RWTH Aachen University

Faculty of Business and Economics

Project Report: Advanced Data and Business Analytics

Development of an user-friendly interface using GAMS and GAMS MIRO

Author and matriculation number:

David Sanders (395334)

and

Noah Schuster (395949)

August 22st, 2024

Advisors:

Dr. Lorena Reyes-Rubiano
Chair of Data and Business Analytics
RWTH Aachen University

David Sanders and Noah Schuster:
Development of an user-friendly interface using GAMS and GAMS MIRO
Project Report: Advanced Data and Business Analytics, RWTH Aachen University, 2024.

Abstract

The Network Design Problem is a critical optimization challenge, often applied in the context of physical networks like transportation. This paper presents the development of a user-friendly interface utilizing the General Algebraic Modeling System (GAMS) and its visualization extension, GAMS MIRO. The interface aims to enhance the usability and accessibility of complex optimization models by integrating data visualization and interactivity features. The interface was specifically tailored for the Network Design Problem, allowing users to adjust inputs flexible and compare the output form various predefined scenarios efficiently. Key results are presented through intuitive renderings that facilitate comprehension and evaluation of results. The project highlights the effectiveness of integrating MIRO within GAMS, making it a valuable tool for both educational purposes and real-world applications. Additionally, the project lays the foundation for more complex dashboards and tools for network optimization, which represent a valuable extension to network planning in both research and business contexts.

Contents

List of Figures

List of Tables

1. Introduction

The introduction is divided into two sections. In **Section 1.1** the general motivation and the given problem statement are presented , from which the objective of the paper is derived. This is followed by an overview of the structure of the paper in **Section 1.2**.

1.1 Motivation and Problem Statement

Optimization problems are mathematical challenges wherein the objective is to identify the optimal solution from a set of feasible solutions. This optimal solution either maximizes or minimizes a given objective function while adhering to specific constraints or limitations. Optimization problems are applicable in a diverse range of disciplines, including economics and finance, engineering, logistics, network planning, and many others. The outcome of optimization serves as a crucial foundation for strategic decisions regarding resource allocation and future development in an entrepreneurial, social or governmental context.

A significant issue with optimization results is that they are often presented in purely numerical form, which can be challenging for decision-makers to interpret. To enhance comprehension, it is essential to employ data visualization techniques. Data visualization not only improves the understanding of complex data but also facilitates better communication of results to stakeholders. This approach offers significant advantages across various domains. For instance, in educational settings, it enhances students' understanding of optimization tasks. Likewise, in management contexts, it facilitates the translation of optimization results into a more accessible format, thereby supporting strategic decision-making.

Another important aspect of optimization is the ability to compare different scenarios. Traditional methods within programming environments often result in static comparisons, which can be both limiting and labor-intensive to implement. A more effective approach involves generating and comparing scenario outputs interactively within a more advanced environment, such as a dashboard or interface. This method allows

for quicker and more flexible scenario definitions and facilitates easier interpretation through visual representation.

This leads to the main objective of the work:

> **Creation of an user-friendly interface that addresses the opportunities and challenges in visualization and interactivity.**

The interface should enable users to adjust inputs interactively and present outputs in a clear and comprehensible format. Additionally, it should enhance the realism of the optimization problem, thereby bringing it closer to the user.

In this context, the specific optimization task being addressed is the Network Design Problem, which will be detailed further in the paper. The chosen tool for this task is GAMS (General Algebraic Modeling System). GAMS is a powerful modeling system that distinguishes itself from other optimization software by providing robust modeling and optimization capabilities while integrating data visualization natively within the software. The visualization component of GAMS is called GAMS MIRO, which unifies all the necessary functions for solving the tasks efficiently, making it an optimal choice for this application.

1.2 Paper Structure

The paper is structured into five Chapers. After the motivation and objectives were outlined in **Chapter 1**, **Chapter 2** presents the mathematical foundation necessary for the subsequent implementation of the optimization models utilized. This chapter begins by explaining the Network Design Problem and briefly addresses the Minimum Spanning Tree, a specific outcome of the Network Design Problem that will be required later. **Chapter 3** focuses on the actual implementation of the GAMS MIRO interface. It starts with the definition of the target image and then considers the work packages required for implementation. The following six work packages were carried out:

- **Data Management:** Adapting input data to enhance realism and further optimization possibilities.
- **Input Features:** Establishing user input capabilities so that input data can be flexibly adjusted within the interface.
- **Output Features:** Developing clear and informative output visualizations to efficiently communicate optimization results to the user.
- **Aggregation of Input and Output Features:** Aggregation of all developed input and output features into a coherent structure.
- **Scenarios:** Defining and implementing pre-configured scenarios to provide the user comparison options.
- **MIRO App:** Export the final MIRO app to make the program available independently of the development environment.

With the implementation details clarified, **Chapter 4** provides practical instructions on using the interface. It explains how to adjust input data and interpret output data. Additionally, it shows the user to create and include scenarios. Finally, **Chapter 5** summarizes the key points, defines the scope of the interface, and provides an outlook on potential future developments.

2. Mathematical Background: Optimization Models

Chapter 2 provides a detailed explanation of the Network Design Problem. It begins with a qualitative description of the objective of the network design problem and its applications, in **Section 2.1**. Following this, the underlying mathematical calculations are discussed to give a comprehensive understanding of the model's functioning. This ensures that the end user can critically evaluate the results obtained from using the optimization model. Additionally, **Section 2.2** explains the mathematical concepts of the Minimal Spanning Tree, which is a specific outcome of the Network Design Problem.

2.1 Introduction Network Design Problem

The Network Design Problem pertains to the optimal design and configuration of networks. This primarily involves physical networks such as transportation and supply networks. To facilitate understanding of the problem, a general and simple example is provided, as illustrated in Figures 2.1 and 2.2. This example aim to make the problem easier to comprehend.

2.1.1 Qualitative Explanation

In reality, it is rare to design a network from scratch. Therefore, it is often necessary to optimize an existing network. In Figure 2.1 such a given network is depicted. It is assumed that all possible connections between the nodes are currently represented in this network. In addition to visualizations like the one in Figure 2.1, this network can also be described using sets and parameters, which is crucial for the mathematical model discussed later. In the following, the definitions are given:

Sets

- V: sorted nodes (indices: i, j, u, v)
- E: edges; with the edge $[i, j] \in E$, where $i < j$ holds

Upon examining the example in Figure 2.1, the nodes symbolize various cities within the network, while the edges denote the potential connections that can be established.

Parameters

- $d(uv)$: expected number of passengers traveling from node u to node v, where $v \neq u$.
- If $v = u$, $d(uv)$ represents the expected number of passengers leaving node u.
- $t(ij)$: travel time for the direct journey from node i to node j with $[i, j] \in E$ and $t(ij) = t(ji)$.
- $c(ij)$: fixed costs per period for constructing an edge that directly connects nodes i and j.
- B: maximum allowed total costs, where $B \geq F \cdot 1$.

Continuing the examination of the initial example network depicted in Figure 2.1, the construction costs $c(ij)$ and the travel time $t(ij)$ are displayed on the respective edges in the format ($c(ij)$ [€], $t(ij)$[h]). The parameters $d(uv)$ and B[€], though not visible in the network, constitute additional input values essential for applying the optimization model.

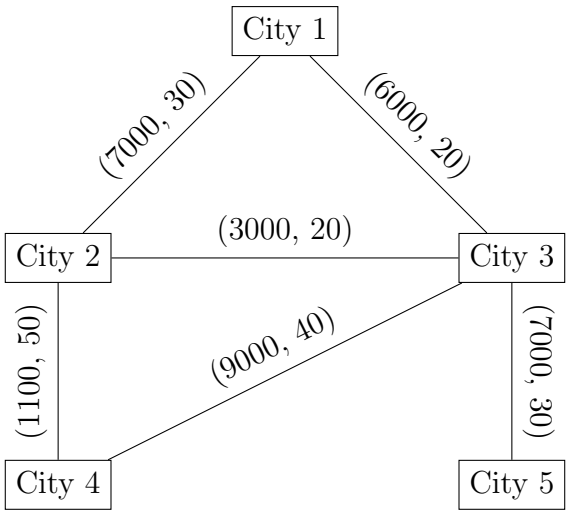


Figure 2.1: Example Network with all possible connections

Qualitatively, it can be noted that the model determines a new network from the initially given network with the goal of minimizing the total travel time of passengers while considering relevant constraints. An exemplary output of this optimization model is shown in Figure 2.2. It can be observed that not all possible connections were constructed. The connection between City 2 and City 4 was removed because its removal led to an optimization of the objective function under the given constraints. Such a result can now be mathematically described by the following objective value and variables:

Objective Value

- G: Total travel time of all passengers $[h]$

Decicion Variables

- x_{uij}: number of passengers starting their trip in node u and traveling from node i to node j with $[i, j]$ in E
- y_{ij}: 1 if edge $[i, j]$ in E is built (0, otherwise)

In Figure 2.2, the optimization model determines the decision variables for each edge within the network. The binary variable y_{ij} if the connection between two cities has been built, and 0 otherwise. The output in Figure 2.2 shows that y_{ij} is 1 for each connection except for the connection y_{24} or y_{42}. In the given example, the variable x_{uij} is calculated five times since there are five possible cities where passengers can start their trip. Depending on those origins u , the network variable x_{uij} indicates the number of passengers traveling between two cities i and j.

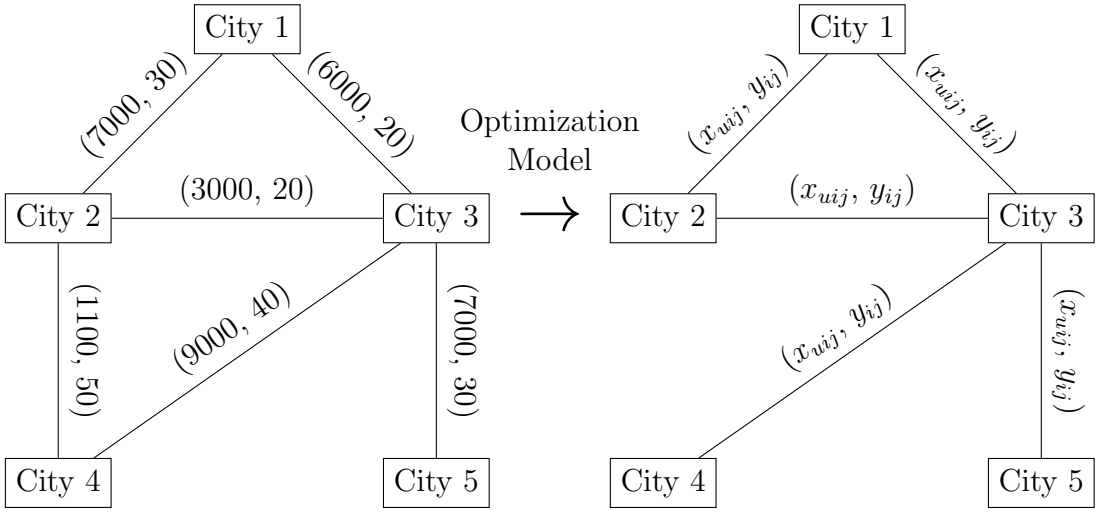


Figure 2.2: Example Network before and after applying the Optimization Model

To gain a deeper understanding of the mathematical process behind network optimization, the underlying optimization model will be systematically explained in the next section.

2.1.2 Mathematical Optimization Model

In this section, the optimization model is mathematically presented and explained. The focus will be on the objective function as well as the five constraints.

$$\text{Minimize } G = \sum_{[i,j]\in E} t_{ij} \cdot \sum_{u\in V} X_{uij} \tag{1}$$

such that:

$$\sum_{[i,v]\in E} X_{uiv} - \sum_{[v,j]\in E} X_{uvj} = d_{uv} \quad \forall u, v \in V \mid u \neq v \tag{2}$$

$$X_{uij} + X_{uji} \leq d_{uu} \cdot Y_{ij} \quad \forall u \in V, [i,j] \in E \mid i < j \tag{3}$$

$$\sum_{[i,j]\in E|i<j} c_{ij} \cdot Y_{ij} \leq B \tag{4}$$

$$X_{uij} \geq 0 \quad \forall u \in V, [i,j] \in E \tag{5}$$

$$Y_{ij} \in \{0,1\} \quad \forall [i,j] \in E \mid i < j \tag{6}$$

1. **Objective Function** is designed to minimize the total travel time for all passengers by summing the product of travel time and the number of passengers for all possible connections.
2. **Flow Balance Constraint** ensures that the difference between the number of passengers leaving node v and the number of passengers arriving at node v equals the expected number of passengers traveling from u to v.
3. **Connection Usage Constraint** ensures that passengers can only use connection ij if it has been built. Therefore, a so-called Big M constraint is used. In this case, d_{uu} is the Big M, chosen because it is always greater than or equal to the sum of passengers starting their trip in u and traveling from i to j in both directions ($x_{uij} + x_{uji}$).
4. **Budget Constraint** ensures that the total construction costs do not exceed the budget.
5. **Non-Negativity Constraint** ensures that the variable x_{uij} is always greater than or equal to 0.
6. **Binary Constraint** ensures that the variable y_{ij} is 1 if the edge between two cities is built and 0 otherwise.

2.2 Minimal Spanning Tree (MST)

A Minimal Spanning Tree (MST) is a fundamental concept in graph theory and network design. It refers to a specific type of subset of the edges in a connected, undirected graph. This subset, designated as T, connects all the vertices together without any cycles and with the minimum possible total edge weight. The concept is crucial in various applications, including the design of efficient communication networks, electrical circuits, and transportation systems.

Formally, given a connected, undirected graph $G = (V, E)$ with a weight function $w : E \to \mathbb{R}$ that assigns a real number to each edge, the MST T is defined as a spanning tree $T \subseteq E$ that minimizes the sum of the weights of the edges in the tree. In mathematical terms, the MST is defined as:

$$T = \arg\min_{T' \subseteq E} \sum_{e \in T'} w(e)$$

where T' is any spanning tree of the graph G. Here, T represents the tree within the graph that satisfies these conditions, ensuring the connection of all vertices with minimized total edge weight.

To further illustrate the concept of the MST, Figure 2.3 depicts an MST for the network used in this section so far. We assume that the edge weights are calculated based on the construction costs c_{ij}.

This concept is also central to the application discussed in this paper, where the Optimization Model for Network Design becomes infeasible if a budget is chosen that is smaller than the necessary budget for constructing the Minimal Spanning Tree. To ensure a feasible solution, the lower bound for the budget is set to the minimum amount required to construct the Minimal Spanning Tree.

What this means in our application is detailed in Chapter 3, where Kruskal's algorithm is used to determine the MST.

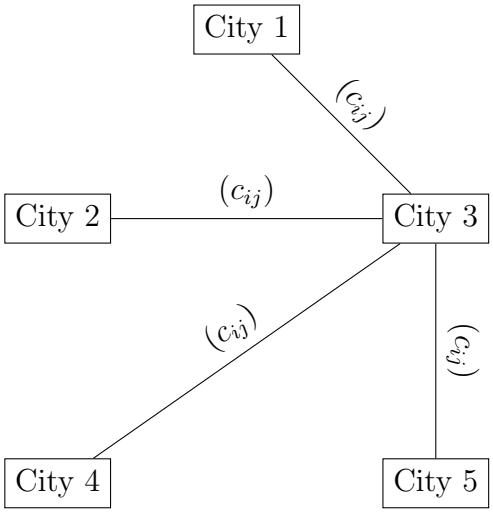


Figure 2.3: Example of a Minimal Spanning Tree

One of the algorithms that can be used to find the Minimal Spanning Tree is Kruskal's algorithm. This algorithm is exemplarily outlined below:

Algorithm 1 Kruskal Algorithm

Prerequisite: A weighted, contiguous, loop-free, undirected graph $G = [V, E, c]$ with n nodes and m edges; with set $\bar{E}$ edge quantity to be determined with minimal spanning tree $T = [V, \bar{E}]$.

1: **Start:** Sort or number the edges k_i of G in order $k_1, k_2, \ldots, k_m$ after non-decreasing weights $c(k_i)$, so that holds: $c(k_1) \leq c(k_2) \leq \ldots \leq c(k_m)$.
2: Set $\bar{E} := \emptyset$ and $T := [V, \bar{E}]$.
3: **for** $\mu = 1, 2, \ldots, m$ **do**
4: choose edge k_μ and check if its inclusion in $T = [V, \bar{E}]$ creates a cycle.
5: **if** k_μ does not create a cycle **then**
6: set $\bar{E} := \bar{E} \cup \{k_\mu\}$.
7: next iteration.
8: **Termination:** The procedure terminates as soon as $\bar{E}$ contains $n - 1$ edges.
9: **Result:** $T = [V, \bar{E}]$ is a minimal spanning tree of G.

The foundational concepts discussed in this chapter serve as the basis for understanding the subsequent implementation steps in GAMS MIRO.

3. Implementation of GAMS MIRO Interface

The following chapter constitutes the core of this work. Initially, **Section 3.1** provides an overview of the interface's target image, using the initial situation as the foundation. **Section 3.2** adresses the specific implementation work packages that were needed to achieve the desired project goal.

3.1 Desired Target Image of the Interface

This section is dedicated to explaining the initial conditions and the objectives set for this project. The depiction of the Network Design Problem within GAMS and GAMS MIRO was initially broadly conceptualized, lacking the necessary detail and precision. It is important to acknowledge that the initial situation presented several deficiencies that needed addressing.

3.1.1 Initital Situation

The following bullets outline the main points that describe the initial situation:

- **Mathematical Model with Incomplete Code Base:** The code provided initially was a raw version of the network design optimization model explained in Section 2. Based on this, it was not possible to create visualizations and interactive options to enhance user experience with the interface.
- **Missing Dashboard:** The initial setup lacked graphical visualizations and only provided default data tables as input and output. This made it difficult to understand what the output represented, as it was hard to imagine the resulting network solely from decision variables presented in a table.
- **Lack of Real-World Relevance:** The set of nodes was limited to five, named City 1 through City 5, without any geographical or demographic information about the cities. This limited the real-world applicability of the model.

- **No Scenario Planning:** Users had to create each scenario independently without guidance on which input settings were useful and would lead to reasonable results. This made the process cumbersome and less user-friendly.

3.1.2 Target Image

As briefly mentioned in the introduction, the goal of this project is to develop a user-friendly and interactive interface utilizing GAMS and GAMS MIRO. This interface aims to facilitate the optimization of a train network by incorporating real-world data and providing an intuitive user experience.

The following section explains what was needed to progress from the initial situation to the desired target outcome:

- **Visualizations of Key Results:** Key results are displayed using suitable and intuitive graphic visualizations, ensuring accessibility and comprehensibility for users.
- **Enabling Interactivity:** Interactivity is a core feature of the dashboard, which supports the modification of input data directly within the dashboard environment.
- **Real-World Relevance of the Model:** The model utilizes the largest German cities as nodes to mirror real-world scenarios accurately. Connections between these nodes represent train tracks, aiming to optimize a train network. The interface accommodates selection among different train types, accounting for the diversity in travel times and reflecting the variety inherent in actual train networks.
- **Predefined Scenarios and Flexibility:** The interface presents a selection of predefined scenarios, crafted to be relevant and conducive to analysis. This feature facilitates the rapid selection of useful input settings. Additionally, the interface permits the adjustment of input data, providing customization options alongside convenience.
- **Appealing and Intuitive Design:** The design of the interface prioritizes appeal and intuitiveness to encourage user engagement. It simplifies the process of input selection and is optimized to enable users to easily derive all pertinent information from the output visuals, thus enhancing both the user experience and the tool's effectiveness.

3.2 Implementation Work Packages

To achieve the desired target while considering the initial situation, six work packages were defined to reach the project outcome. These work packages include Data Management, Input Features, Output Features, Aggregation of Input and Output Features, Scenarios and MIRO App. The subsequent sections will provide detailed explanations of these steps.

3.2.1 Data management

The initial database on which the optimization model was built needs to be expanded, adapted, and reorganized to meet the requirements necessary to achieve our target outcome, as defined in the previous section. In the following paragraphs, the main aspects for defining the target picture are listed.

Increasing the Number of Cities and Selecting Actual Existing Cities

Initially, the set of cities was limited to five distinct cities, labeled City 1 through City 5. This provided users with a very limited selection of cities. Additionally, the generic labeling and lack of geographical references necessitated a revision of this set of cities.

```
1 set v /city1, city2, city3, city4, city5/;
```

To enable the consideration of a wider set of cities, first a structured approach to select a useful set of cities that would lead to reasonable solutions needed to be defined. Therefore, different requirements to ensure a systematic process for determining the set of cities from which the user can choose were established. The following assumptions were made for city selection:

1. The focus was restricted to German cities.
2. Germany was divided into four regions: North, East, South, and West. Each of these four regions was allocated exactly seven cities, resulting in a set of 28 different cities.
3. Within a region, cities were considered in descending order of population size.
4. No two cities within a region should be within a radius of less than 50 km from each other.

This systematic city selection ensures clarity and balanced city distribution for the output when the model is used by the user. Additionally, the division into different regions is important to enable regional area analysis for the user. This will be discussed in more detail in the upcoming sections 3.2.5, where the scenario definition is explained.

Below is a listing that indicates which federal states belong to each region, along with further details on the cities assigned to each region.

1. **North Region:**
   - Federal States: Schleswig-Holstein, Lower Saxony, Bremen, Hamburg, Mecklenburg-Vorpommern
   - Cities: Hamburg, Bremen, Hanover, Kiel, Rostock, Lübeck, Oldenburg
2. **East Region:**

- Federal States: Berlin, Brandenburg, Saxony, Saxony-Anhalt, Thuringia
- Cities: Berlin, Leipzig, Dresden, Erfurt, Potsdam, Magdeburg, Jena

3. **South Region:**

- Federal States: Bavaria, Baden-Württemberg
- Cities: Munich, Stuttgart, Nuremberg, Karlsruhe, Augsburg, Freiburg, Heidelberg

4. **West Region:**

- Federal States: North Rhine-Westphalia, Hesse, Rhineland-Palatinate, Saarland
- Cities: Cologne, Frankfurt, Düsseldorf, Dortmund, Essen, Mainz, Saarbrücken

This leads to a new set of cities, as shown in the following code snippet:

```
set v   /Aachen, Berlin, Bremen, Chemnitz, Dortmund, Dresden,
        Duesseldorf, Erfurt, Frankfurt, Freiburg, Goettingen,
        Hamburg, Hannover, Karlsruhe, Kiel, Koeln, Leipzig,
        Luebeck, Magdeburg, Mannheim, Muenchen, Muenster, Nuernberg,
        Osnabrueck, Potsdam, Regensburg, Saarbruecken, Stuttgart/;
```

To account for all possible variations, every potential edge was created. This means that a connection was established between each city, as demonstrated in the following code snippet.

```
set E(v, i)
E(v, i) = yes;
```

In order to accurately represent the cities geographically on a map, the newly defined set of cities was linked with their respective geocoordinates. This connection of geodata to the cities, along with other parameters, was facilitated through an externally linked `.gdx` file. The detailed functionality of this linkage is described at the end of this section.

Parameter Determination Based on Real-World Assumptions

To connect the cities with the aforementioned geodata, the two new parameters have been added. These parameters contain the longitudinal $lon(v)$ and latitudinal $lat(v)$ coordinates of each city. This data is publicly accessible and has been extracted for each city. Furthermore, the model parameters $d(v, i)$, $t(v, i)$, and $c(v, i)$, introduced in the optimization model in Section 2.1.1, have been redefined.

The expected **number of passengers traveling between cities** $d(v, i)$ was calculated for the 28 selected cities using a gravity model to obtain realistic values. This model, similar to Newton's law of gravitation, posits that the travel volume between two

locations is directly proportional to the activity levels at the origin and destination. These activity levels are primarily dependent on the population size of each city. A key aspect of the model is ensuring a balance that guarantees that the total number of trips originating from and terminating in a region is equivalent. Overall, the integrated model provides a realistic distribution of travel based on the activity level of each city.

The gravity model, along with all other results, is stored in the final project repository in GitHub under the folder `3 Additional - Models and Code Blocks`. The link to the repository is attached at the end of the section

The **travel time** $t(i, j)$ was calculated as follows:

$$\text{travel time [h]} = \frac{\text{distance between two cities [km]}}{\text{train speed [km/h]}}$$

Based on information provided by Deutsche Bahn, we estimated an average travel speed for the ICE train to be 250 km/h and for the RE train to be 125km/h.[1] [2] [3]

The **construction cost** $c(i, j)$ was calculated as follows:

$$\text{ICE: Construction cost [€]} = 500\,\text{Mio. €} + \left(30\,\text{Mio. €} \cdot \frac{1}{\text{km}}\right) \cdot \text{distance [km]}$$

$$\text{RE: Construction cost [€]} = 250\,\text{Mio. €} + \left(15\,\text{Mio. €} \cdot \frac{1}{\text{km}}\right) \cdot \text{distance [km]}$$

The formula is grounded on a base and a per-kilometer amount. These estimations were derived from the actual construction costs of newly built reference routes by Deutsche Bahn.[4] The base amount proved particularly useful because this way the model tends to favor longer routes over shorter ones in scenarios with limited budgets.

External Linkage via GDX File

The integration of the newly determined parameters was not done manually in the GAMS code as in the initial model. Instead, they were managed in an Excel file, which was then converted to a GDX file (GAMS Data Exchange) that stores data in a format readable by GAMS. This allows the GAMS code to access this file and use the data for compilation. The use of external data management has two main advantages:

[1] **Höchstgeschwindigkeit der Fernverkehrszüge im Bestand der Deutsche Bahn AG (Statista)**, abgerufen am 22. August 2024.

[2] **Züge im Fernverkehr und Streckenkarten (Deutsche Bahn AG)**, abgerufen am 22. August 2024.

[3] **Produktkatalog DB Regio (Deutsche Bahn Regio AG)**, abgerufen am 22. August 2024.

[4] **So teuer ist der Ausbau des Bahnnetzes (Frankfurter Allgemeine Zeitung GmbH)**, abgerufen am 22. August 2024.

1. It enables the integration of larger data sets.
2. It makes data handling significantly more comfortable, as creating and managing a large data table within the GAMS code can be cumbersome and unclear due to GAMS syntax.

To make the data in the Excel files readable for GAMS, these files were converted into a `.gdx` file, which stores data in a format that can be read. This conversion was achieved through a short conversion program in GAMS itself. It is important to note that this specific conversion function runs only in a Microsoft environment.

```
$call gdxxrw.exe "C:\Users\David\Desktop\data_general.xlsx"
    par=inhabitants rng=Geo_Data!a2:b29 rdim=1
    par=long rng=Geo_Data!e2:f29 rdim=1
    par=lat rng=Geo_Data!i2:j29 rdim=1
    par=c_total rng=Construction_Cost!a1:ac29
    par=t_total rng=Travel_Time!a1:ac29
```

Tables 3.1 - 3.4 show excerpts of the two Excel tables and their structure. Table 3.1 includes the values for parameter $d(v, i)$, while Tables 3.2 - 3.4 contain the parameters $long(v)$, $lat(v)$, $c(v, i)$, and $t(v, i)$ for all cities.

All data tables can also be found in the final repository, located in the `3 Additional - Models and Code Blocks` folder under `Excel to GDX | Conversion Code`.

Table 3.1: Excel snippet of expected number of passengers $d(v, i)$ between cities

	Aachen	Berlin	Bremen	Chemnitz	Dortmund
Aachen	248960	9386	9489	9118	10105
Berlin	115655	3644830	121334	124140	119024
Chemnitz	9028	9975	9283	246334	9269
Bremen	20517	21290	567559	20271	21447
Dortmund	22518	21525	22105	20861	586181

Table 3.2: Excel snippet of geocoordinates $long(v)$ and $lat(v)$ of the cities

City	Population	Latitude	Longitude
Aachen	248960	50.7753	6.0839
Berlin	3644826	52.52	13.405
Bremen	567559	53.0793	8.8017
Chemnitz	246334	50.8278	12.9214
Dortmund	586181	51.5136	7.4653

Table 3.3: Excel snippet of construction costs $c(v, i)$ between cities in [bn €]

	Aachen	Berlin	Bremen	Chemnitz	Dortmund
Aachen	0	16.77021	10.01711	14.95654	4.305444
Berlin	16.77021	0	9.997972	6.236714	13.18192
Bremen	10.01711	9.997972	0	11.84035	6.398792
Chemnitz	14.95654	6.236714	11.84035	0	12.17158
Dortmund	4.305444	13.18192	6.398792	12.17158	0

Table 3.4: Excel snippet of travel time $t(v, i)$ between cities in [h]

	Aachen	Berlin	Bremen	Chemnitz	Dortmund
Aachen	0	2.169362	1.268947	1.927539	0.507393
Berlin	2.169362	0	1.266396	0.764895	1.690923
Bremen	1.268947	1.266396	0	1.512046	0.786506
Chemnitz	1.927539	0.764895	1.512046	0	1.55621
Dortmund	0.507393	1.690923	0.786506	1.55621	0

The two `.gdx` files were then placed in the same directory as the GAMS code and called in the code to link them with the model, as shown in the following code snippet.

```
$gdxIn data_general.gdx
$load long, lat, t_total, c_total
$gdxIn

$gdxIn data_gravity_model_V2.gdx
$load d_total
$gdxIn
```

Network Design Problem Repository

3.2.2 Input Features

The following section offers an in-depth discussion of the implementation of the input features. This discussion includes the creation of the Global Information Page, facilitated by the README file, along with three different input options: Budget Restriction, Regional Selection, and Selection of Train Types. The conceptual foundations and the practical implementation of these four areas are thoroughly examined in the following sections. The implementation of input widgets adheres to a standardized procedure, irrespective of the complexity of their functionality:

1. **Define the Purpose**: Select the appropriate widget based on the intended functionality of each input widget.
2. **Prepare the Code**: Adapt the MIRO code to facilitate the creation of relevant input options.
3. **Configure Input Widgets in Configuration Mode**: Set up the final input widget within the Configuration Mode.

The Configuration mode serves as a user interface for creating and customizing all graphical MIRO components. Within this mode, both input and output components can be developed. Upon completion of all required steps, the information related to these input and output components is saved in a `.json` file, which the program utilizes during its execution.

After initiating GAMS, access to the GAMS MIRO Configuration Mode is available by selecting ”MIRO” from the top bar. Following this selection, a pop-up window will appear. By choosing ”Run Configuration Mode” the Configuration Mode is activated. The subsequent sections provide detailed explanations of the four areas previously outlined.

README file

The README file is not a input widget, it is more a file which you can upload in GAMS MIRO in order to create an Global Information Page. The Global Information Page is the first page the user encounters upon opening the GAMS Interface. Since this page represents the first impression for the user, it is crucial that all introductory

information is presented clearly and in a structured manner. Additionally, during development, emphasis was placed on using a high-quality design to maximize the user experience.

To enhance the design, the README file was created in PowerPoint. The PowerPoint slide can be integrated into the GAMS interface by editing the README file in the GAMS MIRO Configuration Mode under the "General Settings" menu, followed by "User Interface". The necessary integration steps are outlined in the following section and illustrated in Figure 3.1.

1. Convert the `.pptx` file to a `.png` file.
2. Name the `.png` file as `Intropage.png`.
3. Save the `.png` file in the folder named `static_network_design_germany`, which is located in the same directory as the `network_design_germany.gms` file.
4. Click on "Edit README".
5. Integrate the picture using the following Markdown syntax:

```
<img src="static_network_design_germany/Intropage.png"
style="width:100%;"/>
```

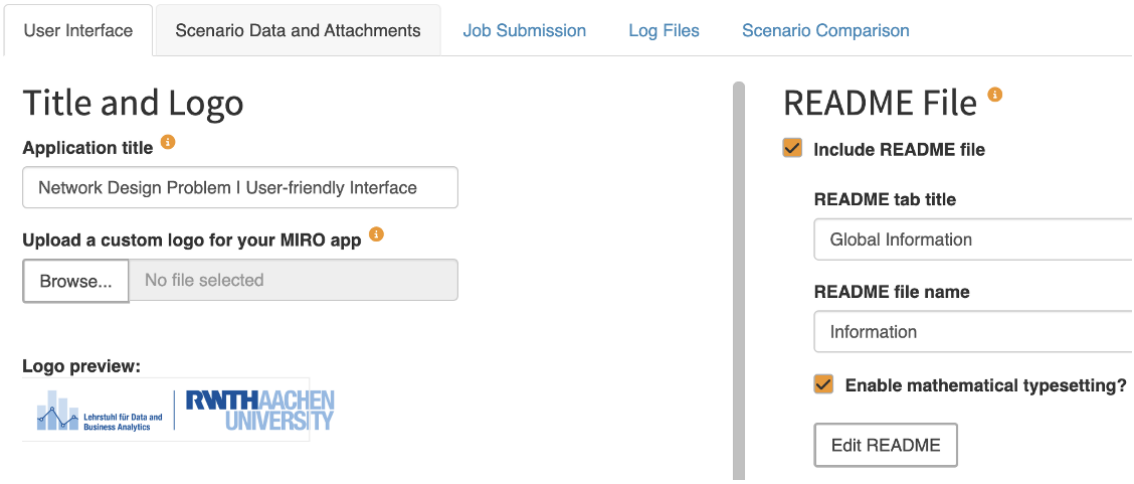


Figure 3.1: Configuration Mode: Implementation README file

Budget Restriction

To ensure maximum ease for users setting the available budget for the network construction, the Budget Widget has been used for the input. Figure 3.2 showcases the appearance of the Budget Widget.

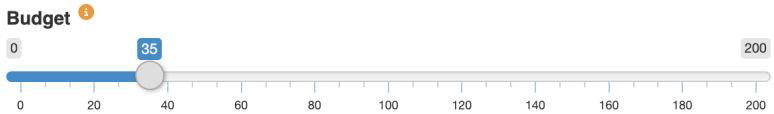


Figure 3.2: Input Widget: Slider for Buget Input

To implement such an input option, specific modifications within the GAMS code are necessary. First a parameter named b had to be established, which represents the user-defined budget. This parameter represents the budget limit and is used in the mathematical optimization model as parameter B. Since each input parameter must be assigned a default value, a value of 35 was assigned. To facilitate the system's recognition of this as an input value, it is crucial to encapsulate the parameter using the commands `$onExternalInput` and `$offExternalInput`.

```
$onExternalInput
    Parameter b / 35 /;
$offExternalInput
```

It is important to mention that always a lower bound for the budget is calculated based on the other input parameters using the Minimal Spanning Tree. This value is used as the budget B in the mathematical model if the user's input falls below the cost of the Minimal Spanning Tree. This approach ensures that a feasible solution always exists. Further details are provided in section 3.2.3, where the Network KPI's output widget is explained.

Based on this code adaptation, it is now possible to configure the widget in GAMS MIRO. After opening the configuration mode, the steps shown in the following paragraph linked to Figure 3.3 need to be executed.

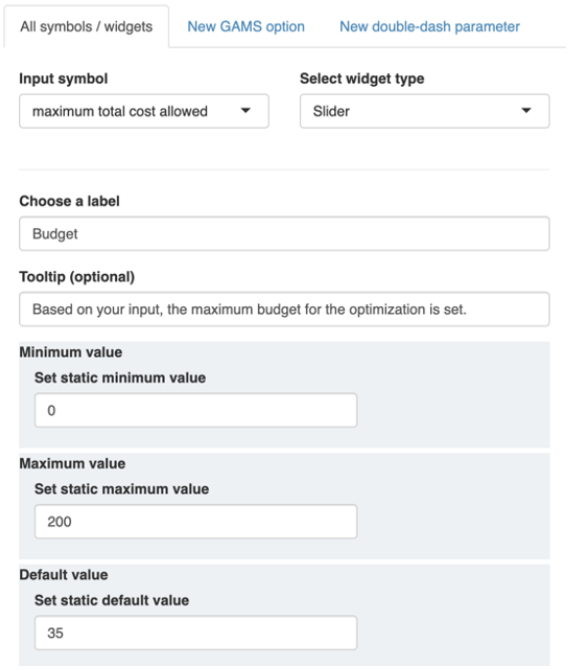


Figure 3.3: Configuration Mode: Implementation Budget Widget

1. Choose b (maximum total cost allowed) as the symbol you want to display.
2. Select the widget type ”Slider”.
3. Label the widget as ”Budget”.
4. Type in the tooltip text which the user can see by hovering over the orange information button. (see Figure 3.2)
5. Set the allowed range for the input between 0 and 200.
6. Set a default value of 35.
7. Save the widget.

Regional Selection

The integration of regional selections enables the user to explore optimized network designs in different regions of Germany. The selection is facilitated by a dropdown menu, allowing the user to make a single selection (see Figure 3.4).

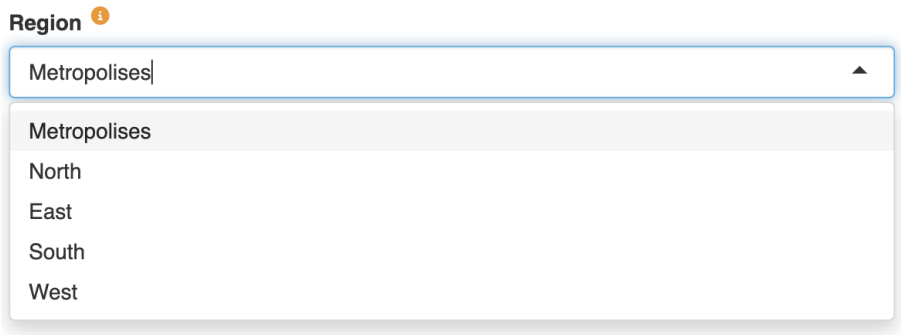


Figure 3.4: Input Widget: Dropdown Menu for Regional Selection

To make such a visualization work, first a new set $v\_subset(v)$ was implemented. This set is used to represent subsets of the entire set v. Initially, it was declared as an empty set. The exact functionality of the set will be explained later in the text.

```
1 Set v_subset(v);
2 v_subset(v) = no;
```

Afterwards, a new set *citySelction* is created to include the various regional selection options.

```
1 Set citySelection /West, North, South, East, Metropolises/;
```

To enable a single-select option for the user, it is necessary to create a singleton set called *subCitySelection*. This set always contains only one entry, which must be from the set *cityselection* and the default value is set to ”West”. By using the commands `$onExternalInput` and `$offExternalInput`, this singleton is provided as an input option for the user.

```
$onExternalInput
    Singleton Set subCitySelction(cityselection) /West/;
$offExternalInput
```

The logical assignment of cities to their respective regions following user selection is detailed next. This mechanism is elucidated via an examination of a specific code snippet. Once a selection is made, the program executes an `if-loop` to ascertain the chosen region, utilizing the singleton set *subCitySelection*. Assuming the user selects "West" the `if-loop` progresses to the "West" block. Here, the previously defined set $v\_subset(v)$ is utilized. This set is populated with cities corresponding to the selected region. The model operates based on $v\_subset(v)$, incorporating the user's regional selection to compute the output.

Furthermore, as illustrated in Figure 3.4, an additional selection option titled "Metropolises" is available. This option enables consideration of the six largest cities in Germany, offering a comprehensive nationwide overview. It is pertinent to note that these cities are pre-defined within the regional selections, ensuring that no new cities are introduced.

```
If (Subcityselction('East'),
v_subset('Berlin') = yes;
v_subset('Potsdam') = yes;
v_subset('Magdeburg') = yes;
v_subset('Leipzig') = yes;
v_subset('Erfurt') = yes;
v_subset('Chemnitz') = yes;
v_subset('Dresden') = yes;);

If (Subcityselction('West'),
v_subset('Saarbruecken') = yes;
v_subset('Frankfurt') = yes;
v_subset('Aachen') = yes;
v_subset('Koeln') = yes;
v_subset('Muenster') = yes;
v_subset('Dortmund') = yes;
v_subset('Duesseldorf') = yes;);

If (Subcityselction('South'),
v_subset('Muenchen') = yes;
v_subset('Mannheim') = yes;
v_subset('Karlsruhe') = yes;
v_subset('Stuttgart') = yes;
v_subset('Freiburg') = yes;
v_subset('Regensburg') = yes;
v_subset('Nuernberg') = yes;);

If (Subcityselction('North'),
v_subset('Kiel') = yes;
v_subset('Hamburg') = yes;
v_subset('Luebeck') = yes;
v_subset('Bremen') = yes;
v_subset('Osnabrueck') = yes;
v_subset('Hannover') = yes;
```

```
35 v_subset('Goettingen') = yes;);

37 If (Subcityselction('Metropolises'),
38 v_subset('Hamburg') = yes;
39 v_subset('Berlin') = yes;
40 v_subset('Muenchen') = yes;
41 v_subset('Koeln') = yes;
42 v_subset('Frankfurt') = yes;
43 v_subset('Stuttgart') = yes;);
```

However, before the user can see the selection field shown in Figure 3.4, in addition to the code modifications, configuration settings need to be adjusted in the Configuration Mode. The following steps explain these adjustments with reference to Figure 3.5:

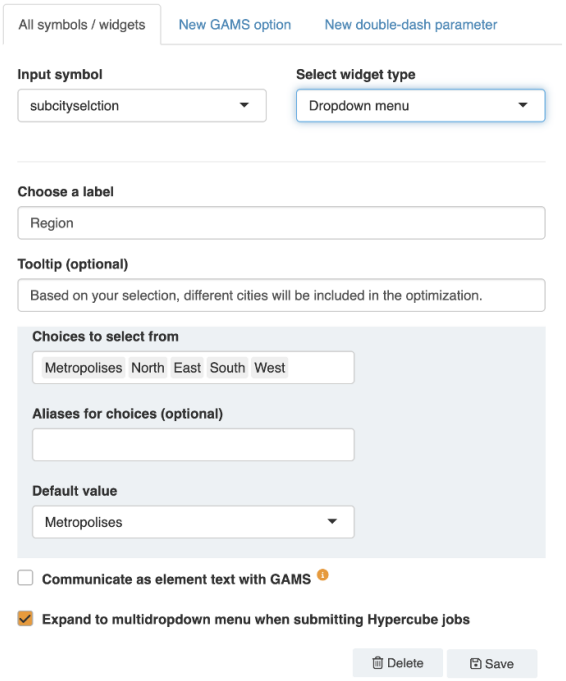


Figure 3.5: Configuration Mode: Implementation Regional Selection Widget

1. Choose *subCitySelection* as the symbol you want to display.
2. Select the widget type "Dropdown menu".
3. Label the widget as "Region".
4. Type in the tooltip text which the user can see by hovering over the orange information button (see Figure 3.4).
5. Set preferred choices to select from: "Metropolises, North, East, South, West".
6. Set a default value to "Metropolises".
7. Save the widget.

Selction of Train Types

In Figure 3.6, the input interface for the user regarding the selection of whether an ICE or an RE network should be built is shown.

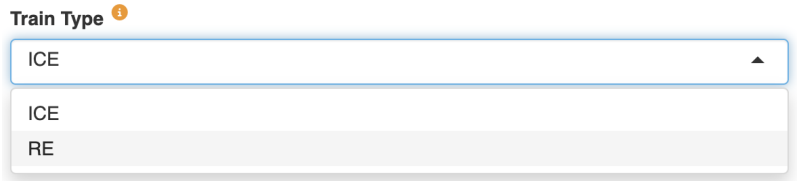


Figure 3.6: Input Widget: Dropdown Menu for Train Type Selection

Similar to the regional selection, the user can choose between the two train types using a single-select option. To implement the functionality, a new set named *trainType* was first created, which contains the two train types.

```
Set traintype /ICE, RE/;
```

Next, a singleton set called *SubTrainType* was created to restrict the user selection to a single choice.

```
$onExternalInput
    Singleton Set subTrainType /ICE/;
$offExternalInput
```

As described in Section 3.2.1, the default parameter selection was based on ICE. Therefore, adjustments to the parameter values are only needed if the user selects RE. This adjustment is implemented using an `if-loop`. In the case of an RE selection, the construction costs $c(v,i)$ and the travel time $t(v,i)$ for all connections between the selected cities are adjusted according to Section 3.2.1, where the initial assumptions were explained.

```
If (subtraintype('RE'),
    c(v,i)$(v_subset(v) and v_subset(i)) = c(v,i)* 0.5;
    t(v,i)$(v_subset(v) and v_subset(i)) = t(v,i) * 2;
);
```

Lastly, adjustments to the settings in the GAMS MIRO Configuration Mode were necessary. These required steps are outlined below, with reference to Figure 3.7:

1. Choose *subTrainType* as the symbol you want to display.
2. Select the widget type ”Dropdown menu”.
3. Label the widget as ”Train Type”.
4. Type in the tooltip text which the user can see by hovering over the orange information button (see Figure 3.4).
5. Set preferred choices to select from: ”ICE, RE”.
6. Set a default value to ”ICE”.
7. Save the widget.

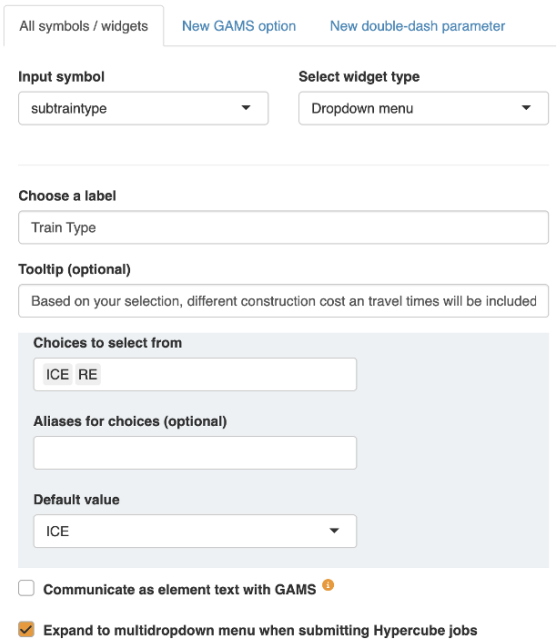


Figure 3.7: Configuration Mode: Implementation Train Type Widget

The implementation process for creating input options for the user has now been fully described. Building on this, the following section will describe the implementation of output visualizations.

It is important to note that while the widgets selected in the program were fully sufficient for the application, GAMS MIRO offers a much wider range of options for providing input widgets. All available options can be found in the linked GAMS MIRO documentation.

GAMS MIRO Documentation - Widgets

3.2.3 Output Features

The following section provides a detailed explanation of the implementation of the output features. The user has three main options for analyzing the outputs: Management Summary, Output Data Sheets, and Input Data Sheets. The conceptual background and the implementation of these three areas are discussed in detail in the subsequent sections.

The implementation of output graphics follows the same steps regardless of the complexity of the representation:

1. **Define message of each visualization**: Based on the message of each visualization, the appropriate graphic type can be selected.
2. **Aggregate data into new parameters**: MIRO requires parameters that are passed from GAMS to MIRO. These must first be declared and aggregated in the code.
3. **Configure graphics in Configuration Mode**: The final graphic is set up in the Configuration Mode.

After all steps have been successfully completed, the information about the graphic is stored in a `.json` file, which the program accesses during operation.

Management Summary

The purpose of the management summary is to present the key simulation results in a clear and accessible manner. Special attention was given to ensuring that the presentations are as visual as possible, in order to facilitate efficient comprehension and interpretation of the results. Irrelevant information was deliberately omitted to focus on the essential aspects. Additionally, the management summary serves as a basis for making the scenarios explained in Section 3.2.4 comparable.

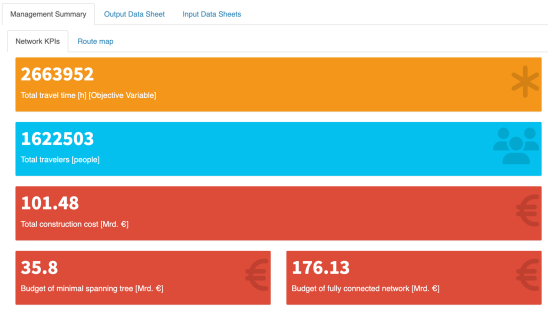


Figure 3.8: Output Widget: Scalars for Network KPIs View

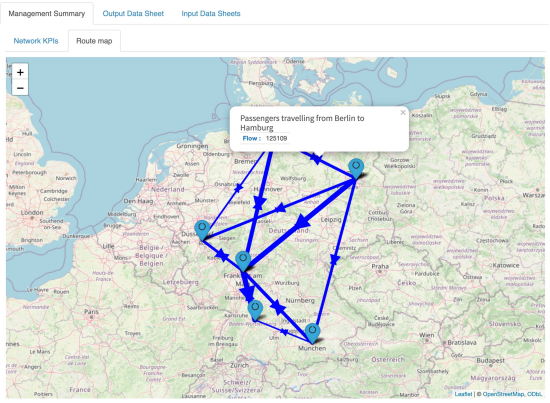


Figure 3.9: Output Widget: Map for Route Map View

As shown in Figure 3.8 and Figure 3.9, the management summary is divided into two sections: a **Network KPI view** and a **Route Map view**. The Network KPI view is designed to present the key simulation results in numerical form. It highlights three central values:

- **Total Travel Time [h] (Objective Value) (orange):** This value represents the cumulative travel time and, as outlined in Section 2.1, serves as a key indicator of network quality. Consequently, it is valuable for comparing different networks. It takes into account both the network's connectivity and the nature of the connections, including whether they involve an ICE or a RE.
- **Total Travelers [people] (blue):** This value represents the total number of passengers traveling. It is primarily dependent on the selected region. Within a given region, this value remains constant, regardless of the types of trains or budgets being compared.
- **Budget [Billion €] (red):** This value indicates the actual budget allocated for the construction of the rail connections. The budget b set by the user at the beginning serves as an upper limit that cannot be exceeded.

To enable the user's interpretation of the selected budget, the corresponding bounds for the current scenario, determined by the region and type of rail connections, are also provided. The Upper Bound, displayed on the right, represents the budget required to establish a fully connected network. In this scenario, every city is linked to every other city, and beyond this budget, no additional connections are possible. Consequently, choosing a budget exceeding the Upper Bound would not be practical.

The lower bound shown on the left-hand side indicates the minimum budget required to generate a Minimum Spanning Tree, as explained in Section 2.2. If the model would attempt to construct a network with a budget below this lower bound, no feasible solution would be possible, as not all passengers would have a connection to their destination. As previously mentioned in Section 3.2.2 regarding the budget input widget, the model adjusts the entered budget to ensure that a feasible solution exists. Later in this chapter it will be described how the model automatically adjusts any input budget that falls below the lower bound by first calculating the MSP and then matching the budget to it.

In Figure 3.9, the second output visualization, the Route Map view, is presented. This visualization consists of an interactive map of Germany. The map highlights two key elements:

- **Nodes (Cities):** The blue pins on the map represent the cities considered by the model based on the user's region selection. When hovering over a city, the user can view information about the city, including its population.
- **Edges (Constructed Connection Elements):** The blue edges represent the connections that have been successfully constructed. In the model, any established connection is automatically bidirectional. When the user clicks on a connection element, they can view the number of passengers traveling between city v and City i.Additionally, the thickness of the arrow symbolizes the traffic volume - a thicker arrow indicates a route with higher traffic density.

To display the described output, some adjustments had to be made to the codebase, similar to those made for the input features. In order to output the data points

shown in the **Network KPI view**, the new parameters *totalTravelTime* (cumulative travel time), *totalTravelers* (total number of travelers), *totalCost* (budget used), b_{max} (budget for a fully connected network), and $b_{minDisplay}$ (budget for the minimal spanning tree) were first declared. Before the parameters could be passed to MIRO, the following calculations had to be performed:

- *totalTravelTime* : The total travel time corresponds to the objective value G_l, which first had to be converted from a variable into a parameter. This is accomplished by simply copying the value.
- *totalTravelers* : The total number of travelers is determined by summing the number of travelers $d(v, i)$ between cities v and i across all possible pairs of cities v and i.
- *totalCost* : To represent the total budget utilized, the construction costs of the actually built connections are aggregated. This is done by first multiplying the construction costs between two cities $c(v, i)$ by the binary variable $y.l(v, i)$. Subsequently, the sum is taken over all cities v and i. In this way, only the connections that have actually been constructed are considered.
- b_{max} : To calculate the upper bound of the fully connected network, the costs associated with the connections between each city v and city i are summed over all possible pairs of cities v and i. The resulting sum is then divided by 2, as the construction costs for each bidirectional connection are incurred only once.
- $b_{minDisplay}$: The parameter represents the lower bound of the budget and corresponds to the MST. As previously mentioned in Section 2.2, this is determined through a supplementary optimization problem.
- *approxSpanningTree* : A preliminary mathematical optimization model is utilized to calculate the budget for the MSP, as explained in Section 2.2, based on the Kruskal algorithm.

The necessary calculations for the paramters as well as the the optimization model *approxSpanningTree* for calculating the MSP are listed in the following code block, presented in the correct GAMS syntax:

```
parameters
    totalTravelTime
    totalTravelers
    totalCost
    b_max
    b_min_display;

totalTravelTime = G.l;
totalTravelers = sum ((v,i), d(v,i));
totalCost = sum((v,i)$(E(i,v)), c(v,i)*y.l(v,i));
maxTree = (sum((v,i)$(v_subset(v) and v_subset(i)),c(v,i)))*0,5;

equations
    objectivefunction2
    flow
```

```
16     coupling;

18 objectivefunction2..
19     G =e= sum((i,j)$(E(i,j)), c(i,j) * y(i,j));
20 flow(u,v)$(ord(u) <> ord(v))..
21     sum((i)$(E(i,v)),x(u,i,v)) - sum((j)$(E(v,j)),x(u,v,j)) =e= d(u,v);
22 coupling(u,i,j)$(E(i,j) and ord(i) < ord(j))..
23     X(u,i,j) + x(u,j,i) =l= d(u,u)*y(i,j);

25 model approxspanningtree /objectivefunction2,flow,coupling/;
26 solve approxspanningtree min G using mip;

28 b_min_display = G.l;
```

A further aspect of the code adjustments that requires closer examination is the minimal budget derived from the *approxSpanningTree* model described earlier. Simply calculating this value and using it as a static input for MIRO is insufficient. To ensure that the optimization model consistently operates with feasible input values, it is necessary to verify whether the provided budget falls below the budget of the MST. If this occurs, the model will instead use the MST budget. As demonstrated in the subsequent code block, the minimum value between the provided budget b and the calculated target value G_l from the *approxspanningtree* model is always used for the actual network design optimization and is stored in the parameter b_{min}.

```
1 parameter b_min;
2 b_min = max(b, G.l);
```

After completing all the calculations described, the parameters for the Network KPI's view are now ready to be transferred to GAMS MIRO. Similar to the input process, this transfer is accomplished through a transparent code block that lists all the parameters intended for later visualization. The function in GAMS is structured as follows:

```
1 $onExternalOutput
2     Parameter totalCost;
3     Parameter totalTravelTime;
4     Parameter totalTravelers;
5     Parameter b_max;
6     Parameter b_min_display;
7 $offExternalOutput
```

To visualize the data in the **Route Map View**, it was also necessary to make preparatory adjustments in the codebase. The visualization of the cities and their connections is based on the following parameters, which are derived from static geographical coordinates and population size given by the `.gdx` file. These parameters are aggregated, filtered, and adjusted based on the optimization results:

- *longs(v)* and *lats(v)* : These parameters include both longitudinal and lateral coordinates, serving two primary functions. Firstly, they provide the foundation

for placing the pins on the map, which represent the cities selected by the user. Secondly, they act as the starting points for the edges, that are illustrating the connections between these cities. The parameter is a duplicate of the static longitudinal and lateral coordinates assigned to the cities. However, before assigning the value to city v in the connection (v, i), it is confirmed whether the connection is indeed established, ensuring that $y.l(v, i) = 1$ holds true. This verification guarantees that only the connections determined by the model are displayed.

- *longz(v)* and *latz(v)* : Similar to the starting coordinates, these parameters represent the target coordinates of the connections (v, i). They follow the same logic based on $y.l(v, i)$.
- *travpass(v,i)* : This parameter serves as the basis for displaying the load on each connection. The variable $x_l(u, v, i)$ generated by the model is summed over the city u, resulting in the total number of passengers traveling from city v to city i, regardless of where their journey originally started.
- *inhabitants(v)* : Clearly, this refers to the population of the displayed cities. For city v, this value is stored in $d(v, v)$ and can be retrieved by copying $d(v, v)$.

To visualize these values in MIRO with the appropriate output graph of the map, they first need to be consolidated in a central table. The adjustments in the code and the aggregation in the *map* table are shown in the following code excerpt:

```
parameter travpass (v,i)

Set mapHdr / lats, longs, latz, longz, travpass, inhabitants/;
Table map(v,i, mapHdr);

map(v,i,'lats')$(y.l(v,i)>=0.9 or y.l(i,v)>=0.9) = lat(v);
map(v,i,'longs')$(y.l(v,i)>=0.9 or y.l(i,v)>=0.9) = long(v);
map(v,i,'latz')$(y.l(v,i)>=0.9 or y.l(i,v)>=0.9) = lat(i);
map(v,i,'longz')$(y.l(v,i)>=0.9 or y.l(i,v)>=0.9) = long(i);
map(v,i,'travpass')$(y.l(v,i)>=0.9 or y.l(i,v)>=0.9) = travpass(v,i);
map(v,i,'inhabitants')$(y.l(v,i)>=0.9 or y.l(i,v)>=0.9) = d(v,v);
```

It is important to note that the variable $y.l$ is not compared directly to 1 but rather to a range greater than 0.9. This adjustment was made due to a minor bug where, after optimization, the $y.l$ values were not exactly equal to 1, resulting in no coordinates being assigned to these values and causing missing segments in the visualization. By comparing to a range, this bug can be effectively resolved.

As with the Network KPI view, the map table must still be passed to MIRO using the following function:

```
$onExternalOutput
    Table map(v,i, mapHdr);
$offExternalOutput
```

Since all necessary aggregations have been completed in the code and the required data points have been passed to MIRO, the initially presented output graphics can now be initialized in the configuration mode. The procedure is similar to that of the input widgets, which was explained in Section 3.2.2. Various visualization options are available for presenting the output graphics. For the Management Summary, the Value Boxes, which allow scalar values to be highlighted, and the map view, which is ideal for visualizing the Network Design Problem using pins and flows, are particularly important.

Any potential output visualization can be configured in the configuration mode through the `Graphs` menu in the left sidebar. First, the following paragraph outlines the necessary steps to set up the **Network KPIs view**.

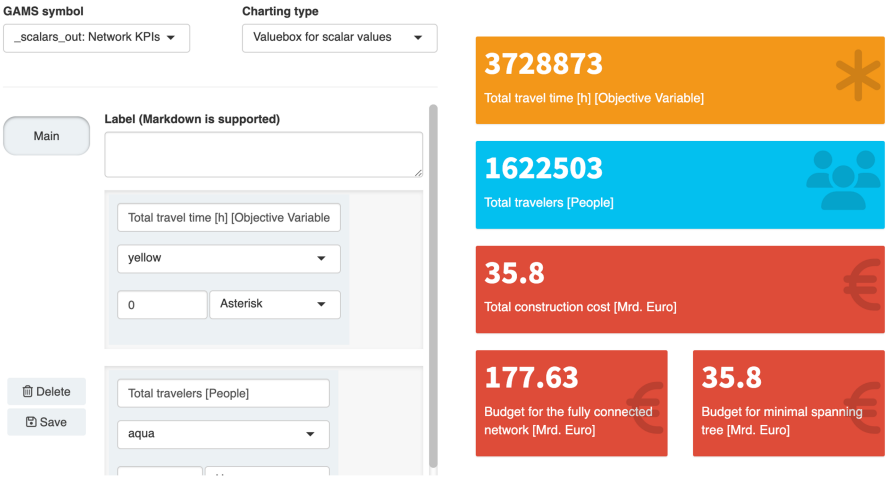
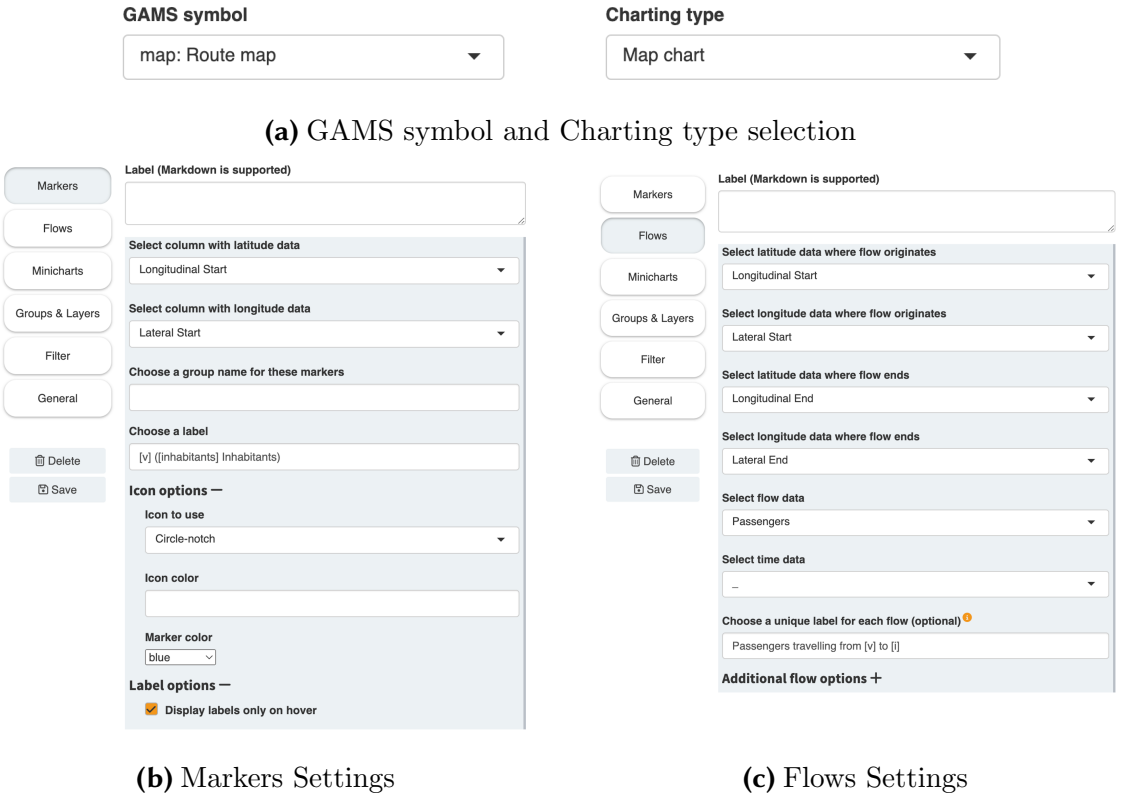


Figure 3.10: Configuration Mode: Implementation Network KPIs View

1. Choose ”scalars out” as the symbol you want to display. This selection contains all scalar values without indices, including the previously defined parameters *totalTravelTime*, *totalTravelers*, *totalCost*, $b_{minDisplay}$, and $b_{\max}$.
2. Select the charting type ”Valuebox for scalar values”.
3. The ”Valueboxes” can be freely arranged. Additionally, four individual customizations can be made for each box:
   - Adjusting the description
   - Modifying the background color
   - Setting the number of decimal places
   - Selecting the field type to be displayed in the background
4. Changes are updated live on the right hand side and can be monitored in real-time.
5. Save the visualization.

The following section will outline the required steps in configuration mode to establish the **Route Map view**. To achieve the desired outcome, it is essential to configure both the markers, displayed in Figure 3.11b, and the flows, displayed in Figure 3.11c.



(a) GAMS symbol and Charting type selection

(b) Markers Settings

(c) Flows Settings

Figure 3.11: Configuration Mode: Implementation Route Map View

1. Choose ”map” as the symbol you want to display. This includes the previously aggregated table with the parameters *longs*(v), *longz*(v), *lats*(v), *latz*(v), *inhabitants(v)*, and *travpass(v,i)*.
2. Select the charting type ”Map chart”.
3. Set up the markers
   (a) Select the variable *lats*(v) as latitude data.
   (b) Select the variable *longs*(v) as longitude data.
   (c) Choose a label for each marker with the name and population of the city v. Parameters can be referenced in the label using square brackets (e.g., ”[v] with [inhabitants] inhabitants” becomes ”Cologne with 1,100,000 inhabitants”).
   (d) Select the preferred icon logo and icon color.
   (e) Enable ”Display labels only on hover” to maintain clarity.
   (f) Save the visualization.
4. Set up the flows
   (a) Select *longs*(v) and *lats*(v) as the starting coordinates.
   (b) Select *longz*(v) and *latz*(v) as the destination coordinates.
   (c) Select the parameter *travpass*(v, i) as the flow data, so that the load for each individual flow is displayed.

(d) Choose a label for each flow that shows both the load and the cities involved in the flow.

(e) Save the visualization.

It is important to note that in the attached figures, the declared variable names are not visible, but rather spelled-out descriptions. This is because in GAMS MIRO, each parameter can be assigned its own label. This label is then used in the subsequent configuration for the variables.

After completing the described steps for the Network KPIs view and the Route Map view, the implementation of the most important output graphic is finished. The next steps involve implementing the static data tables for the input and output data sheets.

Output Data Sheets

In the Management Summary, all information is presented primarily in a graphical format. However, in some cases, it may be useful to have the information available in the form of data tables as well, especially when dealing with larger datasets or when the data needs to be exported.

For this reason, the information simulated by the model regarding the **number of passengers per connection** $travpass(v, i)$ and the data on the **actually constructed connections** $y.l(v, i)$ have been stored in data tables and made available to the users.

The data table is the default output for result presentation if no further graphics have been configured. When configuring the table, there are primarily two options: The table can be output as a pivot table, where the user can individually select the content, or it can be displayed as a static table. For the described use case, the use of static tables is sufficient. The specific settings for all output and input tables can be adjusted in the table settings. These can be accessed under ”Tables” and then ”Output Tables (individual symbols)” in the menu of the configuration mode. The specific settings that were applied are detailed in the following step-by-step guide:

Figure 3.12: Configuration Mode: Implementation Date Sheets

1. Select the desired parameter.
2. Select cities as the ”column to pivot” to achieve a matrix output.
3. Select the table style.
4. Disable ”table filtering”, as it is not necessary in this context.
5. Set the number of entries per table.
6. Set the number of decimal places for the entries.
7. Select the export options. In the example, the CSV, Excel, and PDF options were selected.
8. Repeat this process for each table.

Input Data Sheets

In some cases, it may be useful to review the input data considered by the model in addition to the output data calculated by the model. For this reason, input data sheets have been added alongside the output data sheets. These contain information on the **number of travelers between cities** $d(v, i)$, **construction costs** for all possible connections $c(v, i)$, and **travel times between cities** $t(v, i)$. The setup process for these tables is analogous to the setup process for the output tables described earlier and will therefore not be further elaborated.

It is important to note that not all visualization options in MIRO were utilized in the described output of the project for the Network Design Problem. Additional output graphics are explained in the GAMS MIRO documentation at the following link. The Custom Render feature, which allows for the creation of custom renderings using R-Shiny, offers particularly exciting possibilities.

GAMS MIRO Documentation - Charts

3.2.4 Aggregation of Input and Output Features

Now that all relevant input and output figures have been implemented, they need to be aggregated into the correct structure to achieve the desired layout. This is exemplified for the Management Summary in Figure 3.13.

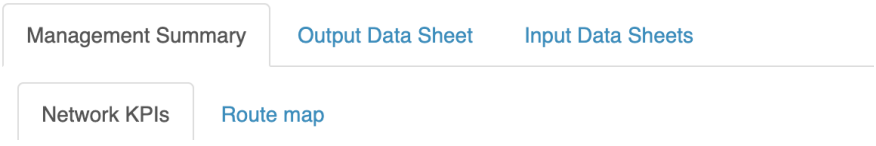
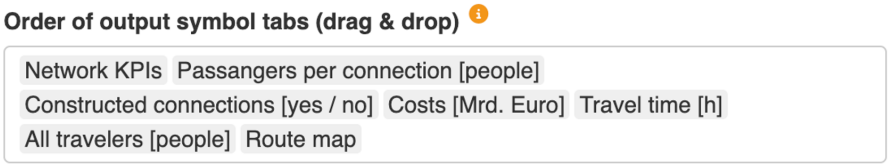


Figure 3.13: Aggregation of Output Widgets

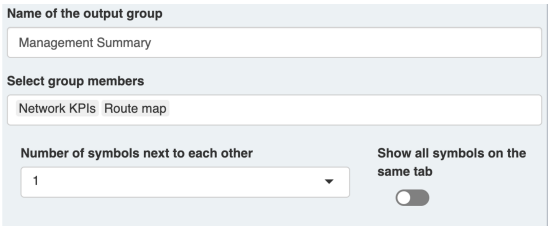
The settings for the output and input structure tabs can be adjusted under the ”Symbol” and ”Symbol Order and Groups” menu in the configuration mode. There, the figures can be arranged via drag-and-drop, as shown in Figure 3.14a, and also grouped, as shown in Figure 3.14b and 3.14c. When grouping figures, it is possible to display the graphics either in a single tab, as done in the User Input, or to arrange the figures across different tabs, as implemented in the Management Summary.

The functionality of the dashboard is now fully established, and the graphics have been implemented at a final level. The only remaining adjustments are the integration of predefined scenarios and the implementation of an app that consolidates everything, allowing the program to be used independently of the development environment. The subsequent sections will cover these remaining steps in detail.



(a) Order of all Output Figures

(b) Grouped User Input



(c) Grouped Management Summary

Figure 3.14: Configuration Mode: Aggregation of Output Widgets

3.2.5 Scenarios

To enable end-users to quickly and easily select input options that yield meaningful results, and to ensure comparability between different scenarios, several predefined scenarios have been implemented for user access.

Figure 3.15 provides an overview of the defined scenarios across their different levels. The scenarios are grouped into the following three degrees of freedom with the mentioned characteristics:

- **Regions**: 1. German Metropolises, 2. North, 3. East, 4. South, 5. West
- **Train Type**: 1. ICE, 2. RE
- **Budget**:
  1. **Low**: Budget set to the cost of the MST.
  2. **Medium**: The average of the respective low and high budget values.
  3. **High**: Budget set to the cost required for a fully connected network where all possible connections are built.

This results in a total of 30 predefined scenarios, which can be implemented and pre-simulated as outlined in the following section. Consequently, users do not need to run additional simulations when selecting a scenario. This approach enables them to quickly and flexibly compare a wide variety of simulation data. Particularly interesting are network comparisons where two parameters remain constant, and one

parameter is varied. For example, the comparison of network performance across different train types is especially noteworthy, as it highlights two opposing effects: increased travel times $t(v, i)$ versus lower investment costs $c(v, i)$. Additionally, comparisons with different budget levels are compelling, as they reveal the most significant differences in the network itself.

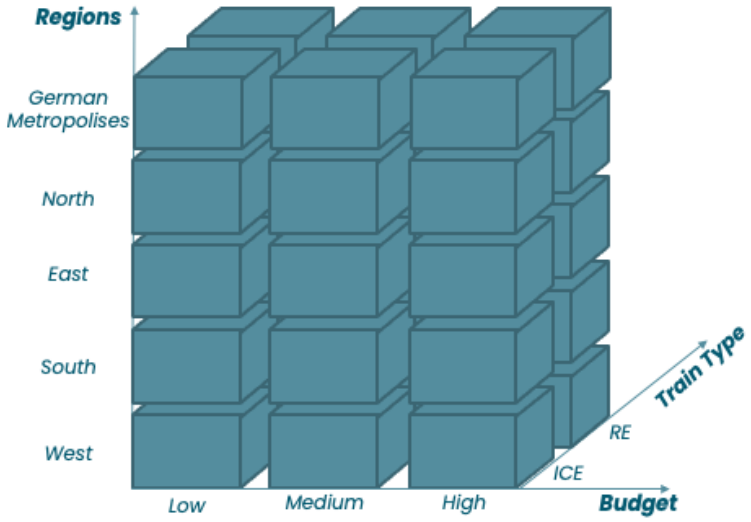


Figure 3.15: Illustration of 30 Predefined Scenarios

Scenarios are created in the GAMS MIRO Base Mode, which can be accessed similarly to the Configuration Mode. This is achieved by selecting "MIRO" from a bar displayed at the top when GAMS is open, followed by clicking on "Run Base Mode." The Base Mode provides the opportunity to simulate the program in a live environment, allowing for the visualization of the current settings from the Configuration Mode as they would appear to the user later on in the final Application.

An exemplary process is provided below to illustrate how to create predefined scenarios and make them accessible to the user. In this example, the scenario "Metropolises | ICE | High" is created. The steps are as follows:

1. Adjust the input settings for regional selection and train type selection according to the example. Leave the budget setting unchanged for now (see Figure 3.16).

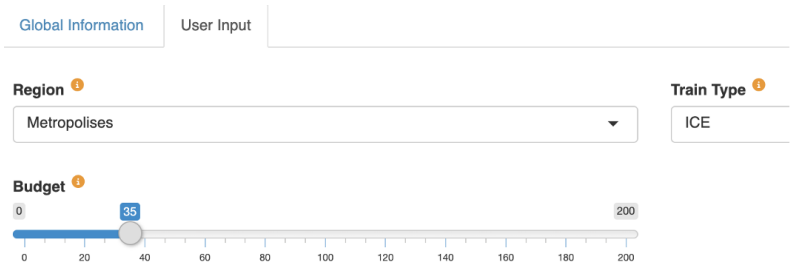
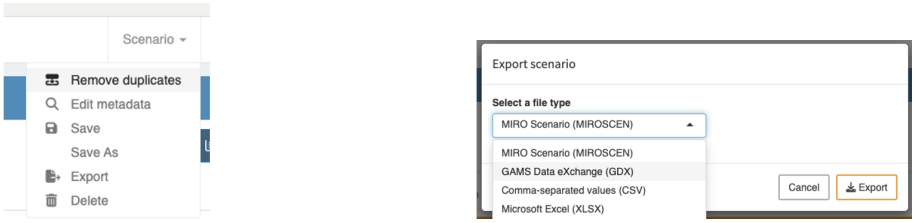


Figure 3.16: Base Mode: Configruation of Szenario

2. Click on ”Solve” on the menu on the left sidebar.
3. Click on ”Scenarios” in the upper right corner of the Base Mode (see Figure 3.17a).
4. Click on ”Export” in the menu that appears (see Figure 3.17a).
5. Select ”GDX” as the file type (see Figure 3.17b).
6. item Click ”Export” (see Figure 3.17b).
7. Name the file appropriately for user identification. In this example, ”Metro | ICE | High” has been chosen.
8. Save the file in the folder `data_network_design_germany`.



(a) Snippet of the GAMS Base Mode **(b)** Snippet of GDX Export

Figure 3.17: Base Mode: Saving Scenarios as GDX

The described procedure should be repeated for all defined scenarios to ensure that each scenario is calculated by the optimization model and stored for later use. After this process the creation of the scenario is complete. The ability to work with scenarios in the final app, select them, and compare them with each other is addressed in Section 4.3 of the following chapter 4 ”User Manual of Gams MIRO App”.

3.2.6 MIRO App

To make the dashboard accessible to users outside the developer mode, it is exported as a standalone app. The advantages of this approach are clear: the user can fully utilize the model and perform simulations without the risk of making unintended changes to the code that could damage the model. After export, the program is available as a `.miroapp` file, which contains the model along with all predefined scenarios. This file can be opened directly with GAMS MIRO. Thus, no prior experience with GAMS is required to operate the program.

To export the app, the following steps must be completed:

1. **Create a .txt file**: To inform GAMS about the files needed for the app deployment, the required files must be listed in a central `.txt file`. The following files and folders are necessary for the upcomming app deployment:
   - `network_design_germany.gms`: Main model
   - `data_general.gdx`: Essential data foundation with information on city coordinates $lat(v)$, $long(v)$, construction costs for route connections $c(v, i)$, and travel times $t(v, i)$.

- `data_gravity_model.gdx`: Essential data foundation for the number of traveling passengers $d(v, i)$.
- `Information`: Unix file containing the Global Information page.
- `data_network_design_germany`: Folder containing all predefined scenarios.
- `static_network_design_germany`: Folder containing all used images.
- `conf_network_design_germany`: Folder containing the relevant `.json` files for the input and output figures.

2. **Integrate App Logo**: To display the app later in the users library, an app logo can also be added. This must be saved as `app_logo.jpeg` in the `static_...` folder.
3. **Integrate App Title, Description and ID**: In addition to the logo, a title, a short description and an ID must also be added. This is done using an additional file named `app_info.json`, which must also be placed in the `static_...` folder. The exact content of the file is provided in the following code listing. The final app icon for the Network Design Problem is shown in Figure 3.18.

```
{
    "title": "Network Design",
    "description": "...",
    "appId": "networkdesign"
}
```

4. **Deployment of final App**: To export the app, go to the "MIRO - Assembly and Deploy" menu in GAMS and deploy the program as a Multi-User App. Simply select the previously created `.tex` file and set the execution environment to Multi User. Optionally, a test deployment can be performed before the actual deployment to check the core functionalities in advance.

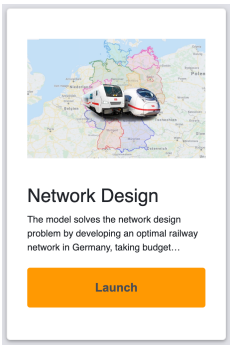


Figure 3.18: MIRO App: Icon Network Design Problem

After the deployment is successfully completed, the program is available as a `.miroapp` file and can be shared with users. Further information on the deployment process and alternative deployment options can be found in the GAMS documentation.

GAMS MIRO Documentation - Deployment

4. User Manual of Gams MIRO App

Chapter 4 centers on the user experience, providing detailed instructions on how to properly install, use, and navigate the application. **Section 4.1** outlines the installation process step-by-step, ensuring that the user can set up the application right. Subsequently, **Section 4.2** offers a guide to the app's functionality, covering all aspects from data input to the comparison of different scenarios.

4.1 Get Started

To install the application, certain prerequisites must be fulfilled. Primarily, the user must have GAMS MIRO installed on their device. A comprehensive installation guide, along with download links for the program, can be accessed through the following link.

`Install GAMS MIRO Desktop`

Once GAMS MIRO is installed, the next step is to download the Network Design Program. The required file, `network_design_germany.miroapp`, is located in the attached repository, under the folder `1 User App - Network Design Problem`.

`Network Design Problem Repository`

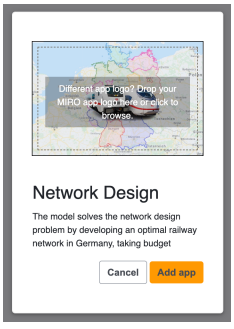


Figure 4.1: MIRO App: Icon during Installation Process

After downloading the program, it must be added to the library in GAMS MIRO. Upon opening the program, the app logo, as illustrated in Figure 4.1, will appear in the libary. By clicking on “Add App,” the user adds the model and can start modeling immediately.

4.2 App Functionality

The following section offers a comprehensive overview of the app’s functionality. The user can navigate threw the menu, shown in Figure 4.2, which includes the following options: **Input**, **Output**, **GAMS Interaction**, **Load Scenarios**, and **Compare Scenarios**. Each of these options is explained in detail in the subsequent sections.

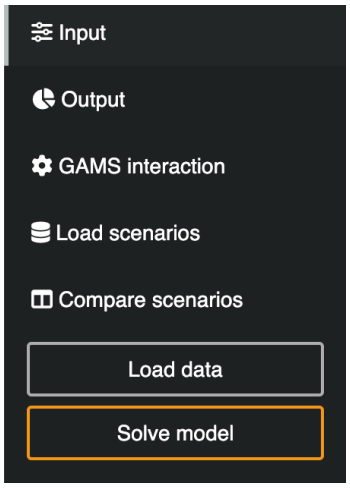


Figure 4.2: MIRO App: Menu Bar

Input

To select the input options, the user must first click on the ”User Input” button (see Figure 4.2). This action will open the Input Page (see Figure 4.3). Here, the user can set the budget by moving the slider at the bottom of the page.The selection options for regions and train types are available by clicking on the fields and choosing from the single-select dropdown menu that appears. Once the selections are complete, the user must press the ”Solve” button to run the model with the chosen inputs.

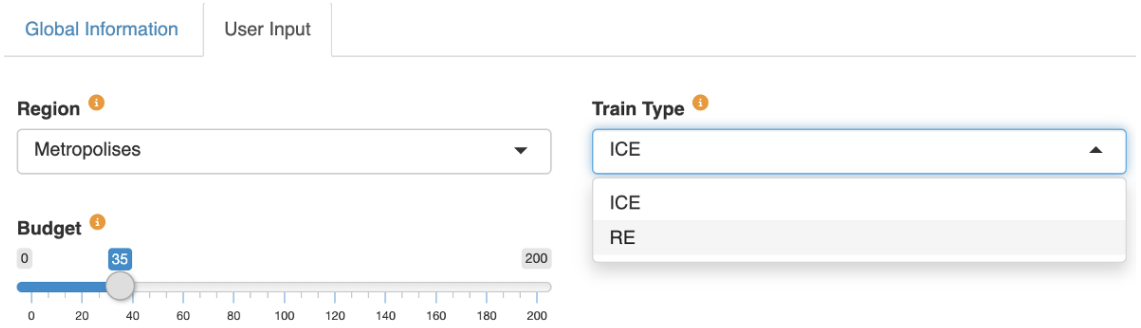


Figure 4.3: MIRO App: Input

Output

After the input has been provided and the results have been calculated by the model, the user can view the output figures of the optimization in the Output section based on the given input values. Figures 4.4 and 4.5 present the information from the Management Summary again. Detailed explanations of the visualized KPIs and the displayed map are provided in Chapter 3.2.3.

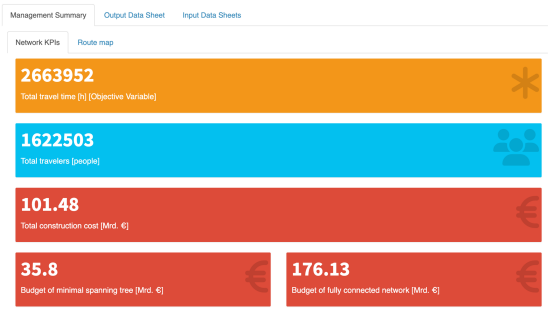


Figure 4.4: MIRO App: Network KPIs Output

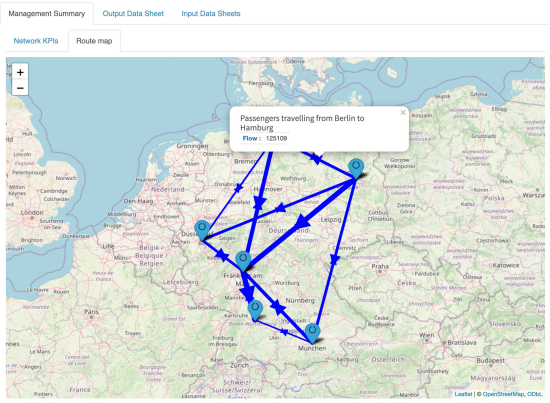


Figure 4.5: MIRO App: Route Map Output

GAMS Interaction

The GAMS Interaction section acts as the primary interface between MIRO and GAMS. After each compilation of a solution with updated input values, the model execution status is displayed here, as illustrated in Figure 4.6. Additionally, the `.log` files are stored in this section. These files contain all key solver steps, as well as the iteration steps of the most recent solution. If necessary, the `.log` files can also be downloaded here.

Figure 4.6: MIRO App: GAMS Interaction

It is important to note that when the user loads a predefined scenario, as described in the following section, no new `.log` files are created. Since both the input data and pre-simulated output data are stored in the scenario, no new compiling is necessary.

Under normal circumstances, this section is not required, as all relevant simulation results are presented as visualized output figures in the output menu. However, in the event of a faulty execution of GAMS MIRO, this section provides users with information about the errors encountered.

Load Scenarios

In the GAMS MIRO application, there are two distinct methods for utilizing scenarios. The first method is accessible by selecting the ”Load Scenarios” tab (refer to Figure 4.2). This action redirects the user to a page resembling Figure 4.7, initially absent of the data table at the lower section of the page. By clicking the ”Fetch results” button, the user retrieves a data table displaying all the predefined scenarios. The Conditions section, positioned at the top of the page, enables the user to apply filters, thereby enhancing the manageability and navigability of the data table. This feature is particularly useful when the user, for example, wants to display only scenarios within the selection of metropolitan regions or when only ICE connections are relevant for an analysis.

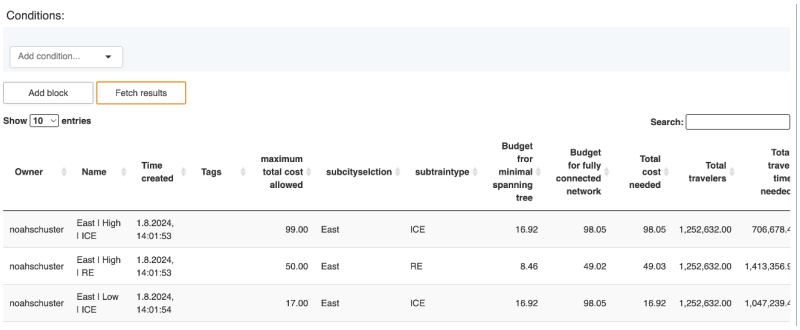
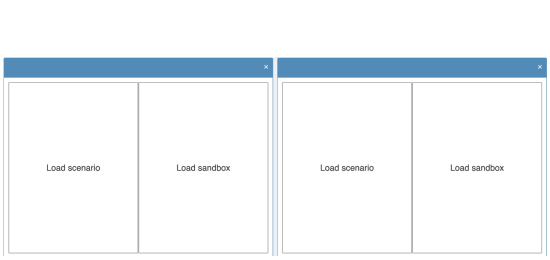


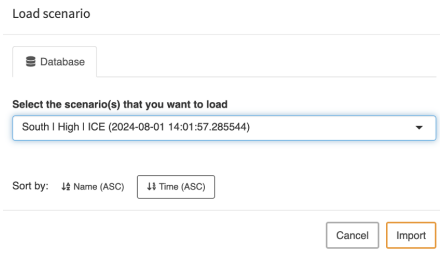
Figure 4.7: MIRO App: Load Scenarios

Compare Scenarios

The second way to use scenarios is by comparing two scenarios with each other. The user begins by clicking on the “Compare scenarios” tab (Figure 4.2). Next, the scenarios to be compared must be selected by clicking the “Load scenario” button for each scenario (Figure 4.8a). This process allows the user to choose the corresponding scenarios for comparison.



(a) Select two senarios



(b) Popup menu

Figure 4.8: MIRO App: Compare Scenarios

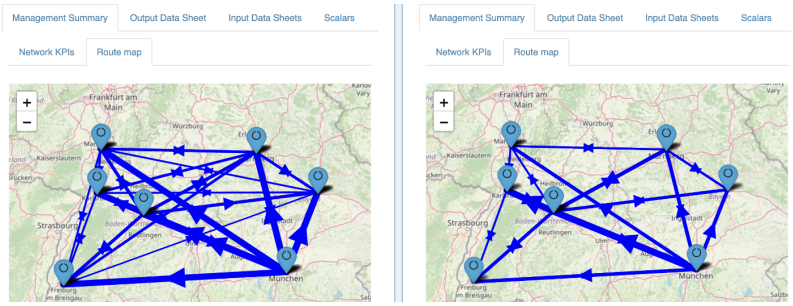


Figure 4.9: MIRO App: Exemplary Szenaro Comparison

Figure 4.9 illustrates an exemplary scenario comparison for the southern German cities, once with a medium level and once with a high level. The example clearly demonstrates the advantages of scenario comparison. In the split view, it is immediately apparent which connections were either constructed or omitted in both scenarios.

All the core functionalities of the GAMS MIRO App have now been demonstrated, enabling the user to fully utilize the app's capabilities. The use of other MIRO applications follows the same pattern, making it easy to apply the knowledge adaptively. The GAMS MIRO team provides a range of example applications for other optimization problems as part of a demo, which are also worth exploring. All example applications can be found at the following link.

GAMS MIRO Documentation - GAMS MIRO gallery

5. Conclusion and Outlook

The implementation of the dashboard has significantly contributed to a deeper understanding of the Network Planning Problem and provided valuable insights into its behavior under specific configurations. The challenges, particularly regarding the minimization of computation time to ensure an optimal user experience, have yielded important findings about the problem's complexity. It became evident that the complexity is primarily influenced by two factors: the available budget and the number of cities involved. A smaller budget leads to a significant expansion of the solution space, thereby increasing the number of network configurations to be evaluated, which substantially heightens the problem's complexity. The number of cities has a similar effect, as each city in the model is connected to every other city. Consequently, with each additional city, the solution space of the model expands exponentielly. To ensure efficient runtime, an upper limit of 7 cities was introduced and the model was limited by this bound. Furthermore, it was learned from the model's behavior that the objective value is positively correlated with travel time, distance, costs, as well as the total number of cities and travelers. This correlation exists because either more connections can be established, or existing connections can be utilized more efficiently.

These insights are also quickly apparent to the user of the described dashboard, as its interactive and visual nature, along with the ability to compare optimization results in the form of scenarios, makes analyzing the model's behavior remarkably easy. Additionally, the results are easy to understand and relate to, as the model is closely aligned with reality while making simplifications in the right areas. This highlights the primary benefit of the application. It is ideally suited for educational purposes, enabling users who have little experience with optimization problems to quickly and intuitively grasp models like the Network Design Problem.

To make the interaction with the dashboard even more engaging and educational, there are several opportunities for further improvement of the application. Firstly, integrating additional modes of transportation within the user selection would be beneficial. Moreover, rather than simulating each mode of transportation separately as is currently done, these could be incorporated into a unified simulation. The

assumptions regarding distances and routes between cities could also be made more realistic by taking into account construction constraints, rather than relying solely on straight-line distances. Finally, it is important to recognize that, in practice, networks are rarely constructed from scratch but are often integrated into existing infrastructure—an aspect that could be further reflected in the model.

Overall, the project has also demonstrated the potential of the GAMS and GAMS MIRO combination. The integration of a visualization application within an optimization tool offers promising possibilities and makes the incorporation of new data visualizations remarkably easy. For the described project, the basic functions of the configuration mode were entirely sufficient to implement the desired visualizations. However, this did not fully utilize all of MIRO's capabilities. In particular, the ability to create custom renders using R-Shiny provides the flexibility to configure input and output features tailored specifically to the application. Moreover, utilizing the GAMS Engine, a cloud solution developed by GAMS, could further enhance the collaborative aspect of both implementation and usage.

Für das beschriebenen Projekt haben die Basisfunktionen des Configuration Mode völlig ausgereicht, um die gewünschten Visualisierungen zu implementieren, allerdings wurden dadurch auch bei weitem noch nicht alle capabilities von MIRO ausgeschöpft.

It has been shown that GAMS MIRO, due to its ease of use and customizable configuration options, is a suitable tool for both simple dashboards and more complex applications. Consequently, the application is well-suited not only for educational purposes but also for business applications.