the result in a file 53
20 An example of how to represent a hypergraph using a graph 53
21 Solver-Visualizer - Visualizing a oriented pathway 54
22 Solver-Visualizer - Visualizing a oriented pathway, with labels toggled off . . 54
23 Solver-Visualizer - The 'Extra restrictions' menu 55
24 Solver-Visualizer - Adding items to a list of a restriction 55
25 Solver-Visualizer - Items added to a list of a restriction 56
26 Solver-Visualizer - Item removed from a list of restrictions 56
27 Benchmark View - User Interface 57
28 Benchmark View - Completed Benchmarks 57
29 Benchmark View - Expanded Benchmarks 58

List of Tables

1 Task Duration and dependencies 12
2 Table of requirements (Human and Material) corresponding to each task . . . 16
3 Estimated Staff Costs per hour (1736 working hours per year [?]) 18
4 Personnel Cost per Activity (PCA) based on the costs defined in table 3 . . . 19
5 Amortization of physical resources 20
6 Incidental Costs 21
7 Final Budget 22
8 Initial Benchmark 30
9 Truth table of the equivalence of is_internal and the system of inequations . . 33
10 Optimizing is_internal by removing a redundant inequation 34
11 Truth table of the '$\wedge$' operator and the construction $a+b=2$ 38
12 Truth table of the negated '$\wedge$' operator and the construction **toNum**(a) + **toNum**$(b)<2$ 38
13 Final Benchmark 41

1 Introduction and contextualization

This article provides context for the research conducted in Zephyr Serret Verbist's Bachelor's Thesis, undertaken at the Faculty of Informatics of Barcelona (FIB) at the Universitat Politècnica de Catalunya, under the guidance of Gabriel Valiente Feruglio. It encompasses an exploration of the research's background, the applied methodology, the rationale behind the study, and the obtained results.

1.1 Terms and concepts used in this study

1.1.1 Hypergraphs

Hypergraphs are an extension of graphs that employs hyperedges instead of edges. A hyperedge is defined by two non-empty sets of nodes: a tail set and a head set. The hyperedge models a relationship originating from all elements in its tail set to all elements in its head set, or it can be undirected.

Let us recall from [**?**] the following definition, where we impose the tail and head of a hyperedge to be nonempty sets of nodes.

Definition. *A directed hypergraph* $H=(V,E)$ *consists of a set of nodes* V *and a set of hyperedges* $E=\{\,(T(e),H(e))\ |T(e)\subseteq V, H(e)\subseteq V, T(e)\neq\emptyset, H(e)\neq\emptyset\}$.

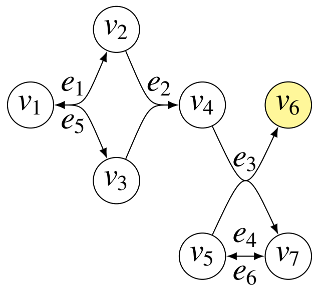


Figure 1: Example of a directed hypergraph $H=(V,E)$ with $V=\{v_1,v_2,v_3,v_4,v_5,v_6,v_7\}$ and $E=\{e_1,e_2,e_3,e_4,e_5,e_6\}$, where $e_1=(\{v_1\},\{v_2,v_3\})$, $e_2=(\{v_2,v_3\},\{v_4\})$, $e_3=(\{v_4,v_5\},\{v_6,v_7\})$, $e_4=(\{v_5\},\{v_7\})$, $e_5=(\{v_2,v_3\},\{v_1\})$, $e_6=(\{v_7\},\{v_5\})$.

1.1.2 Metabolic Pathway

A metabolic pathway is a set of linked chemical reactions that occur in the cell. Hypergraphs are particularly well-suited for modeling metabolic pathways because metabolic reactions often involve multiple substrates and multiple products [**?**]. While these reactions typically have a dominant direction, it is also possible for some reactions to proceed in the opposite direction [**?**]. Figure 2 shows a representation of the Sphingolipid metabolic pathway map in Homo Sapiens.

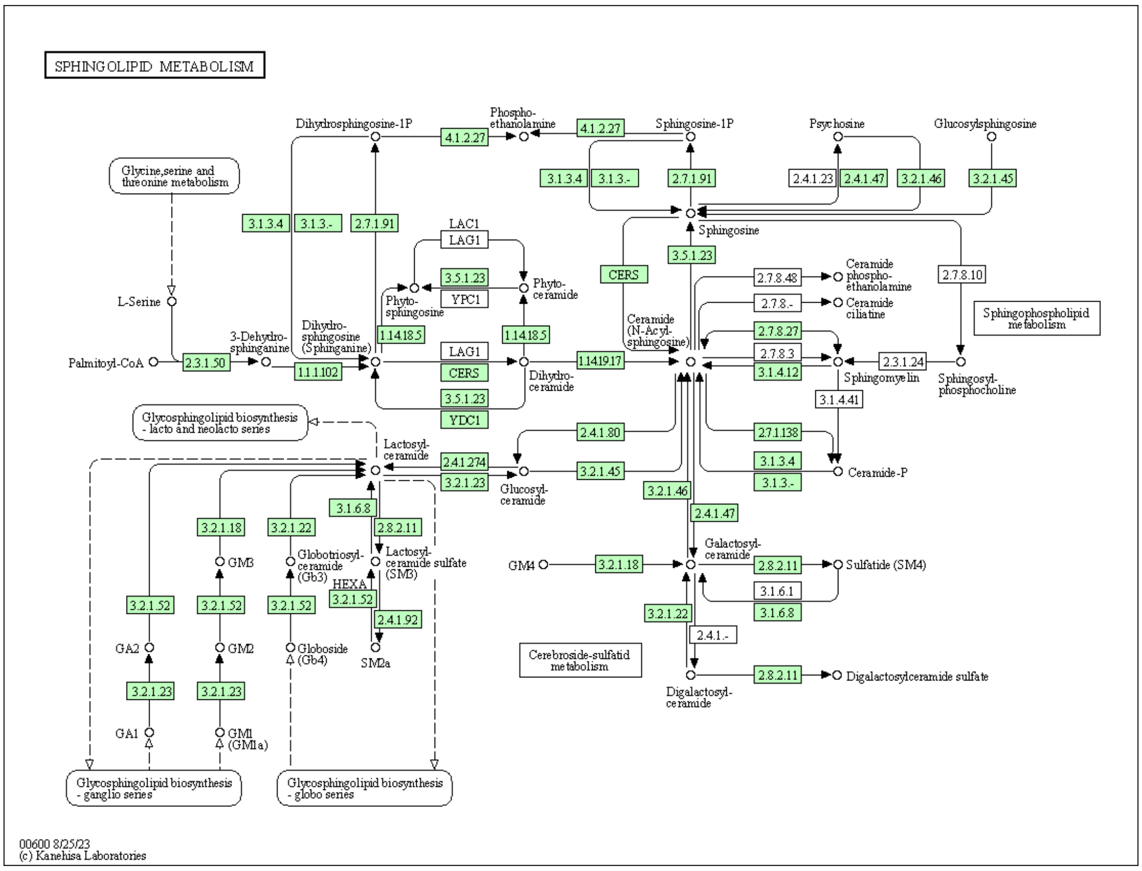


Figure 2: Sphingolipid metabolism. Source: KEGG entry hsa map00600

Throughout this study, we will utilize authentic metabolic pathways sourced from the Kyoto Encyclopedia of Genes and Genomes (KEGG) database. These real-world pathways will serve a dual purpose: firstly, to rigorously assess the effectiveness of our proposed solution, and secondly, to establish a robust benchmark against which future evaluations can be conducted. The determination of practical relevance and application of our results will be subject to evaluation by researchers in the field, ensuring the impartial assessment of our methodology within the context of real biological systems.

Using hypergraphs to model metabolic pathways allows us to capture the complexity and interdependencies inherent in cellular metabolism. This modeling approach provides a robust framework for studying and analyzing the intricate network of biochemical reactions that occur within cells, shedding light on the underlying processes crucial to understanding cellular function and regulation.

1.1.3 Internal and external vertices

In the context of directed graphs and hypergraphs, we introduce the concept of internal and external nodes. An internal node is defined by meeting two essential criteria: it must have an in-degree greater than zero, indicating incoming connections, and it must also have an out-degree greater than zero, signifying outgoing connections. Conversely, if a node fails to satisfy either of these criteria, it is considered external.

Applying this concept to a given metabolic pathway and its corresponding hypergraph, an internal node implies that the associated metabolite serves as both a substrate and a product within the metabolic pathway. In contrast, when a node is external, it indicates that the metabolite functions either as an input into the metabolic pathway or as a final product, thereby shedding light on the role of the metabolite in the broader context of the pathway's inputs and outputs.

1.2 Problem definition

In the context of undirected or partially directed hypergraphs, the objective is to determine an optimal orientation of the hyperedges that minimizes the number of external nodes. An external node is defined as a node within the hypergraph that lacks either incoming or outgoing hyperedges. To address this problem, we aim to develop an integer linear programming (ILP) model capable of finding a feasible solution within a reasonable computational timeframe.

In essence, this problem seeks to find an orientation for the hyperedges that maximizes the internal connectivity of nodes while minimizing their isolation as external nodes. The ILP model will serve as a valuable tool to achieve this goal, providing an efficient means of optimizing the orientation of hyperedges and facilitating the analysis of complex hypergraph structures.

1.3 Stakeholders

In this research project, the primary stakeholders involved are as follows:

Thesis Supervisor and Team: The thesis supervisor and their team play a pivotal role as stakeholders in this endeavor, and they are poised to benefit significantly from the study's success. Their guidance, mentorship, and valuable insights contribute significantly to the project's progress. Moreover, their team, concurrently tackling a similar problem, presents opportunities for collaboration and knowledge sharing that enrich our collective efforts. The success of this research not only advances our academic pursuits but also enhances the knowledge and expertise of our thesis supervisor and their team, positioning them at the forefront of cutting-edge research in this field.

Biologists and Researchers in Life Sciences: While the biological relevance of the project's results is still under evaluation, biologists and researchers in the field of life sciences stand to benefit from the outcomes of this research. The potential implications of the project's findings for understanding and modeling complex biological systems, particularly in the context of metabolic pathways, hold promise for these stakeholders. The insights generated by the hypergraph orientation techniques developed in this project may open new avenues for their own research and analysis, ultimately benefiting the broader scientific community.

2 Justification

In the realm of network science and analysis, the topological characterization of complex networks has been a focal point of research and investigation over the past decade. However, the analogous theory for complex hyper-networks has not seen the same level of development and attention [**?**]. Notably, there exists a critical gap in the field: the lack of a known polynomial-time algorithm for the optimal hypergraph orientation problem, and the computational complexity of this problem remains an open question.

To address this challenge, we have made the deliberate choice to employ Integer Linear Programming (ILP) as our optimization technique. ILP offers a powerful and versatile framework for tackling complex combinatorial problems. Given the uncharted terrain surrounding hypergraph orientation, the adaptability and precision of ILP provides a promising avenue for addressing this computational challenge and advancing our understanding of complex hyper-networks. Through ILP, we aim to explore and optimize hypergraph orientations, thereby contributing to the development of solutions in this less-explored but highly significant area of network theory.

3 Scope

3.1 Objective

The overarching objective of this research project is to develop an Integer Linear Programming (ILP) model for optimizing the orientation of hyperedges within undirected or partially directed hypergraphs. Additionally, we aim to create a user-friendly software application that seamlessly connects to the Kyoto Encyclopedia of Genes and Genomes (KEGG) database. This application will retrieve metabolic pathway entries from KEGG and interface with the ILP model to assess its efficacy in minimizing external nodes. The combined goal is to provide a powerful and practical tool for researchers to analyze and optimize hypergraph orientations within the context of real-world biological pathways.

3.2 Sub-objectives

1. **Mathematical Model Formulation:** Define and refine the ILP model, taking into account hypergraph structures and orientation optimization criteria. This includes specifying decision variables, constraints, and the objective function.
2. **Algorithm Development:** Implement the ILP model as a computational algorithm capable of optimizing hypergraph orientations efficiently and accurately.
3. **Database Integration:** Create a module within the software application to connect to the KEGG database, retrieve metabolic pathway data, and format it for use in the ILP model.
4. **User Interface Design:** Develop an intuitive and user-friendly interface for the application, enabling researchers to interact with the ILP model and KEGG database seamlessly.
5. **Application Development:** Build the software application, integrating the ILP algorithm, database connectivity, and user interface components into a cohesive tool.
6. **Testing and Validation:** Rigorously test the ILP model and the application using synthetic and real-world hypergraphs and metabolic pathway data from KEGG. Validate results against ground truth data where available.
7. **Performance Optimization:** Optimize the ILP model and application for computational efficiency and scalability, ensuring it can handle large-scale hypergraphs and databases. This includes benchmarking multiple ILP models and solvers to identify the best configuration in terms of speed and accuracy.
8. **Documentation and User Guides:** Prepare comprehensive documentation and user guides for the application, making it accessible to a wide range of researchers and practitioners.
9. **Deployment and Accessibility:** Make the software application accessible to the research community through appropriate channels, ensuring it is readily available for use.

3.3 Identification of Functional and Non-Functional Requirements:

In addition to the primary objectives and subtasks, it is imperative to identify both functional and non-functional requirements to guide the development and evaluation of the ILP model and application:

3.3.1 Functional requirements

1. **Data Retrieval:** The application must seamlessly retrieve metabolic pathway data from the KEGG database, allowing users to specify search criteria and obtain relevant results.
2. **Hypergraph Orientation:** The ILP model should accurately optimize hypergraph orientation, minimizing the number of external nodes while preserving the integrity of the data.

3. **User Interface:**** The user interface should provide an intuitive and interactive platform for users to input hypergraphs, set parameters, and visualize results.
4. **Performance:** The application should execute efficiently, providing timely results for both small and large-scale hypergraphs. It should also handle multiple ILP models and solvers for benchmarking.
5. **Result Presentation:** The application should present orientation results in a clear and interpretable manner, enabling users to understand the impact of the orientation on the hypergraph.

3.3.2 Non-Functional Requirements:

1. **Usability:** The user interface should be user-friendly, with clear instructions and an intuitive design to accommodate users with varying levels of expertise.
2. **Maintainability:** Develop the application and ILP model using good programming practices, allowing for updates, bug fixes, and future enhancements.
3. **Scalability:** Ensure that the ILP model and application can scale to accommodate the growing volume and complexity of hypergraphs and databases.
4. **Documentation:** Provide comprehensive documentation, including user guides, technical manuals, and developer resources to support users and maintainers.

3.4 Risk assessment

In the pursuit of this research project, several significant risks have been identified, each with the potential to impact the successful completion of the objectives. Mitigating these risks is essential to maintain the project's timeline and ensure the achievement of its goals. The following risks have been identified as particularly relevant:

1. **Project Timeline:** Unforeseen issues, delays in data acquisition, or unexpected complexities may pose a risk to the project's timeline and the timely achievement of milestones.
   Mitigation: Develop a realistic project timeline, monitor progress regularly, and have contingency plans in place to address any delays or setbacks that may arise.
2. **Computational Power:** The computational demands of solving complex ILP models, especially for large hypergraphs, may exceed available computational resources, potentially affecting the project's efficiency.
   Mitigation: Get access to the computing cluster of the Department of Computer Science.
3. **Integration Challenges:** Developing an application that seamlessly interfaces with the KEGG database may encounter challenges related to database structure changes, API updates, or connectivity issues.
   Mitigation: Stay informed about KEGG updates, establish communication channels with the KEGG team if possible, and have a plan in place to adapt the application to accommodate changes as they occur.
4. **Inexperience with ILP:** Limited prior experience or familiarity with Integer Linear Programming (ILP) modeling may present a challenge in developing and implementing effective ILP models for hypergraph orientation.

4 Methodology: Kanban for Solo Development

For this research project, the Kanban methodology has been selected as the ideal approach to guide the development process.

4.1 Adapting Kanban for Solo Development:

In the absence of changing client priorities, Kanban will be customized to align with the solo development setting:

Visual Task Management:

For efficient task management and to implement the Kanban methodology, I will leverage the Trello platform as my primary tool. Trello's user-friendly interface and visual board system make it well-suited for organizing and tracking the progress of research tasks. Within Trello, I will create dedicated boards that represent various project stages, such as "To-Do", "In Progress", and "Completed". Each task or research activity will be represented as a card within the corresponding board column. This approach will provide a clear, real-time visualization of the project's workflow and enable me to maintain a structured and organized development process throughout the project's lifecycle.

Work in Progress (WIP) Limits:

Kanban's WIP limits will be employed to control the number of concurrent tasks in progress. By setting limits on work items, the development process is streamlined, preventing overloading and maintaining a manageable pace of work.

Continuous Monitoring:

Scheduled bi-weekly meetings with the thesis supervisor will serve as checkpoints for evaluating progress and addressing challenges. In addition, I will maintain consistent email communication with the supervisor. To enhance transparency and provide real-time access to project progress, all code will be hosted on a public Git repository. This repository offers version control, data backup in case of hardware failure, and potential future collaboration. It will feature a single main branch for streamlined development and marked releases to signify project milestones.

Feedback and Iteration:

A fundamental principle of Kanban is continuous improvement. Feedback obtained from supervisor meetings and self-assessment will guide adjustments to the project plan. This iterative approach allows for refinements in task prioritization and resource allocation as the project evolves.

4.2 Benefits of Kanban for Solo Development:

Efficiency: Kanban promotes efficient task management, ensuring that effort is directed toward high-priority items.

Visibility: The visual nature of Kanban provides a clear overview of project progress and pending tasks.

Adaptability: Kanban's flexibility allows for seamless adjustments to the project plan based on evolving needs or new insights.

Reduced Overhead: As a solo developer, Kanban minimizes the overhead associated with more complex project management methodologies.

5 Time planning

The project, initiated in July 2023, is anticipated to conclude between December and January. A single individual is allocated approximately 25 hours per week for project tasks, though this allocation may experience variations.

5.1 Task descriptions

In this section, we introduce the primary tasks and their corresponding subtasks, providing a brief description of each.

5.1.1 T1 - Project Management

In this initial phase, the focus will be on establishing a robust project framework.

T1.1 Definition of the context and project scope: This phase involves clearly defining the context in which the project operates and establishing the boundaries of its scope. It includes identifying the key stakeholders, understanding the project's objectives, and delineating the specific problems or challenges it aims to address. This foundational step sets the direction for the entire project.

T1.2 Temporal planning of the project: Temporal planning focuses on creating a comprehensive schedule for the project, including task sequencing, resource allocation, and timelines. It involves setting realistic milestones and deadlines, taking into account potential project dependencies and risks. Effective temporal planning ensures efficient project management and helps keep the project on track.

T1.3 Budget and sustainability: This phase entails determining the financial resources required for the project's execution, including budget allocation for various tasks and activities. Additionally, it involves a study of the project's sustainability impact, assessing its environmental, social, and economic implications.

T1.4 Integration of Key Project Components: In this stage, we integrate and consolidate the key components of the project, including the defined context and scope, the finalized temporal plan, the budget allocation, and the sustainability impact assessment. This holistic approach ensures that all project elements work seamlessly together and provides a comprehensive reference for project execution.

T1.5 Meetings with the tutor: Regular meetings with the project tutor are essential for maintaining effective communication and mentorship throughout the project's lifecycle. These meetings provide opportunities to discuss progress, seek guidance, address challenges, and ensure that the project remains aligned with its objectives. Meetings with the tutor are valuable checkpoints for the project's success.

5.1.2 T2 - Design of the ILP Model

This phase constitutes the core of the project, involving the conceptualization and formulation of the Integer Linear Programming (ILP) model. It encompasses identifying variables, formulating constraints, and fine-tuning the model to accurately represent hypergraph orientation.

T2.1 Study of ILP Modeling Using AMPL: Dive into the study of Integer Linear Programming (ILP) modeling principles using AMPL (A Mathematical Programming Language) as the platform. This subtask involves gaining a deep understanding of ILP concepts, syntax, and best practices for model formulation.

T2.2 Formulate the Hypergraph Model: Develop an ILP model that accurately represents hypergraph orientation and obtains a solution to our optimization problem. This subtask includes identifying and defining the model's decision variables, objective function, and constraints. Ensure that the model effectively captures the complexities of hypergraph structures.

T2.3 Validation with Handcrafted Data: Test the formulated ILP model using carefully crafted or synthetic hypergraph data. This subtask involves creating hypergraphs with known properties to validate the model's accuracy and functionality.

T2.4 Testing with Real Data: Apply the ILP model to real-world hypergraph data, such as metabolic pathway information from the KEGG database. This subtask assesses the model's performance and adaptability in handling practical and complex data sets.

T2.5 Benchmarking and Optimization: Conduct benchmark tests to evaluate the ILP model's efficiency and scalability. Explore alternative models and solvers.

5.1.3 T3 - App Development:

Following the design of the ILP model, attention will shift towards the development of the application. This phase involves implementing the ILP model within the application framework, ensuring seamless interaction with the KEGG database, and incorporating user-friendly interfaces for efficient utilization.

T3.1 Database Interaction Setup Establish robust mechanisms for the application to interact with the KEGG database, allowing for data retrieval and updates as needed.

T3.2 ILP Model Integration Integrate the designed ILP model into the application's architecture, ensuring that it functions seamlessly within the software.

T3.3 Functionality Implementation Implement the core functionalities of the application, ensuring that it can effectively perform hypergraph orientation tasks based on the ILP model.

T3.4 User Interface Design Design and develop user-friendly interfaces for the application, focusing on usability and efficiency for end-users.

T3.5 User Documentation Creation Develop comprehensive user documentation, including user guides and manuals, to assist users in effectively utilizing the application.

5.1.4 T4 - Implicit Task - Project Documentation:

Throughout the different project tasks, comprehensive documentation is an ongoing process. This includes detailing the methodology, presenting results, and providing user instructions for the application. Additionally, any supplementary materials or resources are continuously compiled for future reference.

5.1.5 T5 - Preparation for the oral defence

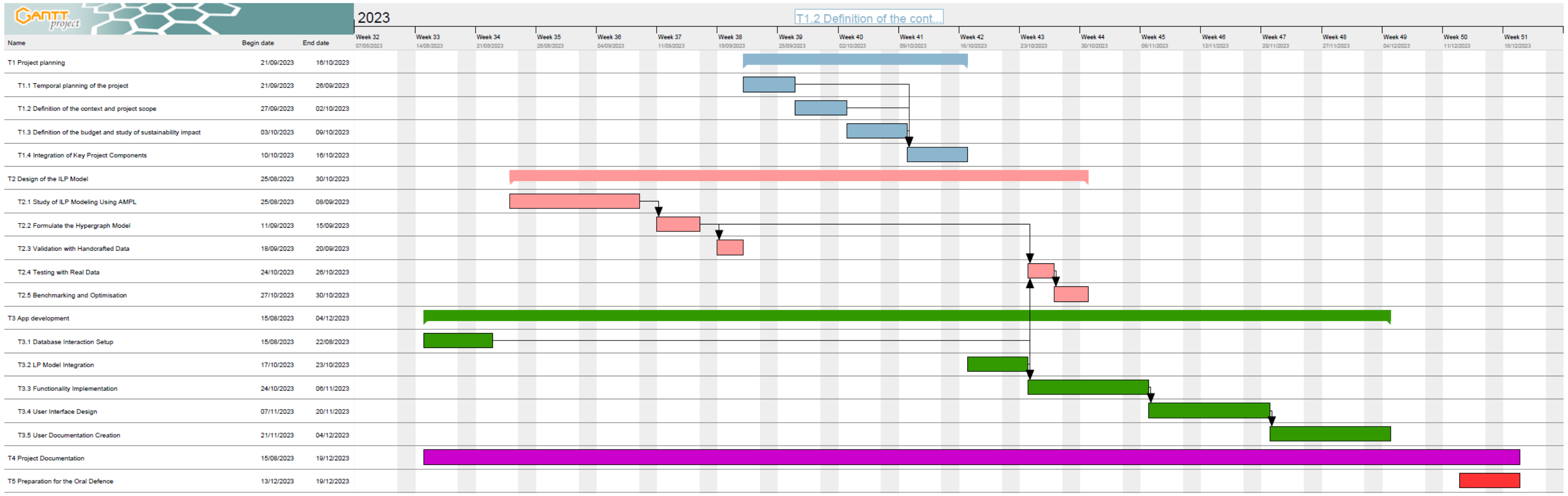
In the lead-up to the oral defense, thorough preparation is essential. This subtask includes finalizing the presentation materials, rehearsing the oral defense, and conducting mock presentations to ensure a confident and effective delivery during the defense session.

5.2 Task duration estimates

In this section we give an estimating of the time required for the completion of tasks and subtasks. Additionally, we will identify and declare any interdependencies between these tasks where applicable. We provide the table 1 which displays the durations and dependencies of the tasks. We also provide the figure 3 which depicts the Gantt Chart representing the proposed workflow of the project.

ID	Task	Dependencies	Duration (hours)
T1	Project Management		85
T1.1	Definition of the context and project scope		25
T1.2	Temporal planning of the project		15
T1.3	Budget and sustainability		15
T1.4	Integration of Key Project Components	T1.1, T1.2, T1.3	15
T1.5	Meetings with the tutor		15
T2	Design of the ILP Model		90
T2.1	Study of ILP Modeling Using AMPL		40
T2.2	Formulate the Hypergraph Model	T2.1	15
T2.3	Validation with Handcrafted Data	T2.2	10
T2.4	Testing with Real Data	T2.2, T3.1	15
T2.5	Benchmarking and Optimization	T2.4	10
T3	App development		210
T3.1	Database Interaction Setup		35
T3.2	ILP Model Integration	T2.2	25
T3.3	Functionality Implementation	T3.1, T3.2	50
T3.4	User Interface Design	T3.3	50
T3.5	User Documentation Creation	T3.4	50
T4	Project documentation		50
T5	Preparation for the oral defence	T1, T2, T3, T4	30
		Total:	435

Table 1: Task Duration and dependencies



Name	Begin date	End date
T1 Project planning	21/09/2023	16/10/2023
T1.1 Temporal planning of the project	21/09/2023	26/09/2023
T1.2 Definition of the context and project scope	27/09/2023	02/10/2023
T1.3 Definition of the budget and study of sustainability impact	03/10/2023	09/10/2023
T1.4 Integration of Key Project Components	10/10/2023	16/10/2023
T2 Design of the ILP Model	25/08/2023	30/10/2023
T2.1 Study of ILP Modeling Using AMPL	25/08/2023	08/09/2023
T2.2 Formulate the Hypergraph Model	11/09/2023	15/09/2023
T2.3 Validation with Handcrafted Data	18/09/2023	20/09/2023
T2.4 Testing with Real Data	24/10/2023	26/10/2023
T2.5 Benchmarking and Optimisation	27/10/2023	30/10/2023
T3 App development	15/08/2023	04/12/2023
T3.1 Database Interaction Setup	15/08/2023	22/08/2023
T3.2 LP Model Integration	17/10/2023	23/10/2023
T3.3 Functionality Implementation	24/10/2023	06/11/2023
T3.4 User Interface Design	07/11/2023	20/11/2023
T3.5 User Documentation Creation	21/11/2023	04/12/2023
T4 Project Documentation	15/08/2023	19/12/2023
T5 Preparation for the Oral Defence	13/12/2023	19/12/2023

Figure 3: Gantt Diagram. Source : own compilation

5.3 Resources

In the pursuit of this research endeavor, several essential resources, both human and material, play a crucial role in facilitating the project's success. These resources are integral to the project's development, documentation, and overall execution.

5.3.1 Human Resources:

In this section, we introduce the team members working on this project and outline their roles and responsibilities:

- **Main Developer (Author):** The principal architect of this project, the author is responsible for the project's design, development, and execution.
- **University Thesis Supervisor:** The invaluable guidance and mentorship provided by the university thesis supervisor are instrumental in shaping the project's direction and ensuring academic rigor.
- **GEP Tutor:** The GEP (GESTIÓ DE PROJECTES) tutor contributes to the project by offering instructions and feedback on the project management aspect of the research.

We have identified five distinct roles, and assigned specific responsibilities within the project:

1. **Junior Project Manager:**
   Responsibilities: Project coordination, task scheduling, and overall project management.
   Assigned team member: Author
2. **Project Manager:**
   Responsibilities: Project supervision, guidance, research leadership, methodology development, and academic oversight.
   Assigned team members: Thesis supervisor, GEP tutor
3. **Junior Researcher:**
   Responsibilities: Research tasks, studying hypergraphs and AMPL modeling.
   Assigned team member: Author
4. **Senior Researcher:**
   Responsibilities: Research and academic guidance, leadership, methodology development.
   Assigned team member: Thesis supervisor
5. **Junior Full Stack Developer:**
   Responsibilities: Software development, programming, and application design.
   Assigned team member: Author

5.3.2 Material Resources:

Overleaf: A collaborative LaTeX platform, Overleaf simplifies project documentation, enabling efficient collaboration and progress tracking with the project team.
Atenea: The university's Atenea platform serves as a communication hub, providing instructions and feedback from the GEP tutor.
PyCharm IDE: A top-tier Python Integrated Development Environment (IDE) empowers the author with advanced programming capabilities, streamlining development tasks.
ChatGPT: A versatile AI tool, ChatGPT assists in text composition and programming tasks, ensuring the generation of grammatically correct and comprehensive text.
AMPL IDE: The AMPL Integrated Development Environment is employed for the development and testing of AMPL (A Mathematical Programming Language) models, a critical component of the project.
GitHub: A robust version control platform, GitHub facilitates collaborative coding, version management, and project organization.
High-Performance Computer: Equipped with a formidable Ryzen 7 5800x3D CPU, 32GB RAM operating at 3600MHz, and an AMD Radeon RX 6900XT GPU, this high-performance computer

serves as the primary development environment. In the event that this setup proves insufficient for computational demands, access to the Department of Computer Science's supercomputer cluster may be requested.

ID	Task	Roles	Material Resource
T1	Project Management	Junior Project Manager, Project Manager	Overleaf, Atenea, ChatGPT, High-Performance Comp
T1.1	Definition of the context and project scope	Junior Project Manager, Project Manager	Overleaf, Atenea, ChatGPT, High-Performance Comp
T1.2	Temporal planning of the project	Junior Project Manager, Project Manager	Overleaf, Atenea, ChatGPT, High-Performance Comp
T1.3	Definition of the budget and study of sustainability impact	Junior Project Manager, Project Manager	Overleaf, Atenea, ChatGPT, High-Performance Comp
T1.4	Integration of Key Project Components	Junior Project Manager, Project Manager	Overleaf, Atenea, ChatGPT, High-Performance Comp
T1.5	Meetings with the tutor	All roles	
T2	Design of the ILPModel	Junior Researcher	AMPL IDE, High-Performance Computer
T2.1	Study of ILP Modeling Using AMPL	Junior Researcher	
T2.2	Formulate the Hypergraph Model	Junior Researcher	AMPL IDE, High-Performance Computer
T2.3	Validation with Handcrafted Data	Junior Researcher	AMPL IDE, High-Performance Computer
T2.4	Testing with Real Data	Junior Researcher	AMPL IDE, High-Performance Computer
T2.5	Benchmarking and Optimization	Junior Researcher	AMPL IDE, High-Performance Computer
T3	App development	Junior Full Stack Developer	GitHub, PyCharm IDE, ChatGPT, High-Performance
T3.1	Database Interaction Setup	Junior Full Stack Developer	GitHub, PyCharm IDE, ChatGPT, High-Performance
T3.2	ILP Model Integration	Junior Full Stack Developer	GitHub, PyCharm IDE, ChatGPT, High-Performance
T3.3	Functionality Implementation	Junior Full Stack Developer	GitHub, PyCharm IDE, ChatGPT, High-Performance
T3.4	User Interface Design	Junior Full Stack Developer	GitHub, PyCharm IDE, ChatGPT, High-Performance
T3.5	User Documentation Creation	Junior Full Stack Developer	GitHub, PyCharm IDE, ChatGPT, High-Performance
T4	Project documentation	Junior Full Stack Developer	ChatGPT
T5	Preparation for the oral defence	Junior Researcher, Junior Full Stack Developer	ChatGPT

Table 2: Table of requirements (Human and Material) corresponding to each task

5.4 Risk Management: Obstacles and alternative plans

In the pursuit of this research project, we have identified several potential risks that may affect its execution. In this section, we discuss these risks, propose solutions, and provide estimates of possible consequences in terms of additional time and resources required.

5.4.1 Project Timeline

Risk: Some tasks may have been underestimated and may take longer than expected for various reasons.
Proposed Solution: It is crucial to monitor our progress and be aware of the expected development stage at any given time. This enables us to adjust our plans if necessary.
Possible Consequences: In the worst-case scenario, we may require an additional 50 to 80 hours of labor.
Probability: Low. We have already conducted a comprehensive field study during the document's preparation, allowing us to identify potential obstacles and adapt the project plan and scope accordingly.

5.4.2 Computational power

Risk: The computer we have may not be powerful enough to handle the largest hypergraphs efficiently.
Proposed Solution: In case our current setup proves insufficient, we will establish contact with the computing cluster of the Department of Computer Science to process more demanding cases.
Possible Consequences: In the event of computational limitations, there might be a delay of multiple days. During this time, Task T2.4 might be affected. However, our project plan allows us to proceed with different tasks that do not rely on T2.4 as a predecessor. At worst, we anticipate a delay of 15 hours.
Probability: Low. Early testing shows that the computer at our dispositions is sufficiently powerful.

5.4.3 Integration Challenges

Risk: Integration with the KEGG database may present challenges, including content and format discrepancies.
Proposed Solution: To address integration challenges effectively, we will implement the following strategies:
- Stay informed about updates to the KEGG database.
- Establish communication channels with the KEGG team if possible.
- Develop a flexible application architecture capable of accommodating changes as they occur.

Possible Consequences: Integration challenges may lead to additional effort and time spent on data processing and application adaptation. The extent of the consequences will depend on the nature and frequency of changes in the KEGG database. We expect a delay of up two 50 hours.
Probability: High. We have already identified several shortcomings in the data available that could impact Tasks T3.3 and T3.4.

5.4.4 Inexperience with ILP:

Risk: The author's limited experience with Integer Linear Programming (ILP) poses a potential challenge.
Proposed Solution: To overcome inexperience with ILP, we will adopt a proactive approach that includes:

- Utilizing educational resources, such as online courses and textbooks.
- Seeking guidance and mentorship from experts in the field.
- Engaging in practical exercises to apply ILP concepts.
- Collaborating with individuals experienced in ILP.
- Committing to continuous learning throughout the project.

Possible Consequences: While there may be an initial learning curve, dedicating time and effort to gaining proficiency in ILP should mitigate potential challenges related to inexperience. Some tasks may have been underestimated and may take longer than expected for various reasons. We may expect up to extra 20 hours.
Probability: Low. We already have a working model.

6 Budget

In this section, we present a comprehensive budget estimation for the successful completion of the project. The budget is categorized into two main components: Personnel Costs per Activity and Generic Costs. This breakdown offers transparency and insight into the allocation of resources.

6.1 Staff costs

As we saw in section 5.3.1, we identified three key individuals (the Author, the thesis supervisor and the GEP supervisor) who are involved in various roles to ensure the successful execution of the project.

In table 3, we present the estimated hourly rates for each of the roles, taking into account Social Security contributions. The cost estimation is based on average salaries for each role in the region of Barcelona.

Role	Annual Salary	Annual Salary + SS (35%)	Cost Per hour
Project Manager	€44,000.00 [?]	€59,400.00	€34.22
Junior Project Manager	€30,000.00 [?]	€40,500.00	€23.33
Junior Researcher	€25,000.00 [?]	€33,750.00	€19.44
Senior Researcher	€39,000.00 [?]	€52,650.00	€30.33
Junior Full Stack Developer	€25,000.00 [?]	€33,750.00	€19.44

Table 3: Estimated Staff Costs per hour (1736 working hours per year [?])

In table 4, we present the estimated cost of each task by defining how many hours each role must spend on it and by using the data from table 3.

What we obtain is an estimation of the Personnel Cost per Activity or PCA.

ID	Task	Hours per role					Total Hours	Price
		Project Manager	Jr.Project Manager	Jr. Researcher	Sr. researcher	Jr. Full Stack Dev.		
T1	Project Management	25	75	5	10	5	120	€3,102.82
T1.1	Context and scope	5	25	0	0	0	30	€754.32
T1.2	Temporal planning of the project	5	15	0	0	0	20	€521.03
T1.3	Budget and sustainability	5	15	0	0	0	20	€521.03
T1.4	Integration of Key Project Components	5	15	0	0	0	20	€521.03
T1.5	Meetings with the tutor	5	5	5	10	5	30	€785.43
T2	Design of the ILPModel	0	0	90	0	0	90	€1,749.71
T2.1	Study of ILP Modeling Using AMPL	0	0	40	0	0	40	€777.65
T2.2	Formulate the Hypergraph Model	0	0	15	0	0	15	€291.62
T2.3	Validation with Handcrafted Data	0	0	10	0	0	10	€194.41
T2.4	Testing with Real Data	0	0	15	0	0	15	€291.62
T2.5	Benchmarking and Optimization	0	0	10	0	0	10	€194.41
T3	App development	0	0	0	0	210	210	€4,082.66
T3.1	Database Interaction Setup	0	0	0	0	35	35	€680.44
T3.2	ILP Model Integration	0	0	0	0	25	25	€486.03
T3.3	Functionality Implementation	0	0	0	0	50	50	€972.06
T3.4	User Interface Design	0	0	0	0	50	50	€972.06
T3.5	User Documentation Creation	0	0	0	0	50	50	€972.06
T4	Project documentation	0	0	0	0	50	50	€972.06
T5	Preparation for the oral defence	0	0	15	0	15	30	€583.24
							Sum:	€19,425.69

Table 4: Personnel Cost per Activity (PCA) based on the costs defined in table 3

6.2 Generic costs

In this subsection, we examine the amortization of material resources utilized during the course of this study. Note that all the software that we use are free to use or have an free educational licence option, therefore they are not included. To calculate the amortization of these material resources, we use the following formula:

$$\text{Amortized Cost} = \frac{\text{Initial Cost} * \text{Hours of use for project}}{\text{Estimated hours of lifespan}}$$

Specifically, we consider the resources in table 5: To calculate the values, we use :

$$\text{Hours of use for project} = 435 \text{ hours}$$
$$\text{Estimated hours of lifespan} = \frac{\text{Estimated years of lifespan}}{1736 \text{ working hours per year[?]}}$$

Hardware	Price	Lifetime (in years)	Amortized price
Desktop Computer	€1,800	5	€90
QHD VA Display	€220	8	€7
Microsoft sculpt ergonomic keyboard	€80	5	€4
Logitec Superlight mouse	€80	4	€5
		Total :	€106

Table 5: Amortization of physical resources

In addition to the direct material resource costs, there are indirect expenses that need to be considered for the project. These costs are essential for the smooth operation and completion of the study.

Electricity
- Description: The cost of electrical consumption for running the computer, display, and other electronic equipment.
- Estimated Monthly Cost: 435 hours * 0.3kW (from personal testing) * 0.1224 €/kWH [?] = 15.97€

Internet Service:
- Description: The cost of internet connectivity, which is crucial for research, communication, and data access.
- Estimated Monthly Cost: 40€ per month [?] * 12 * $\frac{\text{435 project hours}}{\text{8760 hours per year}}$ = 23.83€

6.3 Budget Deviations

6.3.1 Contingency

To account for unavoidable unexpected events, we include an additional 10% in the total budget. This allowance is meant to accommodate the possible extra hours that may be required due to unforeseen circumstances. We have chosen to apply the same contingency factor to both PCA and GC because they are both calculated using a linear function of the total hours used.

6.3.2 Incidental Costs

In the previous sections, we have identified potential risks that may emerge during the course of the project. To address these risks and mitigate their impact on the budget, we propose the inclusion of an ”incidental cost” component. This allocation will serve to cover any extra expenses that may arise due to unforeseen circumstances.

Incident	Estimated Cost	Risk (%)	Incidental Cost
Project Timeline	€2000	10	€200
Computational power	€375	5	€18.75
Intergration challenges	€1250	50	€625
Inexperience with ILP	€500	0	€0
		Total Incidental Cost:	€843.75

Table 6: Incidental Costs

6.4 Management Control

In this section, we establish essential metrics to monitor and control project expenditures, ensuring that we adhere to the original budget. Our approach involves close supervision of task durations and cost estimations per hour, allowing us to align project progress with financial expectations.

6.4.1 Metrics for Budget Adherence:

1. **Task Duration Tracking:** We will closely monitor the actual time required to complete each task compared to the initial estimates. Deviations from estimated task durations will be identified and analyzed promptly.
2. **Cost Estimation vs. Real Costs:** We will assess whether the cost estimations per hour, based on roles and responsibilities, align with actual costs incurred during the project. Any disparities will trigger further investigation.

6.4.2 Adaptation for Budget Control

Adherence to the budget is paramount. In cases where metrics indicate potential budget deviations, we will take proactive measures to realign with financial expectations:

- **Task Reallocation:** If certain tasks consistently exceed estimated durations, we may consider redistributing responsibilities or resources to optimize efficiency.
- **Cost Adjustments:** Should discrepancies arise between estimated and actual costs, we will evaluate the factors contributing to the variance and adjust the budget accordingly.
- **Resource Optimization:** We will explore opportunities for resource optimization, including time and material resources, to ensure efficient utilization and cost-effectiveness.

By implementing these management control measures, we aim to maintain strict budget adherence throughout the project's lifecycle. This proactive approach allows us to adapt promptly and make informed decisions to ensure financial stability and project success.

6.5 Total Budget

We have compiled all the budget components discussed previously into the following table 7 to present the overall budget for the project:

7 Sustainability report

The conclusion of my bachelor's program at UPC provides a valuable opportunity for self-reflection on my understanding of the environmental impact of software engineering projects.

As the author of this thesis, I acknowledge that I possess a foundational understanding of sustainability, encompassing economic, environmental, and social dimensions. However, I am committed to continuous improvement in these areas.

Category	Cost
PCA	€19,425.69
GC	€146.12
Incidental Cost	€843.75

Total	€20,415.56
Total with Contigency	€22,457.12

Table 7: Final Budget

7.1 Economic dimenstion

Regarding PPP: Reflection on the cost you have estimated for the completion of the project

Reflecting on the cost estimation for this project, I find it to be a critical aspect of project planning and management. As detailed in Section 6, we meticulously assessed the costs, factoring in personnel and material resources, to develop a comprehensive budget.

One key takeaway from this exercise is the importance of accuracy in cost estimation. By thoroughly studying the costs associated with personnel and material resources, we aimed to create a realistic budget that aligns with project goals and objectives. However, it's essential to acknowledge that cost estimates are subject to variables and uncertainties that may arise during the project's execution.

Regarding Useful Life: How are currently solved economic issues (costs...) related to the problem that you want to address (state of the art)?, and How will your solution improve economic issues (costs ...) with respect other existing solutions?

It is important to note that this is a new and unique problem. The value of the results generated by the ILP (Integer Linear Programming) model developed in this thesis is yet unknown. Unlike established research topics or methodologies, there are no existing researchers or prior studies that directly address this particular issue. Therefore, it is challenging to draw comparisons or improvements based on existing solutions.

7.2 Environmental Dimension

Regarding PPP: Have you estimated the environmental impact of the project?

As previously mentioned, assessing the environmental impact of this project presents a unique challenge. The value of the results derived from the ILP (Integer Linear Programming) model developed in this thesis remains uncertain, making it challenging to predict its environmental implications. In the absence of concrete data or precedents, any assessment would rely heavily on speculation.

Regarding PPP: Did you plan to minimize its impact, for example, by reusing resources?

The potential for mitigating the impact of this project, such as resource reuse, is contingent on its outcomes and applications. As previously discussed, the value and practical applications of the ILP (Integer Linear Programming) model developed in this thesis are yet to be determined. Consequently, any specific plans for minimizing impact, including resource reuse, are inherently linked to the project's findings and utilization.

Regarding Useful Life: How is currently solved the problem that you want to address (state of the art)?, and how will your solution improve the environment with respect other existing solutions?

It is important to note that this is a new and unique problem. The value of the results generated by the ILP (Integer Linear Programming) model developed in this thesis is yet unknown. Unlike established research topics or methodologies, there are no existing researchers or prior studies that directly address this particular issue. Therefore, it is challenging to draw comparisons or improvements based on existing solutions.

7.3 Social Dimension

Regarding PPP: What do you think you will achieve -in terms of personal growth- from doing this project?

Engaging in this project offers a unique avenue for personal growth. It will deepen my expertise in hypergraph algorithms, honing problem-solving and research skills. Navigating novel challenges will cultivate innovation and analytical thinking. Moreover, as my knowledge expands, new professional opportunities are expected to emerge, enhancing my career prospects. Managing this project independently will also refine my time management and self-discipline. In essence, this endeavor is a pivotal step in my journey of personal and professional growth.

Regarding Useful Life: How is currently solved the problem that you want to address (state of the art)?, and how will your solution improve the quality of life (social dimension) with respect other existing solutions?

It is important to note that this is a new and unique problem. The value of the results generated by the ILP (Integer Linear Programming) model developed in this thesis is yet unknown. Unlike established research topics or methodologies, there are no existing researchers or prior studies that directly address this particular issue. Therefore, it is challenging to draw comparisons or improvements based on existing solutions.

Regarding Useful Life: Is there a real need for the project?

As said previously, the value of the results generated by the ILP (Integer Linear Programming) model developed in this thesis is yet unknown.

8 Creating an effective AMPL model

In this section, we delve into the process of creating an effective AMPL model to find optimal solutions for our problem. We will provide in-depth explanations of the key components, including parameters (param), sets (set), variables (var), objectives, and constraints.

8.1 Understanding the Basics of AMPL

AMPL, short for ”A Mathematical Programming Language,” is a powerful tool for mathematical optimization. To create a successful AMPL model, it’s essential to grasp the following fundamental concepts:

Parameters (param): Parameters in AMPL represent constant values or inputs used in your mathematical model. They serve as the numeric data that influences the model’s behavior. You define parameters using the ‘param‘ keyword, and they can be assigned specific values.
For instance, in a transportation optimization model, you could have a parameter representing the cost of shipping between two locations.

Sets (set): Sets in AMPL are used to represent collections of elements or entities. These entities can be products, locations, time periods, or any relevant entities, depending on your problem domain. You define sets using the ‘set‘ keyword and use them to specify the domains for your variables and constraints.
In a product distribution model, you might define a set to represent all available distribution centers.

Variables (var): Variables in AMPL symbolize the decision variables that the optimization model will adjust to achieve the desired outcome. You use the ‘var‘ keyword to define variables, setting their domains and optional bounds. These variables typically represent what you’re trying to optimize.
For example, in a production scheduling model, you may define variables to represent the quantity of each product to be produced.

Objectives: The objective function in an AMPL model defines what you aim to optimize. It is a mathematical expression that relies on variables and, sometimes, parameters within your model. You can specify objectives to be minimized or maximized.
For example, in a workforce scheduling model, your objective might be to minimize labor costs while meeting staffing requirements.

Constraints: Constraints are mathematical expressions that constrain the potential solutions of your optimization problem. These constraints typically consist of inequalities and equations involving variables, parameters, and sets. They ensure that the solution adheres to specific requirements or limitations.
In a project scheduling model, you might define constraints to manage resource availability or project duration.

In AMPL, you combine these core components to construct a comprehensive mathematical model that represents your real-world optimization challenge. Whether you’re optimizing supply chains, production processes, financial portfolios, or any other problem domain, a thorough understanding of parameters, sets, variables, objectives, and constraints is essential for constructing effective AMPL models that deliver optimal solutions.

8.2 Desired inputs/outputs

In the design of our AMPL model, we aim to create a versatile model that can work with any hypergraph G = (V, E).

Inputs:

- V: The set of vertices.
- E: The set of hyper-edges.

- $\forall e \in E, X(e)$: for each hyperedge, its tail set (which we call X).
- $\forall e \in E, Y(e)$: for each hyperedge, its head set (which we call Y).
- **Optional** $\forall e \in E$, *invertible*(e): for each hyperedge, whether it can be inverted.
- **Optional** $\forall n \in V$, *external*(n): Whether we must force a vertex to be external.

Outputs:

- $\forall e \in E$, *inverted*(e): Whether a hyperedge is inverted in our solution.
- $\forall n \in V$, *is_internal*(e): Whether a vertex is internal in our solution.

8.3 Explaining our models

In this section, we present the initial model prototype and two models created by the professor and his colleagues.

8.3.1 First prototype

In this section, we will provide an overview of the structure and functionality of the first prototype.

As we can see in figure 4, the code is divided in 4 parts.

1. **Inputs** The lines 5 through 9 defines the input fields as we described them in section 8.2.
2. **Helpers** The lines 17 and 18 define to sets that we use as helpers for the rest of the program.
   - **substrate_in**
     For every vertex *i*, we create a set of every hyperedge where *i* acts as a substrate (*i* belongs to the tail set)
   - **product_in**
     For every vertex *i*, we create a set of every hyperedge where *i* acts as a product (*i* belongs to the head set)
3. **Variables** The lines 24 to 29 define the variables. In addition to the outputs that we defined in section 8.2, we also have the binary variables:
   - **has_out**
     For every vertex *i*, the corresponding variable is equal to 1 only if 'i' has at least one outgoing hyperedge, 0 otherwise.
   - **has_in**
     For every vertex *i*, the corresponding variable is equal to 1 only if 'i' has at least one incoming hyperedge, 0 otherwise.
4. **Rules** The lines 35 to 63 last part defines the objective and the constraints.
   - **maximize obj**
     This defines the objective. We want to maximize the number of internal nodes
   - **compute_is_internal**
     The goal of this constraint is to only allow **is_internal** to be 1 when both **has_in** and **has_out** are equal to 1.
   - **substrates_not_inverted**
     This constraint forces the **has_in** variable assigned to every vertex *i* to not be 0 when at least one of *i*'s hyperedges where it appears as substrate hasn't been inverted.
   - **substrates_inverted**
     This constraint forces the **has_out** variable assigned to every vertex *i* to not be 0 when at least one of *i*'s hyperedges where it appears as substrate has been inverted.
   - **products_not_inverted**
     This constraint forces the **has_out** variable assigned to every vertex *i* to not be 0 when at least one of *i*'s hyperedges where it appears as product hasn't been inverted.
   - **products_inverted**
     This constraint forces the **has_in** variable assigned to every vertex *i* to not be 0 when at least one of *i*'s hyperedges where it appears as product has been inverted.

- **products_inverted**
  This constraint forces the **has_in** variable assigned to every vertex *i* to not be 0 when at least one of *i*'s hyperedges where it appears as product has been inverted.
- **not_substrate_at_all**
  This constraint forces the **has_in** variable assigned to every vertex *i* to be 0 when none of *i*'s hyperedges where it appears as substrate hasn't been inverted and none where it appears as product has been inverted.
- **not_product_at_all**
  This constraint forces the **has_out** variable assigned to every vertex *i* to be 0 when none of *i*'s hyperedges where it appears as product hasn't been inverted and none where it appears as substrate has been inverted.
- **respect_invertability**
  This constraint forces the **inverted** variable assigned to every hyperedge *i* to be 0 when it isn't allowed to be inverted by the problem definition.

As observed, the presence of numerous strong constraints creates a significant interdependency among the variables: **inverted**, **has_in**, **has_out**, and **is_internal**. Notably, when there is an outgoing edge, **has_out** can only assume a value of 1, and a similar condition applies to **has_in**. This strict mathematical relationship eliminates any ambiguity. It is through this robust mathematical correlation that we can confidently affirm the fidelity of our model in representing the problem.

```
#####################
## Inputs
#####################

set V; # nodes
set E; # hyperedges

set X{E}; # for each hyperedge, its tail set
set Y{E}; # for each hyperedge, its head set

param invertible{E} binary; # determines whether an edge is invertible

#####################
## Helpers
#####################

set substrate_in{i in V} := {j in E: i in X[j]};
set product_in{i in V} := {j in E: i in Y[j]};

#####################
## Variables
#####################

var inverted {E} binary; #determines whether an edge is inverted

var has_out{V} binary;
var has_in{V} binary;

var is_internal{V} binary;

#####################
## Rules
#####################

maximize obj: sum{i in V} is_internal[i];

subject to compute_is_internal{i in V}:
        is_internal[i] <= has_in[i] + has_out[i] - 1;

subject to substrates_not_inverted{i in V, j in substrate_in[i]}:
        has_in[i] >= 1-inverted[j];

subject to substrates_inverted{i in V, j in substrate_in[i]}:
        has_out[i] >= inverted[j];

subject to products_not_inverted{i in V, j in product_in[i]}:
        has_out[i] >= 1-inverted[j];

subject to products_inverted{i in V, j in product_in[i]}:
        has_in[i] >= inverted[j];

subject to not_substrate_at_all{i in V}:
        has_in[i] <=
            sum{j in substrate_in[i]} (1-inverted[j]) +
            sum{j in product_in[i]} inverted[j];

subject to not_product_at_all{i in V}:
        has_out[i] <=
            sum{j in product_in[i]} (1-inverted[j]) +
            sum{j in substrate_in[i]} inverted[j];

subject to respect_invertability {i in E}:
        inverted[i] <= invertible[i];
```

Figure 4: The model file of the first prototype

8.3.2 Models Developed by the Professor and His Colleagues

In this section, we introduce two models developed by the professor and their colleague. To enhance clarity within the context we presented, various aspects such as parameters, variables, sets, and/or rule names have been modified, ensuring that all the models share similar names. This deliberate adjustment simplifies comprehension and facilitates a more coherent understanding of the models as they are discussed in the subsequent content.

We present below the first model (Figure 5). To respect the author's anonymity, we call it "Model A".

```
#####################
## Parameters
#####################
set V;
set E;

set X{E} within V; # The tail set for each hyperedge
set Y{E} within V; # The head set for each hyperedge
param invertible{E} binary; # determines whether an edge is invertible

#####################
## Variables
#####################

var inverted{E} binary; # 1 iff edge is inverted (Head and Tail)
var is_internal{V} binary; # to maximize


#####################
## Rules
#####################

maximize obj: sum {j in V} is_internal[j];

subject to outgoing_half_implies_internal {j in V}:
    sum {i in E: j in X[i]} (1-inverted[i]) +
    sum {i in E: j in Y[i]} inverted[i]
    >= is_internal[j];
    # will only allow is_internal to be one if there's an outgoing edge.

subject to incoming_half_implies_internal {j in V}:
    sum {i in E: j in Y[i]} (1-inverted[i]) +
    sum {i in E: j in X[i]} inverted[i]
    >= is_internal[j];
    # will only allow is_internal to be one if there's an incoming edge.
```

Figure 5: AMPL Model - Model A

Model A is comparatively simpler, featuring only two rules and two variables, both of which also appear in the first prototype (Figure 4). Although the rules 'outgoing_half_implies_internal' and 'incoming_half_implies_internal' are presented slightly differently, they are mathematically equivalent to the rules 'not_substrate_at_all' and 'not_product_at_all' from the first prototype.

As we will see in the following sections, this model excels in speed but lacks the mathematical robustness needed to accommodate more complex constraints, such as forcing certain vertices to be external or internal from the beginning.

Similarly, the second model "Model B" (Figure 6), is displayed below:

```
reset;
set V;
set E;

param M = card(V);

set X {E} within V;
set Y {E} within V;


var inverted {E} binary;
var source {V} binary;
var sink {V} binary;
var is_internal{V} binary;

maximize internal: sum {j in V} is_internal[j];

#############

s.t. outgoing_1{j in V}: #for every vertex
    sum{i in E: j in Y[i]} inverted[i]  +
    sum{i in E: j in X[i]} (1-inverted[i]) #how many times it appears as substrate
    - M * source[j]
    <= 0;

s.t. outgoing_2{j in V}:
    sum{i in E: j in Y[i]} inverted[i]  +
    sum{i in E: j in X[i]} (1-inverted[i])
    - 0.5 * source[j]
    >= 0;

##############

s.t. incoming_1{j in V}:
    sum{i in E: j in Y[i]} (1-inverted[i]) +
    sum{i in E: j in X[i]} inverted[i]
    - M * sink[j]
    <= 0;

s.t. incoming_2{j in V}:
    sum{i in E: j in Y[i]} (1-inverted[i]) +
    sum{i in E: j in X[i]} inverted[i]
    - 0.5 * sink[j]
    >= 0;

###############

s.t. internal_1 {j in V}:
    source[j] + sink[j] -1 - M * is_internal[j] <= 0;

s.t. internal_2 {j in V}:
    source[j] + sink[j] -1 - 0.5 * is_internal[j] >= 0;
```

Figure 6: AMPL Model - Model B

Model B's design philosophy aimed to follow a mathematical procedure to convert a mathematical problem into an optimization problem. Model B's model is much stronger mathematically than Model A's and strongly inspired the final model chosen for the project.

8.4 Initial Benchmarking

Our benchmarking approach involves running 25 pathway optimizations for each of the models we want to evaluate, including the three largest ones. We conducted these benchmarks using the dedicated **Benchmark View** of our Python program, detailed in Section 10.4. The computer for all benchmarks is the high-end desktop computer described in Section 5.3.2. The outcomes of our initial benchmark are summarized in Table 8, featuring five columns. The first column displays the pathway ID, the second indicates the solver used, and the last three columns depict the time taken by each model to complete the optimization problem in seconds.

Upon examination of the results, a noticeable performance gap emerges, particularly when compared to our first prototype. This initial version exhibits suboptimal performance, taking roughly three times longer than **Model B** and eight times longer than **Model A** to solve the most challenging pathway problem. This difference highlights specific areas where improvements or optimizations could enhance the overall performance of our prototype models.

entry	solver	First Prototype	Model A	Model B
map01100	cbc	74.960s	8.572s	24.458s
map01110	cbc	25.927s	3.938s	10.407s
map01120	cbc	9.042s	1.186s	4.058s
map01200	cbc	0.474s	0.258s	0.521s
map01210	cbc	0.462s	0.343s	0.604s
map01212	cbc	0.467s	0.211s	0.385s
map01230	cbc	0.432s	0.243s	0.419s
map01232	cbc	0.350s	0.244s	0.424s
map01250	cbc	0.401s	0.245s	0.710s
map01240	cbc	1.148s	0.301s	1.373s
map01220	cbc	0.769s	0.276s	0.479s
map00010	cbc	0.357s	0.275s	0.339s
map00020	cbc	0.351s	0.281s	0.308s
map00030	cbc	0.351s	0.256s	0.365s
map00040	cbc	0.296s	0.387s	0.352s
map00051	cbc	0.345s	0.299s	0.341s
map00052	cbc	0.294s	0.194s	0.316s
map00053	cbc	0.389s	0.335s	0.317s
map00500	cbc	0.310s	0.287s	0.304s
map00520	cbc	0.491s	0.355s	0.540s
map00620	cbc	0.357s	0.246s	0.406s
map00630	cbc	0.446s	0.516s	0.970s
map00640	cbc	0.999s	0.464s	0.516s
map00650	cbc	0.769s	0.372s	0.464s
map00660	cbc	0.401s	0.238s	0.271s

Table 8: Initial Benchmark

8.5 Creating and optimizing our final model

This section focuses on optimizing our models. **Serret's DualImply** model will be introduced in this section.

Drawing insights from extensive testing and our initial benchmark, we've identified key principles that contribute to improving the model's efficiency compared to others.

Our optimization principles include:

Simplicity in Representation: An effective model is achieved by minimizing the number of rules and variables for a concise representation.

Multiplication of Binary Variables: Despite the multiplication support for binary variables in the **maximize/minimize** function in newer AMPL versions, we explore the efficiency gained by manually introducing rules and variables to transform the problem back into an integer form, rather than relying solely on AMPL's backend.

Guided by these considerations, we present a redesigned model. This model has undergone continuous refinement throughout the project's development, ensuring it is not only mathematically robust but also thoroughly optimized. The final representation of our optimized model is provided below:

```
# ####################
## Parameters
# ####################
set V;
set E;

param M = card (E);

set X {i in E} within V; # Tail set
set Y {i in E} within V; # Head set

set uninvertibles within E; # set of edges that cannot be inverted
set forced_internals within V; # set of nodes that must be sinks
set forced_externals within V; # set of nodes that must be sources

# ####################
## Variables
# ####################
var inverted {E} binary ; # from X[i] to Y[i]
var has_outgoing {V} binary ;
var has_incoming {V} binary ;
var is_internal {V} binary;

# ####################
## Rules
# ####################
maximize internal : sum {j in V} is_internal[j];

subject to is_internal_1 {i in V}:
        is_internal[i] * 2  <= has_incoming[i] + has_outgoing[i];

subject to outgoing_implies_has_outgoing {j in V}:
        sum {i in E: j in X[i]} (1- inverted [i]) +
        sum {i in E: j in Y[i]} inverted [i]
        >= has_outgoing [j];

subject to has_outgoing_implies_outgoing {j in V}:
        sum {i in E: j in X[i]} (1- inverted [i]) +
        sum {i in E: j in Y[i]} inverted [i]
        <= M * has_outgoing [j];

subject to incoming_implies_has_incoming {j in V}:
        sum {i in E: j in X[i]} inverted [i] +
        sum {i in E: j in Y[i]} (1- inverted [i])
        >= has_incoming [j];

subject to has_incoming_implies_incoming {j in V}:
        sum {i in E: j in X[i]} inverted [i] +
        sum {i in E: j in Y[i]} (1- inverted [i])
        <= M * has_incoming [j];

# ####################
## Extra Restrictions
# ####################

subject to forced_externals_rule {i in forced_externals }:
        has_incoming [i] + has_outgoing [i] <= 1;

subject to forced_internals_rule {i in forced_internals }:
        has_incoming [i] + has_outgoing [i] = 2;

subject to respect_invertability {i in uninvertibles }:
        inverted [i] = 0;
```

Figure 7: Serret’s DualImply Model (with extra restrictions)

This model is designed to handle extra constraints that were not initially part of the original problem definition but are expected to be biologically relevant and extend the problem’s scope. These additional constraints, termed ’extra restrictions,’ offer the following functionalities:

1. The ability to designate certain vertices (or biological compounds) as internal or external.

2. The capability to prevent the inversion of specific hyper-edges (or biological reactions), ensuring the preservation of their original orientation.

A

8.6 Mathematical proof of the final model

In this section, we provide a formal mathematical proof to establish the equivalence between our proposed model and its Integer Linear Programming (ILP) representation. To facilitate this proof, we introduce specific predicates and functions that will help demonstrate the correspondence.

8.6.1 Introduction of Predicates and Functions:

We introduce the following predicates:

is_internal_i :	True when the vertex i is internal
has_outgoing_i :	True when the vertex i acts a substrate in at least one hyperedge
has_incoming_i :	True when the vertex i acts a product in at least one hyperedge

Additionally, we define the following functions:

$\textbf{isOne}(x)$:	Returns True if and only if x == 1, otherwise returns False
$\textbf{toNum}(x)$:	Returns 1 if and only if predicate x is True, otherwise returns 0

These functions serve to establish the equivalence between the mathematical model and the ILP model.

8.6.2 Obtaining is_internal from has_outgoing and has_incoming

The objective of our model is to find an orientation that maximizes the number of internal vertices. Thus, we formulate our maximizing function:

$$\textit{to\_maximize} = \textit{sum}(\ \{\ \textbf{toNum}(\text{is\_internal}_i) \mid \forall i \in V\ \}\ )$$

Now, we mathematically define when a node is internal, building upon the definition provided in Section 1.1.3:

$$\text{is\_internal}_i \equiv \text{has\_incoming}_i \land \text{has\_outgoing}_i \tag{1}$$

We would like to prove that this alternative definition is equivalent:

$$\text{is\_internal}_i \equiv \text{inequation\_1}_i \land \text{inequation\_2}_i \tag{2}$$

Where inequation_1_i is defined as

$$\textbf{toNum}(\text{is\_internal}_i) * 2 <= \textbf{toNum}(\text{has\_incoming}_i) + \textbf{toNum}(\text{has\_outgoing}_i) \tag{3}$$

and inequation_2_i is defined as

$$\textbf{toNum}(\text{is\_internal}_i) >= \textbf{toNum}(\text{has\_incoming}_i) + \textbf{toNum}(\text{has\_outgoing}_i) - 1 \tag{4}$$

To do so we will use the Table 9 to prove that, for every value of has_incoming_i, has_outgoing_i and is_internal_i, the equivalence (1) is the same as the equivalence (2):

is_internal$_i$	has_incoming$_i$	has_incoming$_i$	inequation_1$_i$ (3)	inequation_2$_i$ (4)	equivalence (2)	equivalence (1)
F	F	F	T	T	T	T
F	F	T	T	T	T	T
F	T	F	T	T	T	T
F	T	T	T	F	F	F
T	F	F	F	T	F	F
T	F	T	F	T	F	F
T	T	F	F	T	F	F
T	T	T	T	T	T	T

Table 9: Truth table of the equivalence of is_internal and the system of inequations

Given what we just showed, we can create the following set of AMPL rules that create the interdependence of has_incoming$_i$, has_outgoing$_i$ and is_internal$_i$ based the equivalence 2.

```
subject to is_internal_1 {i in V}:
        is_internal[i] * 2 <= has_incoming [i] + has_outgoing [i];

subject to is_internal_2 {i in V}:
        is_internal[i] >= has_incoming[i] + has_outgoing[i] - 1;
```

Figure 8: Strong AMPL rules used to define has_internal$_i$

Optimization :

The maximizing function aims to maximize the number of internal vertices, as defined in the problem statement. Given that the variable `is_internal`$_i$ is solely used in the maximizing function and no other rules apart from those used to define `is_internal`, we can make the assumption:

> When the solver is presented with a *is_internal* variable that can either be set to true or to false, the solver will always set it to true.

Based on this assumption, we observe that the second inequality in (4) becomes redundant. Table 10 demonstrates that we achieve identical results by utilizing only the first rule.

has_incoming$_i$	has_outgoing$_i$	values of is_internal$_i$ satisfying inequation_1$_i$ (3)	values of is_internal$_i$ satisfying inequation_2$_i$ (4)	values of is_internal$_i$ allowed by definition (2)
F	F	F	{ F, T }	F
F	T	F	{ F, T }	F
T	F	F	{ F, T }	F
T	T	{ F, T }	T	T

has_incoming$_i$	has_outgoing$_i$	values of is_internal$_i$ satisfying inequation_1$_i$ (3)	value of is_internal$_i$ chosen by the solver based solely on inequation_1 (3)
F	F	F	F
F	T	F	F
T	F	F	F
T	T	{ F, T }	T

Table 10: Optimizing is_internal by removing a redundant inequation

Optimization Impact:

Through extensive testing, implementing this optimization resulted in a noteworthy speedup of up to 1.14 with respect to the original method. This enhancement was particularly pronounced when applied to the largest model within our dataset.

8.6.3 Obtaining has_outgoing from the X and Y sets

For this section, we need to introduce some new predicates and functions:

inverted_j:	True if the solution inverts the hyperedge j.
$\textbf{inTailSet}(i)$:	returns the set of hyperedges that have the vertex i in their tail set.
$\textbf{inHeadSet}(i)$:	returns the set of hyperedges that have the vertex i in their head set.

We begin with the following statement:

$$\text{has\_outgoing}_i \iff (\exists\, j \in \textbf{inTailSet}(i) : \neg\text{inverted}_j) \vee (\exists\, j \in \textbf{inHeadSet}(i) : \text{inverted}_j) \quad (5)$$

Through this statement, we are saying that for a vertex to be considered to have an outgoing hyperedge in the solution, it must appear in the tail set of an uninverted hyperedge or in the head set of an inverted hyperedge in the problem input.

Using the rule $p \iff q \equiv (\neg p \vee q) \wedge (p \vee \neg q)$, we can break down the statement above into :

$$(\neg\text{has\_outgoing}_i \vee (\exists\, j \in \textbf{inTailSet}(i) : \neg\text{inverted}_j) \vee (\exists\, j \in \textbf{inHeadSet}(i) : \text{inverted}_j)) \wedge$$
$$(\text{has\_outgoing}_i \vee \neg((\exists\, j \in \textbf{inTailSet}(i) : \neg\text{inverted}_j) \vee (\exists\, j \in \textbf{inHeadSet}(i) : \text{inverted}_j))) \quad (6)$$

As we can see from the last statement, we have two parts that are combined with an $\wedge$. We will deal with each part individually by separating them into two parts.

First Part:

$$\neg\text{has\_outgoing}_i \vee (\exists\, j \in \textbf{inTailSet}(i) : \neg\text{inverted}_j) \vee (\exists\, j \in \textbf{inHeadSet}(i) : \text{inverted}_j) \quad (7)$$

$$\equiv \neg\text{has\_outgoing}_i \vee \Big(\sum_{j \in \textbf{inTailSet}(i)} (1 - \textbf{toNum}(\text{inverted}_j)) >= 1\Big) \vee \Big(\sum_{j \in \textbf{inHeadSet}(i)} \textbf{toNum}(\text{inverted}_j) >= 1\Big) \quad (8)$$

$$\equiv \neg\text{has\_outgoing}_i \vee \Big(\sum_{j \in \textbf{inTailSet}(i)} (1 - \textbf{toNum}(\text{inverted}_j)) + \sum_{j \in \textbf{inHeadSet}(i)} \textbf{toNum}(\text{inverted}_j) >= 1\Big) \quad (9)$$

$$\equiv (1 - \text{has\_outgoing}_i = 1) \vee \Big(\sum_{j \in \textbf{inTailSet}(i)} (1 - \textbf{toNum}(\text{inverted}_j)) + \sum_{j \in \textbf{inHeadSet}(i)} \textbf{toNum}(\text{inverted}_j) >= 1\Big) \quad (10)$$

$$\equiv (\text{has\_outgoing}_i = 0) \vee \Big(\sum_{j \in \textbf{inTailSet}(i)} (1 - \textbf{toNum}(\text{inverted}_j)) + \sum_{j \in \textbf{inHeadSet}(i)} \textbf{toNum}(\text{inverted}_j) >= 1\Big) \quad (11)$$

$$\equiv \sum_{j \in \textbf{inTailSet}(i)} (1 - \textbf{toNum}(\text{inverted}_j)) + \sum_{j \in \textbf{inHeadSet}(i)} \textbf{toNum}(\text{inverted}_j) >= \textbf{toNum}(\text{has\_outgoing}_i) \quad (12)$$

The first step is made using the following consideration: $\exists i \in \mathbb{S} : i \quad \equiv \quad (\sum_{\textbf{toNum}(i) \in \mathbb{S}} i) >= 1$

The last step can be made using the following considerations : when has_outgoing_i is 1, it is trivial to see that the equivalence holds. When it is zero, the last statement will always be true since $(\sum_{j \in \textbf{inTailSet}(i)} (1 - \textbf{toNum}(\text{inverted}_j)) + \sum_{j \in \textbf{inHeadSet}(i)} \textbf{toNum}(\text{inverted}_j)) \in \mathbb{N}_0$.

Second Part:

$$\text{has\_outgoing}_i \lor \neg((\exists\, j \in \textbf{inTailSet}(i) : \neg\text{inverted}_j) \lor (\exists\, j \in \textbf{inHeadSet}(i) : \text{inverted}_j)) \quad (13)$$

$$\equiv \text{has\_outgoing}_i \lor (\neg(\exists\, j \in \textbf{inTailSet}(i) : \neg\text{inverted}_j) \land \neg(\exists\, j \in \textbf{inHeadSet}(i) : \text{inverted}_j)) \quad (14)$$

$$\equiv \text{has\_outgoing}_i \lor (\neg(\sum_{j \in \textbf{inTailSet}(i)} (1-\textbf{toNum}(\text{inverted}_j)) >= 1) \land \neg(\sum_{j \in \textbf{inHeadSet}(i)} \textbf{toNum}(\text{inverted}_j) >= 1)) \quad (15)$$

$$\equiv \text{has\_outgoing}_i \lor ((\sum_{j \in \textbf{inTailSet}(i)} (1-\textbf{toNum}(\text{inverted}_j)) < 1) \land (\sum_{j \in \textbf{inHeadSet}(i)} \textbf{toNum}(\text{inverted}_j) < 1)) \quad (16)$$

$$\equiv \text{has\_outgoing}_i \lor ((\sum_{j \in \textbf{inTailSet}(i)} (1-\textbf{toNum}(\text{inverted}_j)) = 0) \land (\sum_{j \in \textbf{inHeadSet}(i)} \textbf{toNum}(\text{inverted}_j) = 0)) \quad (17)$$

$$\equiv \text{has\_outgoing}_i \lor (\sum_{j \in \textbf{inTailSet}(i)} (1-\textbf{toNum}(\text{inverted}_j)) + \sum_{j \in \textbf{inHeadSet}(i)} \textbf{toNum}(\text{inverted}_j) = 0) \quad (18)$$

$$\equiv \sum_{j \in \textbf{inTailSet}(i)} (1-\textbf{toNum}(\text{inverted}_j)) + \sum_{j \in \textbf{inHeadSet}(i)} \textbf{toNum}(\text{inverted}_j) <= |E| * \textbf{toNum}(\text{has\_outgoing}_i) \quad (19)$$

The first step is made using the following consideration: $\exists i \in \mathbb{S} : i \quad \equiv \quad (\sum_{i \in \mathbb{S}} i) >= 1$

The last step can be made using the following considerations: when **toNum**(has_outgoing$_i$) is 0, it is trivial to see that the equivalence holds. When it is 1, the value of the double sum should not be hindered, therefore we multiply **toNum**(has_outgoing$_i$) by the highest value the double sums can have : the cardinality of the hyperedges (since a vertex cannot be part of both the head and tail set).

8.6.4 Obtaining has_incoming from the X and Y sets

In the same way that we obtained has_outgoing from the X and Y sets, we will do the same for has_incoming.

$$\text{has\_incoming}_i \iff (\exists\, j \in \textbf{inTailSet}(i) : \text{inverted}_j) \lor (\exists\, j \in \textbf{inHeadSet}(i) : \neg\text{inverted}_j) \quad (20)$$

Using the rule $p \iff q \equiv (\neg p \lor q) \land (p \lor \neg q)$, we can break down the statement above into :

$$\begin{aligned}&(\neg\text{has\_incoming}_i \lor (\exists\, j \in \textbf{inTailSet}(i) : \text{inverted}_j) \lor (\exists\, j \in \textbf{inHeadSet}(i) : \neg\text{inverted}_j)) \land \\ &\quad (\text{has\_outgoing}_i \lor \neg((\exists\, j \in \textbf{inTailSet}(i) : \text{inverted}_j) \lor (\exists\, j \in \textbf{inHeadSet}(i) : \neg\text{inverted}_j)))\end{aligned} \quad (21)$$

Again, we will break this statement down into two parts:

First Part:

$$\neg\text{has\_incoming}_i \lor (\exists\, j \in \textbf{inTailSet}(i) : \text{inverted}_j) \lor (\exists\, j \in \textbf{inHeadSet}(i) : \neg\text{inverted}_j) \tag{22}$$

$$\equiv \neg\text{has\_incoming}_i \lor \Big(\sum_{j \in \textbf{inTailSet}(i)} \textbf{toNum}(\text{inverted}_j) >= 1\Big) \lor \Big(\sum_{j \in \textbf{inHeadSet}(i)} (1 - \textbf{toNum}(\text{inverted}_j)) >= 1\Big) \tag{23}$$

$$\equiv \neg\text{has\_incoming}_i \lor \Big(\sum_{j \in \textbf{inTailSet}(i)} \textbf{toNum}(\text{inverted}_j) + \sum_{j \in \textbf{inHeadSet}(i)} (1 - \textbf{toNum}(\text{inverted}_j)) >= 1\Big) \tag{24}$$

$$\equiv (1 - \text{has\_incoming}_i = 1) \lor \Big(\sum_{j \in \textbf{inTailSet}(i)} \textbf{toNum}(\text{inverted}_j) + \sum_{j \in \textbf{inHeadSet}(i)} (1 - \textbf{toNum}(\text{inverted}_j)) >= 1\Big) \tag{25}$$

$$\equiv (\text{has\_incoming}_i = 0) \lor \Big(\sum_{j \in \textbf{inTailSet}(i)} \textbf{toNum}(\text{inverted}_j) + \sum_{j \in \textbf{inHeadSet}(i)} (1 - \textbf{toNum}(\text{inverted}_j)) >= 1\Big) \tag{26}$$

$$\equiv \sum_{j \in \textbf{inTailSet}(i)} \textbf{toNum}(\text{inverted}_j) + \sum_{j \in \textbf{inHeadSet}(i)} (1 - \textbf{toNum}(\text{inverted}_j)) >= \textbf{toNum}(\text{has\_incoming}_i) \tag{27}$$

The first step is made using the following consideration: $\exists i \in \mathbb{S} : i \quad \equiv \quad (\sum_{\textbf{toNum}(i) \in \mathbb{S}} i) >= 1$

The last step can be made using the following considerations : when $\textbf{toNum}(\text{has\_incoming}_i)$ is 1, it is trivial to see that the equivalence holds. When it is zero, the last statement will always be true since $(\sum_{j \in \textbf{inTailSet}(i)} (1 - \textbf{toNum}(\text{inverted}_j)) + \sum_{j \in \textbf{inHeadSet}(i)} \textbf{toNum}(\text{inverted}_j)) \in \mathbb{N}_0$.

Second Part:

$$\text{has\_incoming}_i \lor \neg((\exists\, j \in \textbf{inTailSet}(i) : \text{inverted}_j) \lor (\exists\, j \in \textbf{inHeadSet}(i) : \neg\text{inverted}_j)) \tag{28}$$

$$\equiv \text{has\_incoming}_i \lor (\neg(\exists\, j \in \textbf{inTailSet}(i) : \text{inverted}_j) \land \neg(\exists\, j \in \textbf{inHeadSet}(i) : \neg\text{inverted}_j)) \tag{29}$$

$$\equiv \text{has\_incoming}_i \lor \Big(\neg\Big(\sum_{j \in \textbf{inTailSet}(i)} \textbf{toNum}(\text{inverted}_j) >= 1\Big) \land \neg\Big(\sum_{j \in \textbf{inHeadSet}(i)} (1 - \textbf{toNum}(\text{inverted}_j)) >= 1\Big)\Big) \tag{30}$$

$$\equiv \text{has\_incoming}_i \lor \Big(\Big(\sum_{j \in \textbf{inTailSet}(i)} \textbf{toNum}(\text{inverted}_j) < 1\Big) \land \Big(\sum_{j \in \textbf{inHeadSet}(i)} (1 - \textbf{toNum}(\text{inverted}_j)) < 1\Big)\Big) \tag{31}$$

$$\equiv \text{has\_incoming}_i \lor \Big(\Big(\sum_{j \in \textbf{inTailSet}(i)} \textbf{toNum}(\text{inverted}_j) = 0\Big) \land \Big(\sum_{j \in \textbf{inHeadSet}(i)} (1 - \textbf{toNum}(\text{inverted}_j)) = 0\Big)\Big) \tag{32}$$

$$\equiv \text{has\_incoming}_i \lor \Big(\sum_{j \in \textbf{inTailSet}(i)} \textbf{toNum}(\text{inverted}_j) + \sum_{j \in \textbf{inHeadSet}(i)} (1 - \textbf{toNum}(\text{inverted}_j)) = 0\Big) \tag{33}$$

$$\equiv \sum_{j \in \textbf{inTailSet}(i)} \textbf{toNum}(\text{inverted}_j) + \sum_{j \in \textbf{inHeadSet}(i)} (1 - \textbf{toNum}(\text{inverted}_j)) <= |E| * \textbf{toNum}(\text{has\_incoming}_i) \tag{34}$$

8.6.5 Additional Constraints: forcing internal

For a vertex to be considered internal, it must have at least one outgoing edge and at least one incoming edge.

In the previous section we have already defined the predicates has_outgoing and has_incoming. Since this is a constraint and not a maximizing function we cannot use the integer multiplication operator '*' as it would make the model non-linear.

Therefore we opted for the following construction which we will show is equivalent to the 'and' logical operator.

a	b	a ∧ b
False	False	False
False	True	False
True	False	False
True	True	True

toNum(a)	**toNum**(b)	**toNum**(a) + **toNum**(b) = 2
0	0	False
0	1	False
1	0	False
1	1	True

Table 11: Truth table of the '∧' operator and the construction $a+b=2$

Therefore we can say that :

$$i \in \text{Forced\_internals} \iff \text{has\_outgoing}_i \land \text{has\_incoming}_i \quad (35)$$
$$\equiv i \in \text{Forced\_internals} \iff \textbf{toNum}(\text{has\_outgoing}_i) + \textbf{toNum}(\text{has\_incoming}_i) = 2 \quad (36)$$

8.6.6 Additional Constraints: forcing external

For a vertex to be external, it must not be internal, so we are looking for the negation of the previous constraint.

We therefore opted with the following construction which we will show is equivalent to the previous operator negated.

a	b	¬(a ∧ b)
False	False	True
False	True	True
True	False	True
True	True	False

toNum(a)	**toNum**(b)	**toNum**(a) + **toNum**(b) < 2
0	0	True
0	1	True
1	0	True
1	1	False

Table 12: Truth table of the negated '∧' operator and the construction $\textbf{toNum}(a)+\textbf{toNum}(b)<2$

Therefore we can say that :

$$i \in \text{Forced\_externals} \iff \neg(\text{has\_outgoing}_i \land \text{has\_incoming}_i) \quad (37)$$
$$\equiv i \in \text{Forced\_externals} \iff \textbf{toNum}(\text{has\_outgoing}_i) + \textbf{toNum}(\text{has\_incoming}_i) < 2 \quad (38)$$

8.6.7 Additional Constraints: respecting the invertability of hyperedges

Upon activating the **Use extra restrictions** button, a new menu surfaces. This menu empowers the user to establish additional constraints, as elaborated in Section 8.6.5. Within this menu, users can designate specific reactions (or hyperedges) as non-inverted, or classify particular compounds as external or internal. This feature provides a finer degree of control over the orientation and classification of elements in the hypergraph.

Some hyperedges are invertible, some are not. To distinguish them, we define a subset of the hyperedges which we call 'uninvertibles'.

The univertibles use the following definition:

$$i \in \text{internals} \iff \textbf{toNum}(inverted_i) = 0 \tag{39}$$

8.6.8 Comparing the mathematical correctness of Model A and Serret's models

Given what we have declared above, we can see that Zephyr's model is a direct implementation of the mathematical derivations we obtained from the definitions.

On the other hand, Model A is less complete in regards of its mathematical correctness. In fact, here are the rules that model A follows:

$$\text{has\_outgoing}_i \implies (\exists\, j \in \textbf{inTailSet}(i) : \neg\text{inverted}_j) \lor (\exists\, j \in \textbf{inHeadSet}(i) : \text{inverted}_j) \tag{40}$$

$$\text{has\_incoming}_i \implies (\exists\, j \in \textbf{inTailSet}(i) : \text{inverted}_j) \lor (\exists\, j \in \textbf{inHeadSet}(i) : \neg\text{inverted}_j) \tag{41}$$

Which are the same rules we defined above with the exception that it use a unidirectional implication instead of a bidirectional one.

In fact, Nasini's rules allows to omit the differentiating has_outgoing and has_incoming and directly finds whether a vertex is internal :

$$\text{is\_internal}_i \implies (\exists\, j \in \textbf{inTailSet}(i) : \neg\text{inverted}_j) \lor (\exists\, j \in \textbf{inHeadSet}(i) : \text{inverted}_j) \tag{42}$$

$$\text{is\_internal}_i \implies (\exists\, j \in \textbf{inTailSet}(i) : \text{inverted}_j) \lor (\exists\, j \in \textbf{inHeadSet}(i) : \neg\text{inverted}_j) \tag{43}$$

This is sufficient for cases where there are no additional restrictions, but Model A's model will fail when you introduce them.

8.7 Final Benchmark and conclusions

For the sake of comparing Model A with a mathematically equal model, we also introduce a variation of Serret's DualImply model (Figure 7) which we call Serret's UniImply model (Figure 9. As stated in the previous section, we cannot add the extra restrictions to that model since the model is not sufficiently mathematically robust.

```
reset;
#####################
## Parameters
#####################

set V; # nodes
set E; # hyperedges

set X{E}; # for each hyperedge, its tail set
set Y{E}; # for each hyperedge, its head set

set uninvertibles within E; # set of edges that cannot be inverted
set forced_externals within V; # set of nodes that must be sinks
set forced_internals within V; # set of nodes that must be sources

set substrate_in{i in V} := {j in E: i in X[j]};
set product_in{i in V} := {j in E: i in Y[j]};

#####################
## Variables
#####################
var inverted {E} binary; #determines whether an edge is inverted

var has_incoming{V} binary;
var has_outgoing{V} binary;
var is_internal{V} binary;

#####################
## Rules
#####################

maximize internal: sum{i in V} is_internal[i];

subject to not_substrate_at_all{i in V}:
        has_outgoing[i] <=
            sum{j in substrate_in[i]} (1-inverted[j]) +
            sum{j in product_in[i]} inverted[j];

subject to not_product_at_all{i in V}:
        has_incoming[i] <=
            sum{j in product_in[i]} (1-inverted[j]) +
            sum{j in substrate_in[i]} inverted[j];

subject to is_internal_1 {i in V}:
        is_internal[i] * 2  <= has_incoming[i] + has_outgoing[i];
```

Figure 9: Serret's UniImply model

In the same fashion as we made the initial benchmark, we complete the final benchmark and display the results in Table 13.

entry	solver	First Prototype	Serret DualImply	Serret UniImply	Model A	Model B
map01100	cbc	74.960s	24.274s	9.383s	8.572s	24.458s
map01110	cbc	25.927s	9.704s	4.631s	3.938s	10.407s
map01120	cbc	9.042s	3.279s	1.460s	1.186s	4.058s
map01200	cbc	0.474s	0.485s	0.266s	0.258s	0.521s
map01210	cbc	0.462s	0.507s	0.239s	0.343s	0.604s
map01212	cbc	0.467s	0.329s	0.202s	0.211s	0.385s
map01230	cbc	0.432s	0.389s	0.260s	0.243s	0.419s
map01232	cbc	0.350s	0.331s	0.224s	0.244s	0.424s
map01250	cbc	0.401s	0.382s	0.257s	0.245s	0.710s
map01240	cbc	1.148s	1.080s	0.476s	0.301s	1.373s
map01220	cbc	0.769s	0.539s	0.260s	0.276s	0.479s
map00010	cbc	0.357s	0.307s	0.273s	0.275s	0.339s
map00020	cbc	0.351s	0.299s	0.312s	0.281s	0.308s
map00030	cbc	0.351s	0.313s	0.265s	0.256s	0.365s
map00040	cbc	0.296s	0.286s	0.293s	0.387s	0.352s
map00051	cbc	0.345s	0.331s	0.245s	0.299s	0.341s
map00052	cbc	0.294s	0.418s	0.222s	0.194s	0.316s
map00053	cbc	0.389s	0.384s	0.399s	0.335s	0.317s
map00500	cbc	0.310s	0.319s	0.237s	0.287s	0.304s
map00520	cbc	0.491s	0.359s	0.421s	0.355s	0.540s
map00620	cbc	0.357s	0.460s	0.235s	0.246s	0.406s
map00630	cbc	0.446s	0.786s	0.993s	0.516s	0.970s
map00640	cbc	0.999s	0.352s	0.604s	0.464s	0.516s
map00650	cbc	0.769s	0.987s	0.437s	0.372s	0.464s
map00660	cbc	0.401s	0.634s	0.359s	0.238s	0.271s

Table 13: Final Benchmark

Note: It's worth noting that the lackluster performance of Serret's UniImply model can be attributed to its utilization of a greater number of variables and rules compared to the more streamlined and elegant definitions employed by Model A. This highlights the importance of model simplicity and efficiency in achieving optimal outcomes, with Model A showcasing a favorable balance between mathematical robustness and computational speed.

In conclusion, our comprehensive study and benchmarking analysis highlight Serret's DualImply model as an excellent solution for addressing the problem of orienting metabolic pathways to maximize the number of internal compounds. The model's mathematical robustness, coupled with its flexibility for extension through extra restrictions, positions it as a strong contender. The observed performance during benchmarking also establishes its efficiency relative to competing models.

However, it's essential to note that Model A, with its faster execution and slightly weaker mathematical robustness, remains a relevant option within this domain. The choice between the two models may ultimately depend on specific user requirements and priorities.

9 Database Integration

9.1 Acknowledgment of Attempted Integration using bioservices

In the course of my research, I explored the possibility of integrating the KEGG database using the bioservices library to fetch reaction data. Regrettably, this approach encountered significant challenges, and I would like to provide an overview of the issues encountered during the integration attempt.

The bioservices library was initially considered as a means to retrieve reactions from the KEGG database. However, this approach proved to be impractical for several reasons.

Firstly, the method of fetching all reactions using bioservices proved to be extremely slow, making it infeasible to acquire the data in a reasonable time frame. This approach was not scalable for large-scale data retrieval, which is essential for comprehensive analysis.

Furthermore, due to the extensive number of requests sent to the KEGG server in a short period of time, the server interpreted our actions as a potential denial of service attack. As a result, our IP address was temporarily blocked, rendering us unable to access the KEGG database for several hours.

Given these limitations, it became evident that alternative strategies for data integration were necessary to overcome these challenges.

The attempt to integrate the KEGG database using the bioservices library is documented in Figure 10, which shows the code used in this effort.

```python
from bioservices import KEGG
k = KEGG()
reactions = k.reactionIds
for reaction in reactions:
    data = k.parse(k.get(reaction))
    if 'PATHWAY' not in data.keys():
        continue
    for pathway in sorted(data['PATHWAY'].keys()):
        print("{}: {}: {}".format(
            reaction,
            pathway,
            data['EQUATION']))
```

Figure 10: Integration attempt using bioservices

This experience has been valuable in highlighting the complexities and limitations of real-world data integration in scientific research, as well as the need for robust, efficient, and responsible data retrieval methods.

9.2 REST API Data Mapping: Needs and Retrievals

To effectively gather information from KEGG's REST API, we must retrieve the following:

1. Comprehensive pathways, including their identifiers (IDs) and user-friendly descriptions.
2. Reaction identifiers associated with each pathway.
3. Compound identifiers serving as substrates or products for each reaction.

Regrettably, the REST API lacks specific calls to fulfill all our requirements directly. Instead, we employ a strategic approach by utilizing multiple endpoints and amalgamating their outcomes to obtain the necessary data.

Here is the list of endpoints that we use:

1. https://rest.kegg.jp/list/pathway Returns the list of all pathway ids and a human readable description.

Extract from output:
```
map01100 Metabolic pathways
map01110 Biosynthesis of secondary metabolites
map01120 Microbial metabolism in diverse environments
```

2. https://rest.kegg.jp/link/reaction/pathway Returns the list of all pathway ids and the reaction ids which belong to them.

Extract from output:
```
path:map00010 rn:R00014
path:rn00010 rn:R00014
path:map00010 rn:R00199
```

3. https://rest.kegg.jp/list/compound Returns the list of all compound ids and names used to refer to them

Extract from output:
```
C00001 H2O; Water
C00002 ATP; Adenosine 5'-triphosphate
C00003 NAD+; NAD; Nicotinamide adenine dinucleotide; DPN;
        Diphosphopyridine nucleotide; Nadide; beta-NAD+
```

4. https://rest.kegg.jp/link/compound/reaction Returns the list of all reactions ids and all commpounds ids that belong to that reaction

Extract from output:
```
rn:R00001 cpd:C00001
rn:R00002 cpd:C00001
rn:R00004 cpd:C00001
```

5. https://rest.kegg.jp/list/reaction Returns the list of all reaction ids and a textual representation of their equation.

Extract from output:
```
R00001 polyphosphate polyphosphohydrolase; Polyphosphate + n
    H2O <=> (n+1) Oligophosphate
R00002 reduced ferredoxin:dinitrogen oxidoreductase (ATP-
    hydrolysing); 16 ATP + 16 H2O + 8 Reduced ferredoxin <=>
    8 e- + 16 Orthophosphate + 16 ADP + 8 Oxidized ferredoxin
R00004 diphosphate phosphohydrolase; pyrophosphate
    phosphohydrolase; Diphosphate + H2O <=> 2 Orthophosphate
```

6. https://rest.kegg.jp/get/*[Reaction id]* Returns the entry for a specific reaction, including the ids of the pathways it belongs to and the equation in both identifier and human readable form. **Note :** You can at most query 10 different reactions at once.

Extract from output:

```
ENTRY       R00014                      Reaction
NAME        pyruvate:thiamin diphosphate
                acetaldehydetransferase (decarboxylating)
DEFINITION  Pyruvate + Thiamin diphosphate <=> 2-(alpha-
                Hydroxyethyl)thiamine diphosphate + CO2
EQUATION    C00022 + C00068 <=> C05125 + C00011
PATHWAY     rn00010  Glycolysis / Gluconeogenesis
            rn00020  Citrate cycle (TCA cycle)
            rn00620  Pyruvate metabolism
            rn00785  Lipoic acid metabolism
```

As evident, the fulfillment of the first and second requirements is straightforward through the utilization of the first and second endpoints, respectively.

However, addressing the third and final requirement—acquiring the list of compound identifiers for each reaction—poses a more intricate challenge. To achieve this efficiently, we must leverage all available endpoints, strategically combining their outputs. This comprehensive approach ensures a thorough extraction of the necessary data.

9.3 Equation Discovery: Unveiling Reaction Formulas and Compound IDs Across Varied Endpoints

In order to precisely outline our objectives, we aim to establish a mapping between reaction IDs and two distinct lists of compound IDs, one for substrates and the other for products.

Throughout this section, compounds are typically presented either as compound IDs or by one of their various names (e.g., ”Nicotinamide adenine dinucleotide”). It’s essential to note that most compounds have multiple names.

To underscore the fact that each compound is associated with several names, we choose to use the term **synonym** instead of **name** throughout the rest of this section.

By leveraging the capabilities of the third endpoint, we can generate a mapping from synonyms to a list of compounds that share the same synonym. It’s important to acknowledge that certain synonyms may refer to multiple distinct compounds, although the majority pertain to only one.

Example.

”Quinone”: [”C00472”[1]*, ”C15602”*[2]*]*

”Ethanol”: [”C00469”[3]*]*

Using the fourth endpoint, we can obtain a mapping of reaction IDs to a list of compound IDs, although we cannot determine if the compound acts as a substrate or product in the reaction.

Example.

”R00001”: [”C00001”, ”C00404”, ”C02174”]

[1]https://rest.kegg.jp/get/C00472
[2]https://rest.kegg.jp/get/C15602
[3]https://rest.kegg.jp/get/C00469

"R00002": ["C00001", "C00002", "C00008", "C00009", "C00138", "C00139", "C05359"]

By utilizing the fifth endpoint, we can acquire a mapping of reaction IDs to an equation where the substrates and products are represented in the form of synonyms.

Example.

"R00001": "polyphosphate polyphosphohydrolase; Polyphosphate + n H2O <=> (n+1) Oligophosphate"

"R00002": "reduced ferredoxin:dinitrogen oxidoreductase (ATP-hydrolysing); 16 ATP + 16 H2O + 8 Reduced ferredoxin <=> 8 e- + 16 Orthophosphate + 16 ADP + 8 Oxidized ferredoxin"

The subsequent step involves using the three previously obtained mappings to establish a new mapping from **reaction IDs** to the **substrates' compound IDs** and **products' compound IDs**.

Starting from the mapping of reaction IDs to an equation where the substrates and products are in their synonym form, we proceed to **discard** the equation's name by removing everything that precedes the last appearance of the "; " substring. If the "; " substring doesn't appear, then we retain everything.

Next, we separate the **substrates** and **products** using the substring "<=>".

From this point, we extract **each individual compound** by splitting them using the "+" character.

At this stage, we are left with synonyms prefixed by a quantifier, such as a number or some expression with 'n' (for example, n+1). To remove this quantifier, we subtract any matches to the following regular expression:

```
^\d+n? |^\(n\+\d+\) |^n
```

ˆ matches with the start of a string.
\d matches with any digit (equivalent to [1-9]).
+ is a quantifier that matches to one or more instances.
? is a quantifier that matches to one or no instances.
\+ matches with the char '+'.
| signifies alternate options. In this case, we can match any of the three options.

This leaves us with one of the **synonyms** for each **compound** in the equation.

By utilizing the mapping **synonym** to **list of compound IDs**, we can generate a list of candidate compound IDs. We filter out every candidate that does not appear in the list obtained through the mapping **reaction ID** to **list of compound IDs**. At this stage, if we have more than one or no candidates, we mark this reaction as a **broken reaction**. If we have exactly one candidate left, it indicates that we have identified the correct compound, and we can add it to our final mapping of **reaction ID** to **substrates and products** accordingly.

Below is a Python extract that demonstrates this procedure.

```python
    def compound_verbose_and_reaction_to_id(self, compound_verbose, reaction_id):
        """
        Returns the compound id that matches the verbose compound name

        Parameters:
            compound_verbose: a verbose compound name
            reaction_id: the reaction id that the compound belongs to

        Returns:
            a single compound id, or None if no match
        """

        synonym = re.sub(r'^\d+n? |^\(n\+\d+\) |^n ', '', compound_verbose)

        if synonym not in self.map_synonym_to_compound_id:
            print("Couldn't find the synonym :", synonym)
            return None
        candidates = [
            candidate # compound id
            for candidate in self.map_synonym_to_compound_id[synonym]
            if candidate
                in self.map_reaction_id_to_list_compound_id.get(reaction_id, [])
        ]
        if len(candidates) > 1:
            print("compound: ", synonym, " has multiple candidates: ", candidates,
                  " for reaction: ", reaction_id, ", marking as broken")
            return None
        if len(candidates) < 1:
            print("compound: ", synonym,
                "has no candidates in reaction", reaction_id, ", marking as broken")
            return None
        print("compound:
            ", synonym, " has id: ", candidates[0], " for reaction: ", reaction_id)
        return candidates[0]


    def fetch_reaction_substrates_products_ids(self):
        """
            Fetches the list of reactions
                and creates the map reaction_id to substrate_ids and product_ids:
            Uses the result from the KEGG api as well as :
            - self.map_reaction_id_to_list_compound_id
            - self.map_synonym_to_compound_id
        """
        for reaction in REST.kegg_list("reaction").read().split('\n')[:-1]:
            broken = False
            reaction_id, reaction_equation_verbose = reaction.split('\t')
            reaction_equation_verbose = reaction_equation_verbose.split(";
                ")[-1] # remove the name of the equation
            self.map_reaction_id_to_substrates_products_ids[reaction_id]
                = {"substrates": [], "products": []}
            substrates_verbose,
                product_verbose = reaction_equation_verbose.split(' <=> ')
            for substrate_verbose in substrates_verbose.split(' + '):
                substrate_id
                    = self.compound_verbose_and_reaction_to_id(substrate_verbose,
                    reaction_id)
                if substrate_id is not None:
                    self.map_reaction_id_to_substrates_products_ids[reaction_id]\
                    ["substrates"].append(substrate_id)
                else:
                    self.broken_reaction_ids.append(reaction_id)
                    broken = True
                    break
            if broken:
                continue
            for product_verbose in product_verbose.split(' + '):
                product_id = self.compound_verbose_and_reaction_to_id(product_verbose,
                    reaction_id)
                if product_id is not None:
                    self.map_reaction_id_to_substrates_products_ids[reaction_id]\
                    ["products"].append(product_id)
                else:
                    self.broken_reaction_ids.append(reaction_id)
                    break
```

Figure 11: Obtaining compound ids from the human readable equation

Applying this procedure allows us to successfully retrieve the IDs for every substrate and product in an impressive 95.5% of the reactions, precisely 11,456 out of 11,991, with just 5 requests to the KEGG REST API server.

For the remaining 535 reactions marked as **broken**, we initiate GET requests using the sixth endpoint to retrieve compound IDs directly from the API results. Given our ability to request up to 10 reactions at a time, we accomplish this in 54 requests, bringing the total number of requests for the complete database creation needed to run our program to 59.

Remarkably, this efficient process enables us to construct the entire dataset in approximately 1 minute and 30 seconds, eliminating the risk of IP bans since we are making just a few requests.

We store the final database as a JSON file, weighing in at 3.22 MB, enabling our program to run seamlessly without requiring an internet connection.

10 Navigating the Design and User Experience of our Python GUI

Throughout our project, a key aspect of our development journey involved creating a straightforward Python GUI program. This application empowers users to effortlessly experiment with various models and seamlessly obtain insightful visualizations of hypergraphs. Additionally, users can leverage the GUI to benchmark different models, providing a user-friendly interface for comprehensive exploration.

Features at a Glance:

Model Trial: Easily try out different models with a user-friendly interface, allowing for quick experimentation and comparison.

Hypergraph Visualization: Gain valuable insights through simple visualizations of hypergraphs, providing a clear representation of complex structures.

Benchmarking Capabilities: Evaluate model performance efficiently by benchmarking against diverse datasets, aiding in the assessment and selection of the most suitable models.

In the following sections, we will delve into the design principles and functionalities of our Python GUI, offering guidance on its usage and showcasing its potential for enhancing your modeling experience.

10.1 Installing and Running the GUI

To kickstart your exploration of oriented hypergraphs with our Python GUI, follow these simple steps to install and run the program:

Step 1: Get the Code

Visit our GitHub repository at `https://github.com/ZephyrSV/tfg_python/` and clone or download the code to your local machine.

Step 2: Install python

Ensure that you have Python installed on your system. If not, download and install it from the official Python website: `https://www.python.org/downloads/`

Step 3: Install AMPL

Make sure you have at least the free community version of AMPL installed on your device. You can download it from the AMPL website: https://ampl.com/products/solvers/free-software/

Step 4: Set up the virtual environment Next, create a virtual environment (venv) to isolate dependencies for the GUI. Open your terminal or command prompt and navigate to the directory containing the program files. Execute the following commands:

```
# Create a virtual environment
python -m venv venv

# Activate the virtual environment
# On Windows
venv\Scripts\activate
# On Unix or MacOS
source venv/bin/activate
```

This creates and activates the virtual environment. Note: Depending on your terminal app, you may need to use a different activation script (e.g., 'activate.ps1' on Windows Powershell).

Step 5: Install Dependencies

While the virtual environment is active, install the required libraries using the following command:

```
pip install -r requirements.txt
```

This command installs the necessary dependencies for the GUI within the virtual environment.

Step 6: Run the GUI

With the virtual environment still active and AMPL installed, navigate to the directory containing the program files and run the following command:

```
python3 App.py
```

This command launches the Python GUI application.

10.2 Pathway Selector

The "Pathway Selector" serves as the central hub for navigating and interacting with our program.

Below, we show 2 screenshots.
The first one (Figure 12) is of the main page as it is shown to the user.
The second (Figure 13) shows the same image except that we have included labels for each feature present on the main page. We will then refer to these labels in the rest of the section.

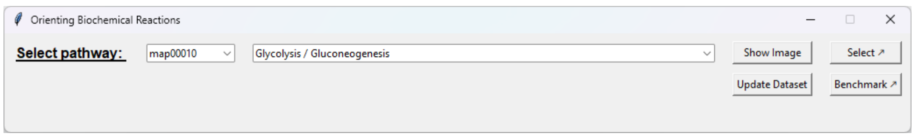

Select pathway: map00010 Glycolysis / Gluconeogenesis

Figure 12: Pathway Selector - User view

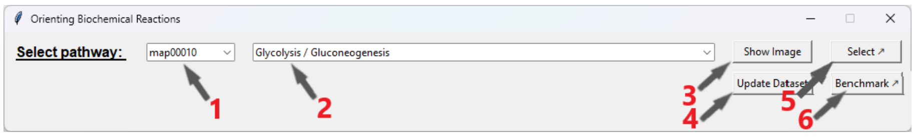


Figure 13: Pathway Selector - Labeled Features

Let's explore the features available on this main page:

Pathway Selection Options:

1. **Select by ID:** Utilize the dropdown menu to choose a pathway based on its unique identifier (ID). The user can also type in a search string in the field and pressing enter. The dropdown will then filter out all entries that do not contain the search string.
2. **Select by Description:** Alternatively, you can opt for pathway selection by choosing a human-readable description from the dropdown menu. In the same way as before, the user

Buttons and Actions:

3. **Show Image:** Clicking this button will fetch and display the image of the selected pathway from the KEGG database, providing a visual representation of its structure. See Figure 14 for an example.
4. **Update Dataset:** This button allows the user to rebuild the dataset used to create the models of the pathways.
5. **Select:** Access the solver/visualizer tool by clicking this button, enabling in-depth exploration and analysis of the chosen pathway.
6. **Benchmark:** Evaluate and compare model performances on 25 models, facilitating comprehensive assessments.

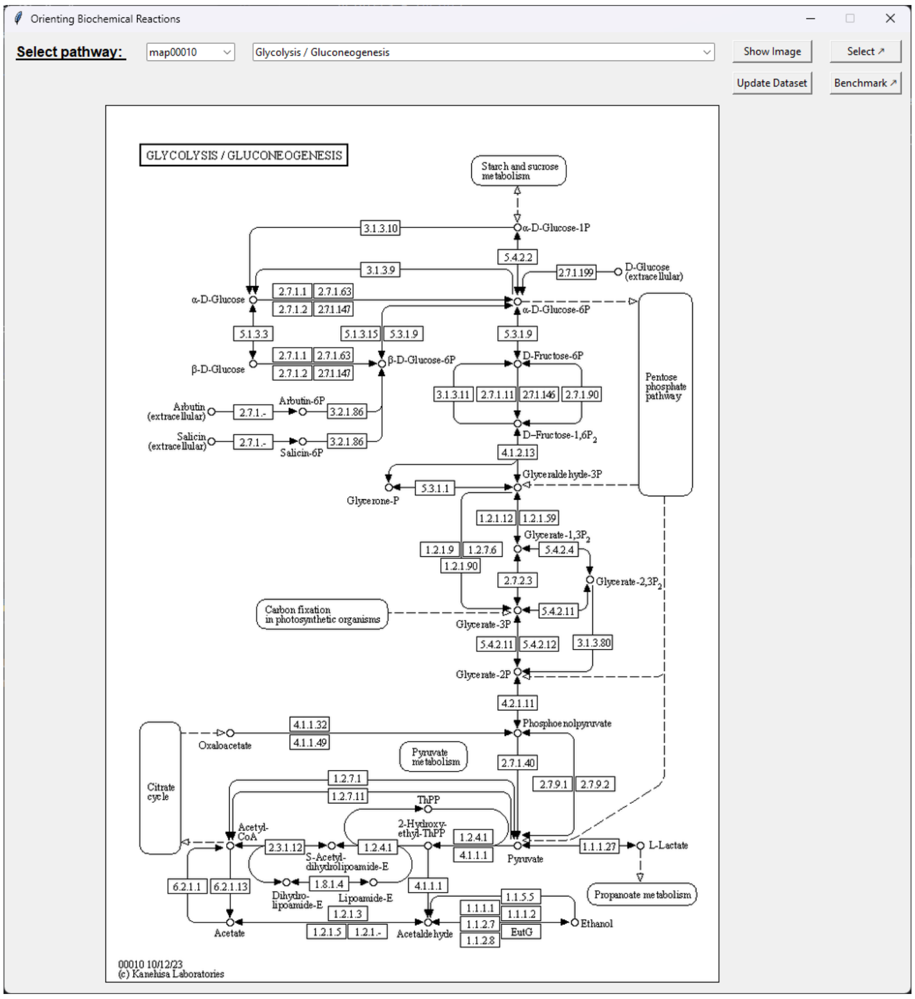
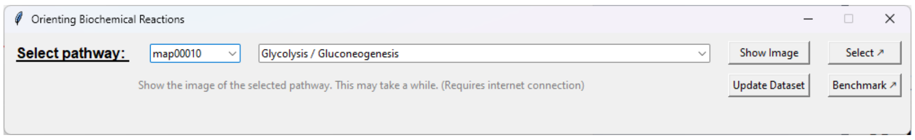

Orienting Biochemical Reactions
Select pathway:
map00010
Glycolysis / Gluconeogenesis
Show Image
Select ↗
Update Dataset
Benchmark ↗
GLYCOLYSIS / GLUCONEOGENESIS

Figure 14: Show Image - Image corresponding to the entry 'map00010'

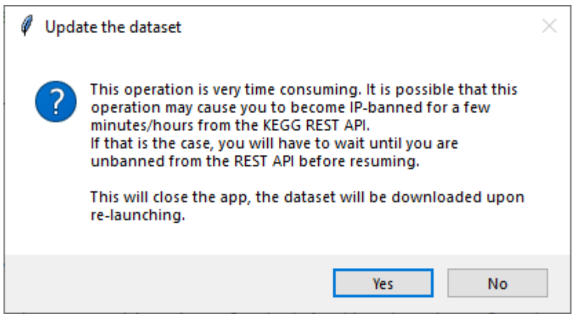
Note: Tooltips are available for each feature. Hover your mouse over any feature to view a brief explanation of its functionality. View Figure 15 for an example.



Show the image of the selected pathway. This may take a while. (Requires internet connection)

Figure 15: Feature tooltips - tooltip of the show image button

Upon pressing the **Update Dataset** button, a crucial warning message promptly emerges to alert the user about the potential duration of the operation, as depicted in Figure 16.



This operation is very time consuming. It is possible that this operation may cause you to become IP-banned for a few minutes/hours from the KEGG REST API.
If that is the case, you will have to wait until you are unbanned from the REST API before resuming.

This will close the app, the dataset will be downloaded upon re-launching.

Figure 16: Warning Message on Updating Dataset

The warning message reads as follows:

> *This operation is very time-consuming. It is possible that this operation may cause you to become IP-banned for a few minutes/hours from the KEGG REST API. If that is the case, you will have to wait until you are unbanned from the REST API before resuming.*
>
> *This will close the app, and the dataset will be downloaded upon re-launching."*

This cautionary message aims to ensure that users are fully aware of the potential consequences and the time commitment associated with updating the dataset. It emphasizes the possibility of temporary IP-banning and provides guidance on the necessary steps if such an event occurs.

10.3 Pathway Solver/Visualizer

The "Solver/Visualizer" view provides a comprehensive environment for in-depth exploration and analysis of the selected pathway. This view offers tools and features that empower users to interact with the oriented hypergraph representation and derive meaningful insights.

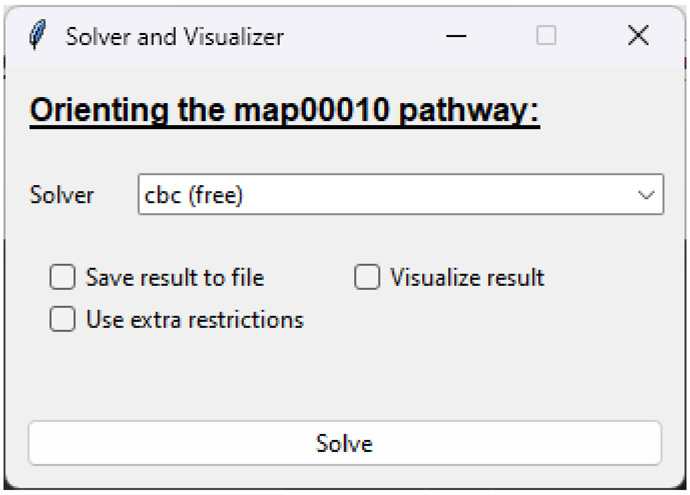


Figure 17: Solver/Visualizer - User View

Let's delve into the key components and functionalities of this view:

- **Solver Selection Dropdown:** Users can choose from a dropdown menu to select the solver they want to use (CBC, Gurobi or CPLEX).[4]
- **Save to File Tickbox:** A tickbox allowing users to specify whether to save the solver results to a file.
- **Visualize Result Tickbox:** Users can choose to visualize the solver results by ticking this option.
- **Use Extra Restrictions Tickbox:** This tickbox enables users to apply additional restrictions to the solver process if needed.
- **Solve Button:** The SOLVE button triggers the solving of the AMPL problem with the chosen solver and configurations.

Upon pressing the **Solve** button, a new instance of AMPL is created, the model and the pathway are loaded and after setting the solver, the optimization problem is completed.

The ensuing Figure 18 illustrates the output presentation:

[4]**Note:** CBC is marked as free, as you don't need a licence to use it. Unlike Gurobi and CPLEX.

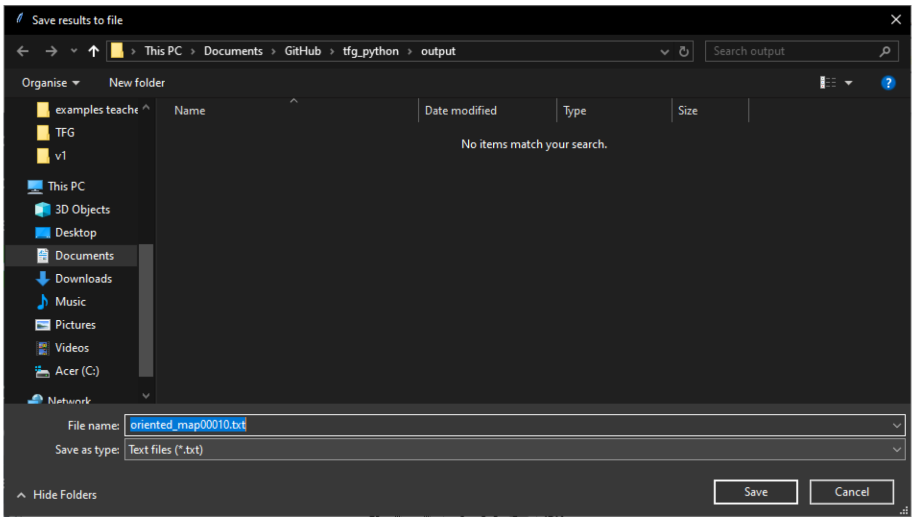
```
###############################################
### Computed in 0.1814289093017578 seconds
### Number of internal vertices  28
###############################################

### reactionID : [Substrates] : [Products]

### Reactions that were not inverted
R01122 : C00235 C17324 : C00013 C04432
R02119 : C00190 C00371 : C00015 C03300
R03133 : C00979 C00371 : C01513 C00033
R03726 : C00019 C04083 : C00170 C01804
R04038 : C00235 C00020 : C00013 C04713
R05708 : C04083 C00001 C00028 : C00147 C07330 C00030
R07260 : C00029 C15545 : C00015 C15546
R08051 : C00235 C00002 : C16424 C00013
R08053 : C16424 C03024 C00007 : C16428 C03161 C00001
R08054 : C16426 C03024 C00007 : C16429 C03161 C00001
R08055 : C04713 C03024 C00007 : C16430 C03161 C00001
R08061 : C16424 C00001 : C16426 C00009
R08062 : C16426 C00001 : C04713 C00009
R08063 : C16428 C00001 : C16429 C00009
R08064 : C16429 C00001 : C16430 C00009
R08065 : C16430 : C16445
R08066 : C16427 C03024 C00007 : C16431 C03161 C00001
R08067 : C16431 : C16447
R08070 : C16430 C00001 : C16431 C00009
R08071 : C16445 C00001 : C16447 C00009
R08073 : C16431 C00001 : C00371 C00121
R08074 : C16447 C00001 : C02029 C00121
R08075 : C00371 C00029 : C16443 C00015
R08076 : C02029 C00029 : C16448 C00015
R08077 : C04432 : C16467
R08078 : C16467 : C16441
R08079 : C16441 C00001 : C16449 C00009
R08080 : C16449 C00001 : C15545 C00121
### Reactions that were inverted
R02118 : C00015 C03423 : C00029 C00371
R05702 : C00371 C00005 C00080 : C02029 C00006
R08052 : C16426 C00013 : C00235 C00008
R08068 : C00371 C03161 C00001 : C04083 C03024 C00007
R08069 : C16427 C00009 : C04713 C00001
R08072 : C04083 C00121 : C16427 C00001
```

Figure 18: Output for map00908 - Zeatin biosynthesis

By default, the output is directed to the console. However, when the **Save to file** tickbox is selected, the program prompts the user to designate a location for saving the output, as depicted in Figure 19.



This PC › Documents › GitHub › tfg_python › output

No items match your search.

File name: oriented_map00010.txt

Save as type: Text files (*.txt)

Figure 19: Solver-Visualizer - Saving the result in a file

f the user opts for the **Visualize results** tickbox, a representation of the model, generated using the Python library networkx is presented. This representation utilizes a graph to depict our hypergraph. The transformation from a hypergraph to a graph is achieved through the following mathematical procedure:

Procedure. Representing a hypergraph using a graph.

Given an hypergraph H ={V, E}, where V is the set of **vertices** and E of **hyperedges**, we create the graph G ={$V \bigcup E, E_T \bigcup E_H$}, where:

$$E_T = \{\{u, \{T, H\}\} \mid u \in V, \{T, H\} \in E, u \in T\}$$

$$E_H = \{\{u, \{T, H\}\} \mid u \in V, \{T, H\} \in E, u \in H\}$$

We give an example of applying this procedure in Figure 20.

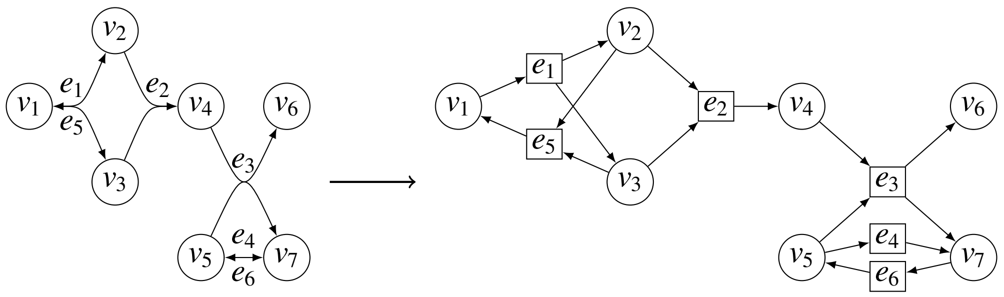


Figure 20: An example of how to represent a hypergraph using a graph

In our GUI, the graph visually distinguishes various elements: **external compounds** appear as red dots, **internal compounds** as green dots, and **reactions** as grey squares. Hyperedges with inverted orientation are uniquely identified by coloring their edges blue. A practical demonstration of this visualization is presented in the form of a screenshot in Figure 21.

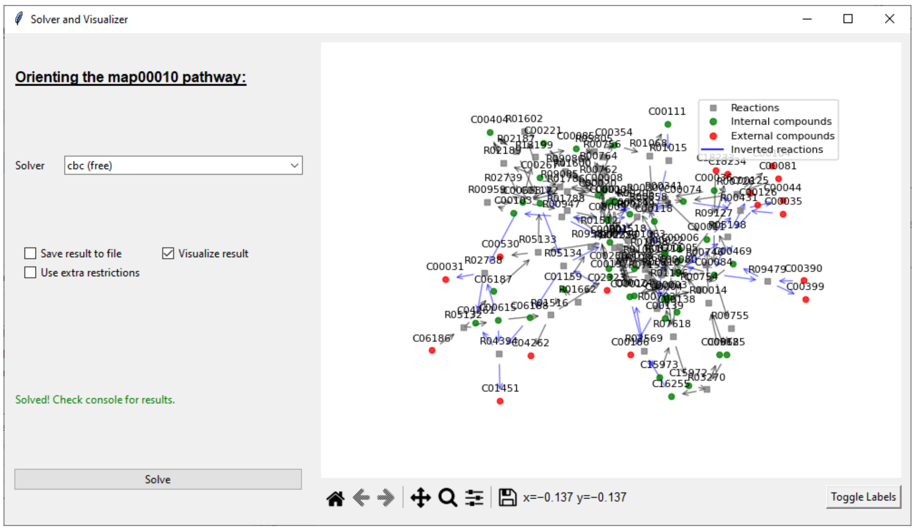


Figure 21: Solver-Visualizer - Visualizing a oriented pathway

In addition to the graphical representation, a Matplotlib-style toolbar is situated at the bottom right of the screen. This toolbar facilitates functionalities such as zooming in, moving the graph, and saving a screenshot of the graph. Furthermore, a dedicated button enables the toggling of labels on and off, enhancing visibility, especially in scenarios where nodes are densely clustered. The accompanying screenshot (Figure 22) is the output of pressing the save button on the same pathway with improved visibility, achieved by activating the **Toggle Labels** button.

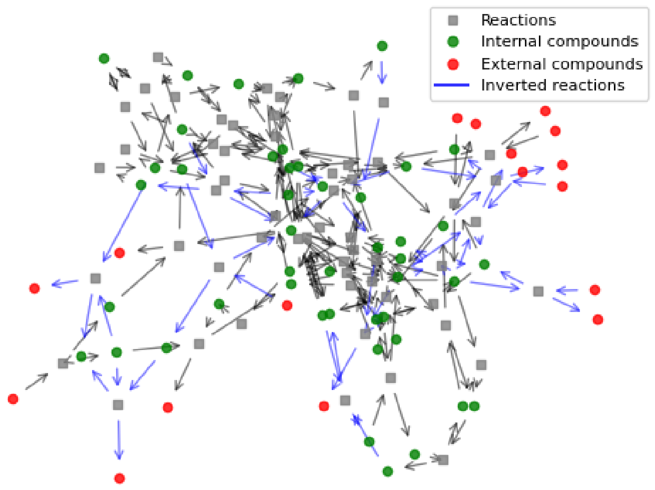
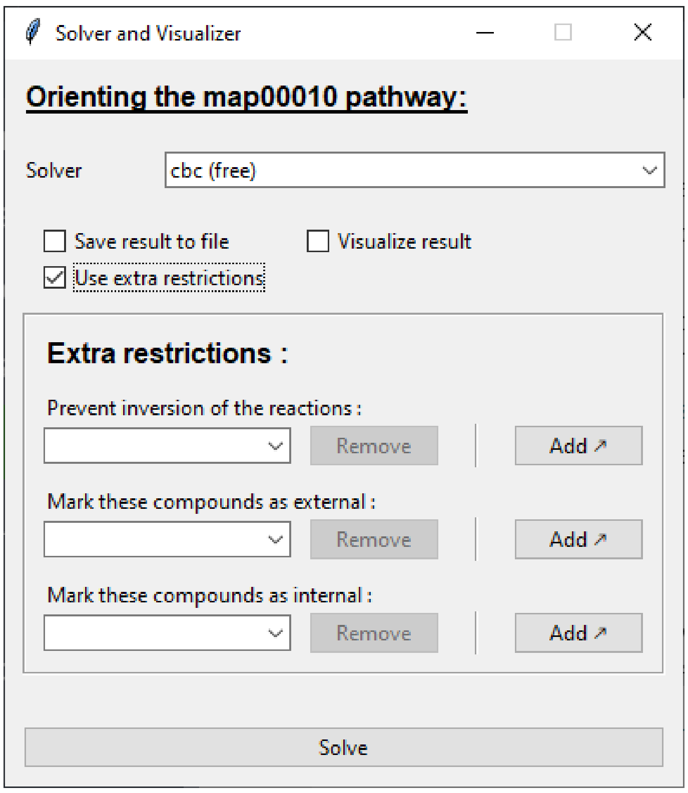


Figure 22: Solver-Visualizer - Visualizing a oriented pathway, with labels toggled off

Upon activating the **Use extra restrictions** tickbox, a new menu surfaces. This menu empowers the user to establish additional constraints, as elaborated in Section 8.6.5 through to Section 8.6.7. Within this menu, users can prevent the inversion of specific reactions (or hyperedges), or classify particular compounds as external or internal. This feature provides a finer degree of control over the orientation and classification of elements in the hypergraph.



Solver and Visualizer

Orienting the map00010 pathway:

Solver: cbc (free)

☐ Save result to file ☐ Visualize result
☑ Use extra restrictions

Extra restrictions :

Prevent inversion of the reactions :

Mark these compounds as external :

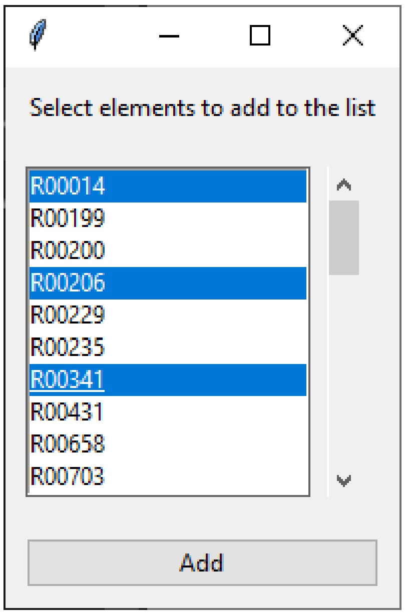
Mark these compounds as internal :

Figure 23: Solver-Visualizer - The 'Extra restrictions' menu

By default, each of the extra restriction fields begins empty.

To introduce a restriction, for example: preventing the inversion of reactions, users can click on the **Add** button.
This action triggers a popup window displaying a list of all potential values.
In the case of our example, this list comprises reactions not yet marked as un-invertible.

Following the popup invocation, users can select one or more items to incorporate into the restriction list. As demonstrated in Figure 24, an example showcases a user adding the reactions 'R00014', 'R00206', and 'R00341'.



Select elements to add to the list

- R00014 (selected)
- R00199
- R00200
- R00206 (selected)
- R00229
- R00235
- R00341 (selected)
- R00431
- R00658
- R00703

Figure 24: Solver-Visualizer - Adding items to a list of a restriction

Upon successful addition, the items are stored in the corresponding dropdown list. Figure 25 illustrates the successful incorporation of items into the list.

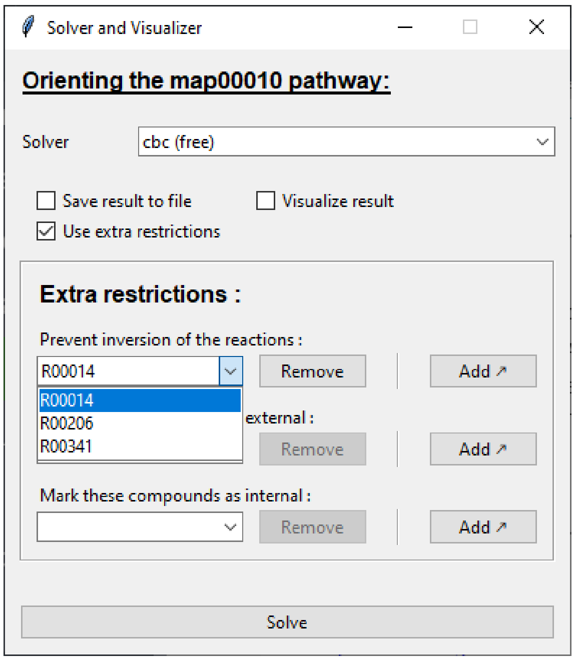


Figure 25: Solver-Visualizer - Items added to a list of a restriction

To remove an element from a restriction list, users can select the desired element from the corresponding dropdown menu and activate the **Remove** button. Figure 26 visually depicts the list after the user has pressed the **Remove** button, demonstrating the successful removal of an element from the restriction list.

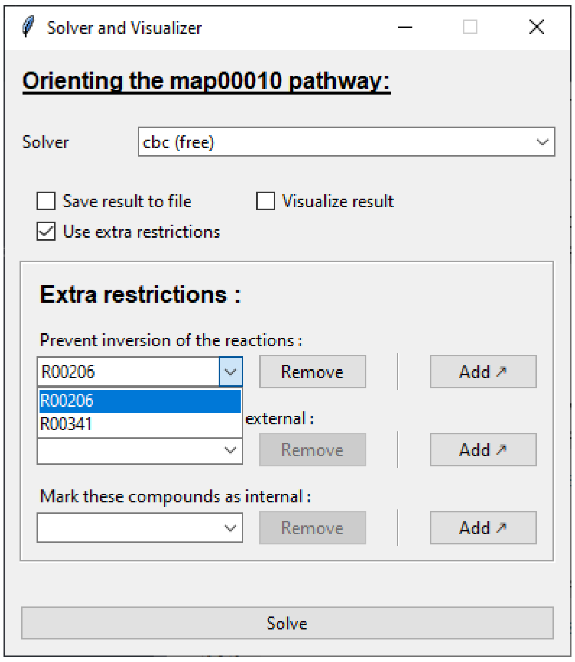


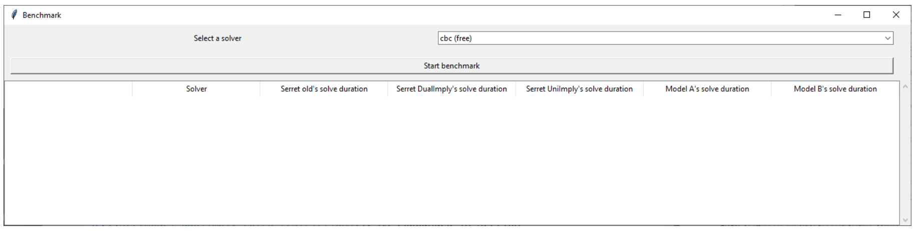
Figure 26: Solver-Visualizer - Item removed from a list of restrictions

During the optimization problem of orienting the pathway, we employ the AMPL model 'Model A' (refer to Section 8.3.2), which, as demonstrated in Section 8.7, proves to be the most efficient model.

However, should the user opt for extra restrictions, we pivot to utilizing the model 'Serret DualImply extra restrictions' (refer to Section 8.5). As expounded in Section 8.6, this model is specifically designed to accommodate the additional constraints, as the mathematical robustness of the other model is insufficient for supporting these extra restrictions.

10.4 Benchmark View

Upon initiating the "Benchmark View," users encounter a sleek and user-friendly interface, as showcased in Figure 27. Key components of this interface include a dropdown menu, enabling users to choose from solvers such as CBC, Gurobi, or CPLEX, and a prominent "Start Benchmark" button. The purpose of this thoughtful design is to simplify the benchmarking process, ensuring a seamless experience for users evaluating solver performance across a spectrum of optimization scenarios.

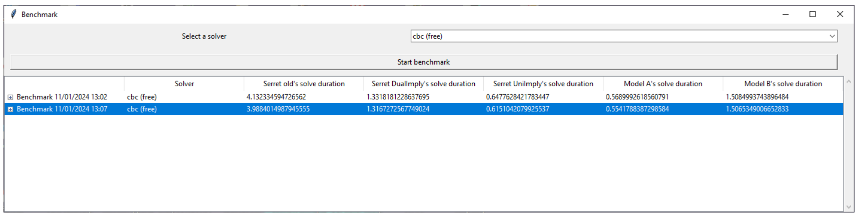


	Solver	Serret old's solve duration	Serret DualImply's solve duration	Serret UniImply's solve duration	Model A's solve duration	Model B's solve duration

Figure 27: Benchmark View - User Interface

Upon triggering the "Start Benchmark" button, the benchmarking procedure commences. Each model, namely 'Model A,' 'Model B,' 'Serret Old,' 'Serret UniImply,' and 'Serret DualImply,' undergoes 25 optimizations. Subsequently, the results are presented in a Tk.TreeView below, offering users a comprehensive overview of solver performance across various models and optimizations.

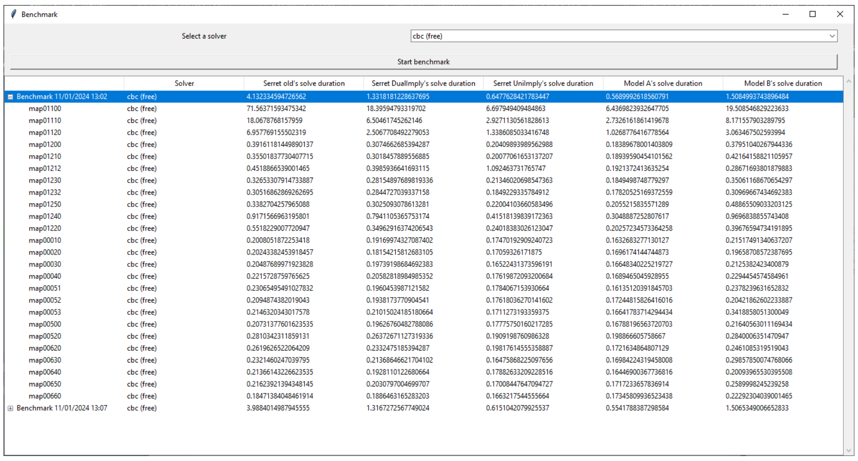
The benchmarking process may take up to 4 minutes. Figure 28 illustrates the view after the completion of 2 benchmarks, each providing information on the date and time of completion, the utilized solver, and the average time taken by each model to solve individual optimization problems.



	Solver	Serret old's solve duration	Serret DualImply's solve duration	Serret UniImply's solve duration	Model A's solve duration	Model B's solve duration
⊞ Benchmark 11/01/2024 13:02	cbc (free)	4.132334594726562	1.3318181228637695	0.6477628421783447	0.5689992618560791	1.5084993743896484
⊞ Benchmark 11/01/2024 13:07	cbc (free)	3.9884014987945555	1.3167272567749024	0.6151042079925537	0.5541788387298584	1.5065349006652833

Figure 28: Benchmark View - Completed Benchmarks

For a more detailed analysis, users have the option to expand the benchmark report and delve into the completion details of each of the 25 entries, as demonstrated in Figure 29.



	Solver	Serret old's solve duration	Serret DualImply's solve duration	Serret UniImply's solve duration	Model A's solve duration	Model B's solve duration
⊟ Benchmark 11/01/2024 13:02	cbc (free)	4.132334594726562	1.3318181228637695	0.6477628421783447	0.5689992618560791	1.5084993743896484
map01100	cbc (free)	71.56371593475342	18.39594793319702	6.697949409484863	6.4369823932647705	19.508546829223633
map01110	cbc (free)	18.0678768157959	6.50461745262146	2.9271130561828613	2.7326161861419678	8.171557903289795
map01120	cbc (free)	6.957769155502319	2.5067708492279053	1.3386085033416748	1.0268776416778564	3.063467502593994
map01200	cbc (free)	0.39161181449890137	0.3074662685394287	0.20409893989562988	0.18389678001403809	0.37951040267944336
map01210	cbc (free)	0.35501837730407715	0.3018457889556885	0.20077061653137207	0.18939590454101562	0.42164158821105957
map01212	cbc (free)	0.4518866539001465	0.3985936641693115	1.092463731765747	0.1921372413635254	0.28671693801879883
map01230	cbc (free)	0.32653307914733887	0.28154897689819336	0.21346020698547363	0.1849498748779297	0.35061168670654297
map01232	cbc (free)	0.30516862869262695	0.2844727039337158	0.1849229335784912	0.17820525169372559	0.30969667434692383
map01250	cbc (free)	0.3382704257965088	0.3025093078613281	0.22004103660583496	0.2055215835571289	0.48865509033203125
map01240	cbc (free)	0.9171566963195801	0.7941105365753174	0.41518139839172363	0.3048887252807617	0.9696838855743408
map01220	cbc (free)	0.5518229007720947	0.34962916374206543	0.24018383026123047	0.20257234573364258	0.39676594734191895
map00010	cbc (free)	0.2008051872253418	0.19169974327087402	0.17470192909240723	0.1632683277130127	0.21517491340637207
map00020	cbc (free)	0.20243382453918457	0.18154215812683105	0.17059326171875	0.1696174144744873	0.19658708572387695
map00030	cbc (free)	0.20487689971923828	0.19739198684692383	0.16522431373596191	0.16648340225219727	0.2125382423400879
map00040	cbc (free)	0.2215728759765625	0.20582818984985352	0.17619872093200684	0.1689465045928955	0.2294454574584961
map00051	cbc (free)	0.23065495491027832	0.1960453987121582	0.1784067153930664	0.16135120391845703	0.2378239631652832
map00052	cbc (free)	0.2094874382019043	0.1938173770904541	0.17618036270141602	0.17244815826416016	0.20421862602233887
map00053	cbc (free)	0.2146320343017578	0.21015024185180664	0.1711273193359375	0.16641783714294434	0.3418858051300049
map00500	cbc (free)	0.20731377601623535	0.19626760482788086	0.17775750160217285	0.16788196563720703	0.21640563011169434
map00520	cbc (free)	0.2810342311859131	0.26372671127319336	0.1909198760986328	0.198866605758667	0.2840006351470947
map00620	cbc (free)	0.2619626522064209	0.2332475185394287	0.19817614555358887	0.1721634864807129	0.2461085319519043
map00630	cbc (free)	0.2321460247039795	0.21368646621704102	0.16475868225097656	0.16984224319458008	0.29857850074768066
map00640	cbc (free)	0.21366143226623535	0.1928110122680664	0.17882633209228516	0.16446900367736816	0.20093965530395508
map00650	cbc (free)	0.21623921394348145	0.2030797004699707	0.17008447647094727	0.1717233657836914	0.2589998245239258
map00660	cbc (free)	0.18471384048461914	0.1886463165283203	0.1663217544555664	0.17345809936523438	0.22292304039001465
⊞ Benchmark 11/01/2024 13:07	cbc (free)	3.9884014987945555	1.3167272567749024	0.6151042079925537	0.5541788387298584	1.5065349006652833

Figure 29: Benchmark View - Expanded Benchmarks