Department of Mechanical Engineering

4CM70 Integrated System Design

Instructors: Tim Wilschut
Pascal Etman

Company contact: Jos Verdaasdonk (RWS)
Sjoerd Knippenberg (TU/e)

Group number: 7 - Rijkswaterstaat - Ship locks

Group members:

Sam Boeijen	1251147
Daan den Hartog	1003445
Teun Jansen	1224945
Levi Stevens	1259024
Derek van de Wiel	1023099
Jesse van den Helm	1029901

Eindhoven, January 19, 2022

Finding a structured method to determine an alternative route

Sam Boeijen - Daan den Hartog - Teun Jansen - Levi Stevens - Derek van de Wiel - Jesse van den Helm

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Rijkswaterstaat Ministerie van Infrastructuur en Milieu

Introduction

Rijkswaterstaat (RWS) is responsible for controlling water level and facilitate traffic by water. Locks play a important role in doing this. In case locks cannot be used as a result of renovations of calamities, an alternative route should easily and quickly be determined. To do this in a structured manner while showing the specifications of the route, a dependency structure matrix (DSM) will be designed. Each route has pros and cons based on different factors as intensity, operating times, waterway class and distance. By using the DSM these properties can be visualized and determined for every alternative route possible. Alternative routes will in practice always have to be verified by means of a map. Therefore a complete map of all corridors and interconnections has been made (Figure 2).

DSM structure

The elements of the DSM consist of the chambers of all locks and nodes from the map (Figure 2). An interconnection is formed when a node is connected with another without interference of a third node. The DSM (Figure 3 shows a piechart in the corresponding entry of the matrix to indicate a connection. This piechart contains information about the CEMT class, operating times and distance of the connection. In the diagonal of the matrix the CEMT class and utilization of the nodes are visualised.

Determining an alternative route

The map is more intuitive than the DSM and can therefore best be used to get an indication of the possibilities. Figure 1 shows the method to determine a possible route using the DSM.

Note that this is a schematic DSM that only indicates the presence of an interconnection. Subsequently, the resulting route can be analysed based on the properties of the passed dependencies and nodes. When all connections have been checked it will be clear whether the route is possible or in case problems occur, further action is required. To assist finding an alternative route a tool in the form of a Dijkstra algorithm is created. In the Dijkstra algorithm the start and end node are entered together with the class of the ship and the unavailable nodes. Based on these parameters it finds the shortest route to the destination.

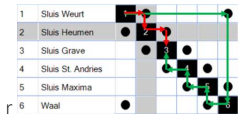


Figure 1. Method to determine an alternative route with the DSM

Conclusion

Corresponding to the model purpose, the designed DSM can be used to determine an alternative route in case a lock is unavailable. To understand the reason for selecting a route and to get feeling with it, the resulting route can be validated by using the map of all corridors. To improve the model, RWS can implement methods that take into account water levels, real travelling times and minimum waterway classes (CEMT). The Dijkstra algorithm can also be expanded such that it takes multiple properties into account.

Map of all corridors

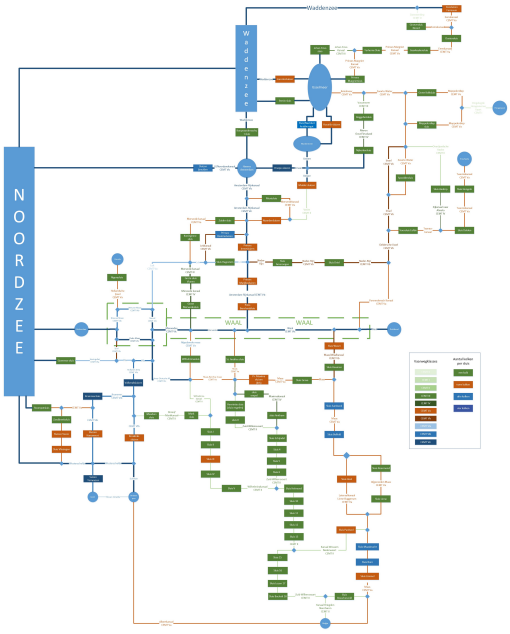
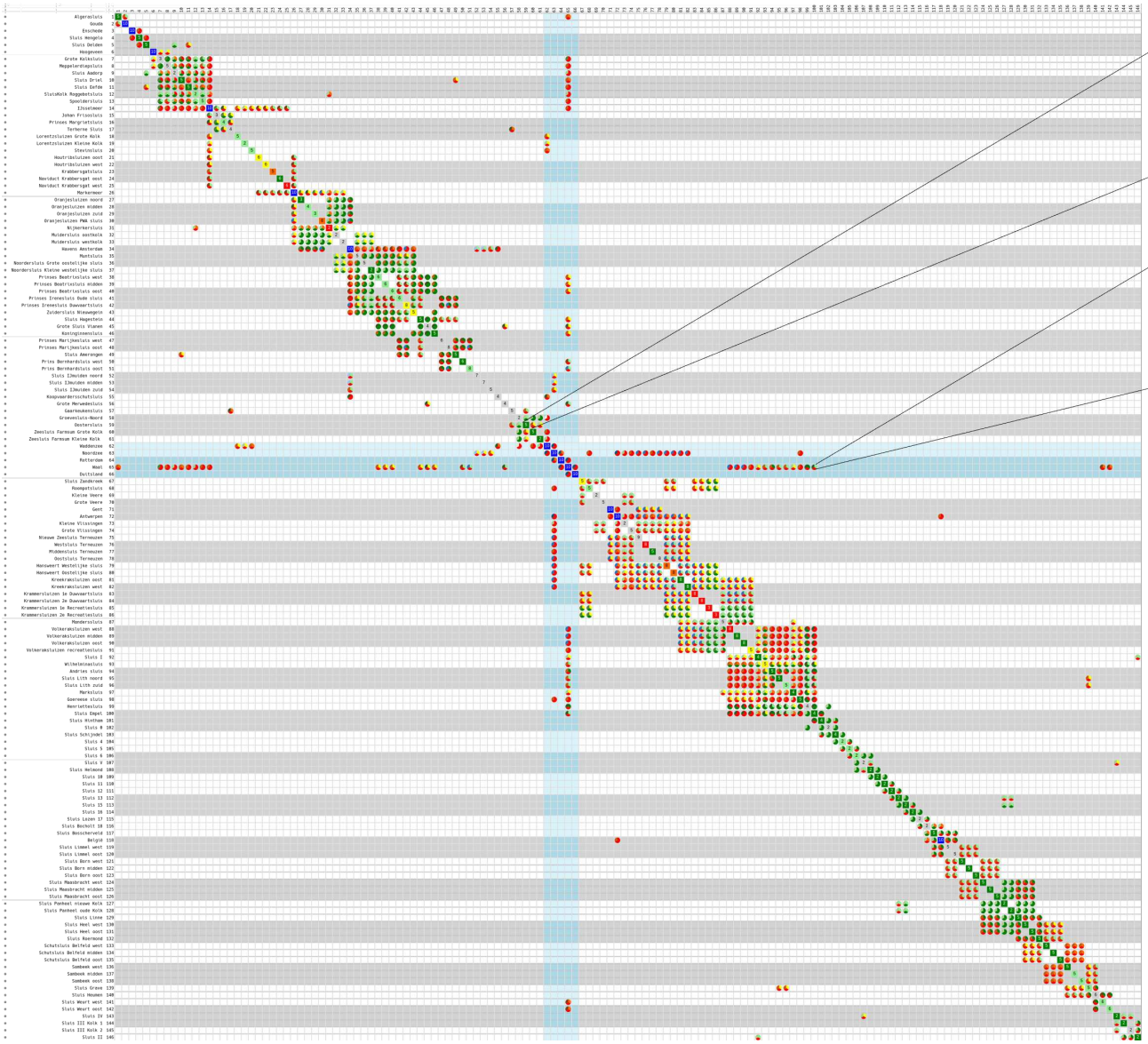
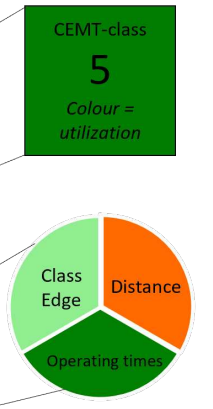
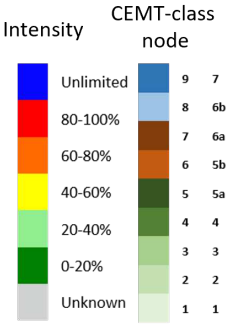
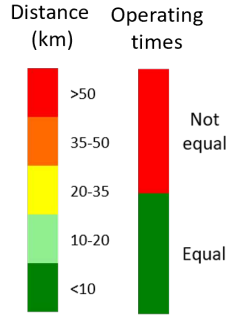


Figure 2. Map of all corridors









4CM70 INTEGRATED SYSTEM DESIGN
Master Course TU Eindhoven, January 21, 2022

Finding a structured method to determine an alternative shipping route

Sam Boeijen, Daan den Hartog, Teun Jansen, Levi Stevens, Derek van de Wiel, Jesse van den Helm

Abstract

Locks play a important role in facilitating waterborne transport. In case locks are unavailable as a result of renovations of calamities, an alternative route should easily and quickly be determined. This study looks for a structured way of determining an alternative shipping route on behalf of Rijkswaterstaat. A dependency structure matrix is a tool that will be used to this. First, all important properties that play part are analysed. By making use of data and documents of Rijkswaterstaat a database has been created. This database presents all information necessary to set up the dependency structure matrix. By using the dependency structure matrix in combination with a map containing all locks and waterways in the Netherlands, it should be possible to find the optimal alternative route. This report explains how the DSM has been created and shows how it can be used.

1 Introduction

This study is an assignment of Rijkswaterstaat (RWS). RWS is an organization that maintains and develops roads, water systems and waterways in the Netherlands. One of the most important taks of RWS is to provide a safe and smooth flow of traffic. This can either be traffic on land or water. RWS is also responsible for controlling the water level and the prevention of floods. In order to control the amount of water flowing trough waterway locks are being used. Locks can stop water while they also allow water and ships to pass when necessary. Locks can consist of multiple chambers. A chamber usually has two doors that holds an amount of water. By adding or removing water with pumps the difference in water level can be lifted and the ship can pass. It is clear that these locks are critical for traffic by water. However, a lot of these locks are planned to be renovated in the next four decades. This means that these locks, while being renovated, will not be able to be used. This does not only apply to renovation but also in case of repairs and calamities ship locks cannot be used. It is necessary to find an alternative route for a ship to still go to its destination. To do this in a structured manner whilst showing the specifications of the route, a dependency structure matrix will be designed (DSM). Each route has pros and cons based on different factors. By using the DSM these properties can be visualised and determined for every alternative route possible.

The model should make sure that these properties can be gathered quickly. All corridors and interconnections should be considered. The DSM will be delivered with a user manual to ensure that the DSM can be used by people from RWS to either determine an alternative route or get other information from it. In this manual lock Heumen and lock Weurt can be used as an example to clarify its usage. Alternative routes will in practice always have to be verified by means of a map. Therefore a complete map of all corridors and interconnections has been made.

Similar problems or model purposes have not been found in literature. Eppinger and Browning (2012) however introduces three comparable DSM methods to represent complex systems. Those models are divided in Product, Process and Organization Architecture DSMs. The system of RWS is considered a static architecture, so it is categorised as an Product Architecture DSM. This includes components, Functions and subsystems which are matched with the system of the Dutch waterways. However, a Product Architecture DSM has focus on a real products with its components and relations. The considered system of RWS is a network of routes trough the Netherlands. The challenge is to design a DSM of a huge network with similar components but a lot of different relations.

Wilschut et al. (2018) designed a DSM to support the development of the lock platform of RWS. This study contains interesting information and methods related to the construction of the locks. The objective of the research focuses on the similarity, modularity, and commonality of existing locks of RWS. This differs from the model purpose of the alternative routes, however the structure is comparable since the elements are considered the locks for some objectives. For example, the multi-level Markov Clustering algorithm is used. This can reduce the size of the DSM by creating multiple DSMs which have many high valued dependencies. This can maybe be used to separate the corridors (or high activity regions) in the Netherlands. This is also used in Eppinger and Browning (2012). In an example (3.9) a DSM^{3D} is used to highlight the differences of various designs. In the case of RWS this can be used to rank the relevance of the relations. All DSMs will have the same cross relations since the adjacent locks remain the same. But the relations vary in relevance and strength. A ranking on importance should be made to estimate the feasibility of an alternative route.

2 Case: system description

As mentioned in Section 1, the goal of this study is to design a structured method to determine an alternative route. In order to determine a alternative route a few specifications need to be known. First of all, the waterway has to have sufficient dimensions to accommodate a ship. By using the CEMT-class the minimal dimensions of locks and waterways are indicated. Another important property is the intensity of a lock. The intensity indicates the number of ships that pass the lock. This is important to keep in mind because for locks that already are being used close its maximum capacity it can be difficult to accommodate extra ships that are redirected. Operating times also play a part as well. These are the times that the locks are being operated, either locally or remotely. Operating times may not be the most important factor for determining a route as they might be adjusted based on their needs. However, they still should be considered. Last factor to consider determining an alternative route is the distance. An alternative route should desirably be as short as possible. How these properties are determined and how these are visualized in the DSM will be discussed in Section 3.

To structure the different routes within the Netherlands RWS created corridors. The corridors represent a starting point and a destination. RWS has a separated maps of all corridors. The corridors can overlap which makes it difficult to get a clear overview of all the canals and locks in a corridor. This complicates finding an alternative route whenever a lock cannot be used. To solve this problem a complete map of all corridors is made. Figure 1 shows the map. Besides the locks the map also shows the waterways that are labeled with colours to indicate the CEMT-class of both the locks and waterways.
In the map some nodes have been added to either get a clear starting point or destination or to simplify a route. These nodes are: Rotterdam, Amsterdam, Enschede, Hoogeveen, Gouda, Belgium, Germany, North Sea, Wadden Sea, IJssel Lake, Marker Lake, Gent and Antwerp. The Waal is also added as a node because of its many connections and its central location. This causes the Netherlands to be split into a northern and southern part which greatly reduces the number of necessary connections and gives the opportunity to obtain two separate DSMs.

3 DSM modeling method

This section explains in more detail how the DSM is modelled. To generate the DSM, the *ragraph*-Python software was used. BV (2020) With this software, a clear and understandable visualisation of the connections between locks, and their properties, is achieved. As explained in Section 2, the DSM should serve to check the alternative route concerning the relations between each lock. Figure 1 Shows the map with all nodes in the network. These nodes are the locks and some of the waterways which all can be considered as elements in the DSM. The explanation for this design and the data which is used is elaborated in this section. Some of the relations use assumptions. The assumptions are discussed in this section in case this provides understanding the relations. Additional assumptions regarding the data or the system are discussed in Appendix B.

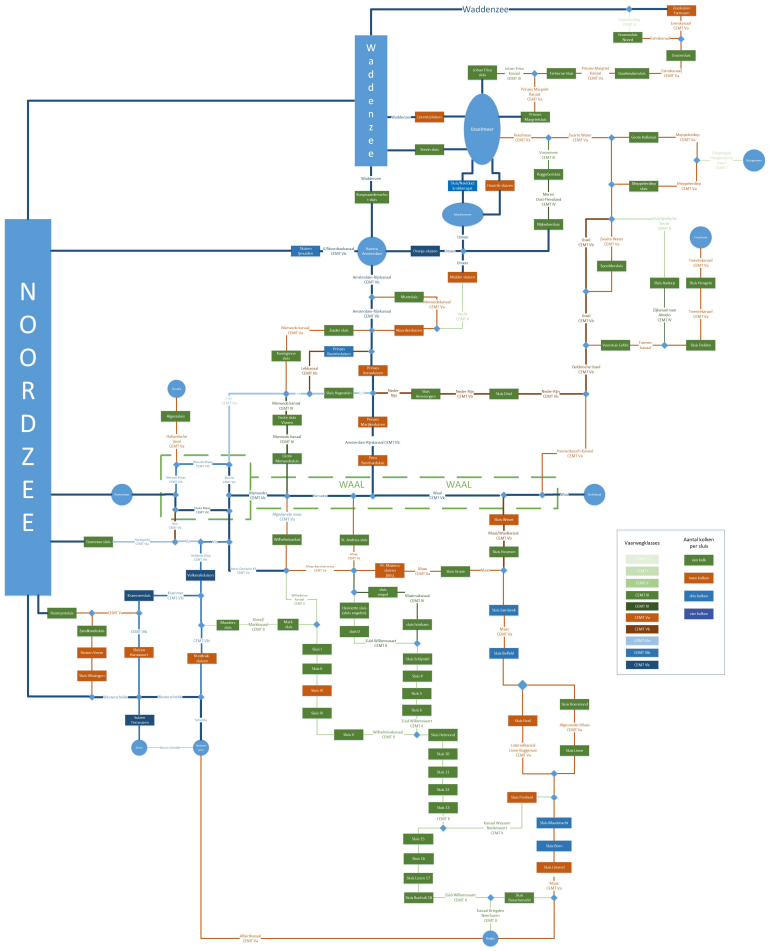


Figure 1: The complete map of the main waterways from the Netherlands

3.1 Collected data

Rijkswaterstaat provides several sets of data on which the relations can be based. The data which is used in this project is provided in Appendix C. It also explains the sources of the data in more detail. As stated in Section 2, the relevant relations for the DSM are CEMT-class, Utilization, Distance and the operating times. These relations are explained in this section.

3.1.1 CEMT-class

The CEMT-class determines the dimensions of the ships, locks and waterways. The higher the CEMT-class, the larger the ship or waterway. For example, a ship with a Class V can pass a lock of Class VI but can not pass a lock of Class IV. The Class of the waterway or lock should at least be equal to the ship Class to allow that ship to pass trough. It is for that reason important to know the maximum CEMT-class which is allowed to pass trough a lock or waterway. The CEMT-class of each lock is extracted from the Excel-datasets of RWS database (2019). The waterway classes follow from the corridor maps of RWS. Locks or waterways for which this data was not available are compared with similar locks or waterways based on dimensions. This data is used in the DSM and elaborated in Section 4.

3.1.2 Utilization

The utilization of a lock gives an idea if an alternative route can handle the traffic. It depends on the capacity and current intensity of a lock. The capacity is defined as the theoretical traffic that can pass trough a lock. What this exactly means is elaborated below. The intensity is defined as the current traffic that passes trough a lock. Both parameters are expressed in payload per year.
The intensity and capacity are calculated below.

The intensity is expressed in payload per year using the next equation:

$$I_p = I_n * (1 - \gamma) * \frac{P_x}{1000000} \tag{1}$$

It is important to mention that the payload of the intensity is based on an assumption. An amount of ships which passes the lock is known, but the distribution of CEMT-classes passing trough the lock is not known. An average is taken based on maximum CEMT-class a lock allows. The assumption is that 50% of the ships has the maximum CEMT-class of the lock, 20% the max CEMT-class -1, 20% the max CEMT-class -2 and 10% the max CEMT-class -3. The minimum CEMT-class is 2, so all calculations with a Class lower than 2 are set equal to CEMT II. The capacity is also expressed in payload per year using the next equation.

$$C_{hour} = (1 - \gamma) * \frac{A_{lock}}{A_{CEMT}} * \frac{60}{t_{cyc}} \tag{2}$$

$$C_p = \frac{C_{hour}}{1000000} * T * P_{CEMT} * 0.6 \tag{3}$$

Equation 3 is multiplied by 0.6 as can be seen. In practice, the waiting time of ships increases exponentially above a value of 0.6 of the total capacity. This is caused by the amount of ships that should wait for the next cycle. In Rijkswaterstaat (2020) is also mentioned that a value of 0.6 causes a bottleneck.

Table 1: Parameter symbols and units

Symbol	Parameter	Unit
A_{lock}	Area of the lock	m^2
A_{CEMT}	Max area of the ship with corresponding CEMT-class	m^2
C_p	Capacity based on payload	$mlnton/year$
C_{hour}	Capacity based on payload per hour	$ton/hour$
I_n	Intensity based on amount of ships	$ships/year$
I_p	Intensity based on payload	$mlnton/year$
t_{cyc}	Event duration: The time it takes for a ship passing a lock	min
T	Operating times	$hours/year$
P	Payload of a ship	kg
P_{CEMT}	Max payload of a ship with certain CEMT-class	kg
γ	Part of recreational ships	-

The utilization represents the relative inverse of the Capacity minus the intensity $[(C_p - I_p)/C_p]^{-1}$ of the corresponding lock. If a lock with a high utilization is considered as an alternative, it should be compared with the intensity of the unavailable lock. This intensity is not visible in the DSM but should be known since it is an input. This comparison is elaborated in an example in Section 4.

KOOMAN-model for intensity

The capacity is a very sensitive parameter with a lot of uncertainties and assumptions. At RWS a model often is used called the KOOMAN model (Kooman and Bruijn (1975)). This model simulates the arriving and leaving times of the ships which depends on the size and payload. The inputs (existing of lock dimensions, cycle duration, operating times, share of loaded ships, share of recreation ships and average payload) generate a theoretical capacity and permissible intensity. The permissible intensity should approach the value of the capacity which is calculated for the utilization. Not all input data is available for every lock of RWS. It is not possible to compare the model with the previous calculations of the capacity in a fair way due to the assumptions. But since the KOOMAN model takes into account more dependencies, it is considered more reliable. If the require data is available, it is recommended to calculate the capacity in the future with the KOOMAN model if there is sufficient data available. An important factor is the cycle duration. This equals the times it takes for the doors to open/close added to the time it takes to level the water.

3.1.3 Distance

The distance between each set of two locks is calculated using the coordinates of each lock. The coordinates were taken from Rijkswaterstaat (2021)

In this data from Rijkswaterstaat, the coordinates of each lock can be found, in X- and Y-coordinates, based on the RD-coordinates (*Rijksdriehoekscoordinaten*). The distance used in the DSM is the straight-line (absolute) distance between two locks, for each pair of connected locks. This causes a discrepancy between the distance in the DSM and the actual distance that needs to be travelled by ship. However, this is the only viable way to calculate the distances without having to measure out each connection by hand.

3.1.4 Operating times

When determining an alternative route for a ship, it is useful to know whether the operating times of the locks correspond. Otherwise it might occur that the ship is sent via a lock that is not operational at that time. This data is freely available from the government (Rijkswaterstaat (2022)).

Table 2: Overview of used relations

Factor	Diagonal/Entry	Explanation
CEMT-class	Diagonal and Entry	Because of its importance, the Class is visualized on both locations in the DSM. In the diagonal it represents the CEMT-class of the corresponding lock, but in the entry it represents the lowest CEMT-class of the corresponding locks and waterway in between.
Utilization	Diagonal	The utilization represents the relative C_p–I_p of the corresponding lock. If a lock with a high utilization is considered as an alternative, it should be compared with the intensity of the unavailable lock. This intensity is not visible in the DSM but should be known.
Operating times	Entry	For the operating times it is relevant to know if the operating times between to locks are similar. If not, this should be communicated with the shippers when taking this alternative route.
Distance	Entry	This factor is visualized in the entry since it represents the absolute distance between a connection of locks.

3.2 DSM structure

It is important to show the relations between all direct connections of the locks. Also waterways and harbours are important to show in the DSM since these are often the destinations of the ships. This however, results in a very large amount of elements and thus in a large DSM. For that reason the DSM is divided into two separate structures: locks that are geographically north of the Waal, and locks that are geographically south of the Waal. This is possible because the Waal is a perfect division between these networks since the Waal contains no locks, and it traverses The Netherlands entirely from east to west. The Waal, and to a lesser extend the Nooordzee, serve as a Bus in the DSM, connecting the north and south network.

3.3 Element attributes

Before explaining the relations between elements, individual information about the elements is also relevant. This information can not be visualized in the cells since it only applies to the lock itself. A DSM constructed in Eppinger and Browning (2012) (example 5.7), showing a network of stakeholders, uses the diagonal to show the implications in the network for that specific stakeholder. Similar to this, the diagonal in the DSM that will be created in this study is used to visualise information about the locks itself. It concerns the utilization and the CEMT-class of the lock. The reason for that is that it is important to know the utilization and CEMT-class of the locks when checking the alternative route.

3.4 Element dependencies

The DSM entries present the dependencies between two elements. As just explained, determining the best alternative route requires information about multiple dependencies. Hence, it is necessary to be able to visualise these in just one cell of the matrix. Eppinger and Browning (2012) (example 3.3) provides

a good method to do this. In this example a printing company designed a DSM that is used to present multiple dependencies of the components of a printing system in one cell. One cell can contain four colours that each indicates the presence of a type of interface. Unlike Eppinger and Browning (2012), this study will not only use the colours to indicate the presence of a connection, but will also indicate the strength of the dependency. This requires different colour tones. The three dependencies which are visualized in the entries are CEMT-class, distance and operating times. All parameters are considered a relation since they are compared with each other.

4 Results

4.1 Cell explanation

The data for several different variables is presented in the DSMs (Figure 6, Figure 7 and Figure 8 of Appendix A). Not all data, however, is useful as a relation between two different locks. That is why the data is subdivided into entries on the diagonal of the matrix, applying only to the lock in question, and the relations between two locks.

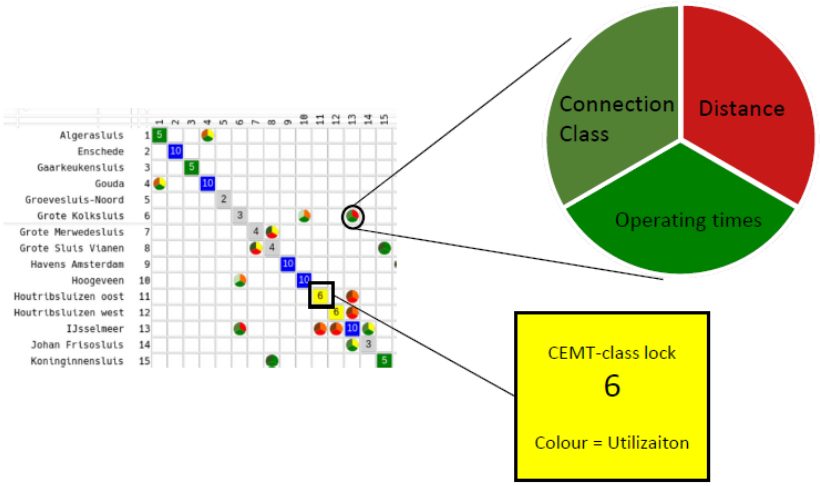


Figure 2: Schematic representation of the diagonal and off-diagonal entries in the matrix

Diagonal

On the diagonal two different variables are presented. Firstly, the CEMT-class of the lock or waterway is displayed with a number. This data is useful, since it immediately shows the largest possible class ship that can pass through the lock. The class is displayed using a number, ranging from 1 to 10. These numbers represent the CEMT-classes I - VIc.
Furthermore, the utilization of the lock is displayed, as calculated in Section 3.1. This is done by using a colourscheme, described by the following:

0.0 - 0.2	green
0.2 - 0.4	lightgreen
0.4 - 0.6	yellow
0.6 - 0.8	orange
0.8 - 1.0	red

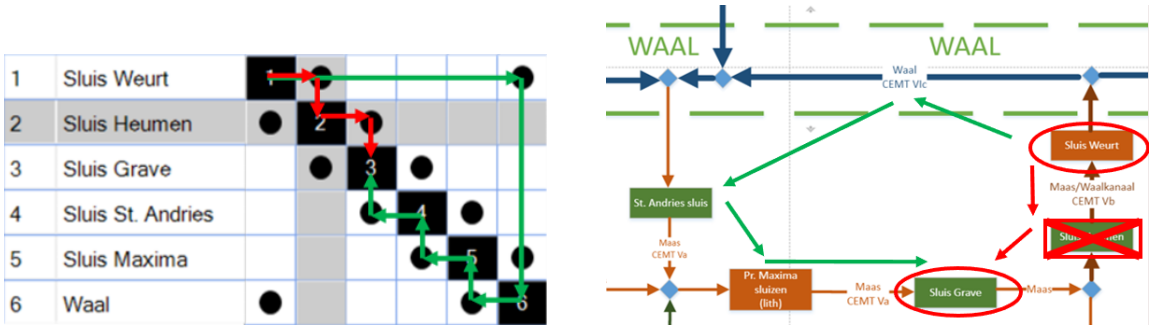
For a lock with high utilization, a red colour is displayed in its corresponding diagonal cell. This immediately indicates that it might be better to opt for a different route, when searching to reroute a large amount of ships. One might rather choose for a lock that shows a green colour on the diagonal, indicating low utilization. For elements on the axes that do not represent locks (cities, seas and lakes), the colour blue is used. For elements whose data is unknown, the diagonal is coloured gray.

Cell

In the cells, data is displayed that applies to the connections between two locks. Each cell that is a connection between two locks displays a piechart, consisting of three different elements; CEMT-class, corresponding operating times, and distance. A schematic representation is displayed in Figure 2.
The class that is displayed in the cells is taken as the lowest class between the source lock, the waterways in between, and the target lock. Showing this, immediately the highest possible class ship that can travel the route between the two locks, and also pass through both locks, can be deduced. When only the classes of the locks are shown in the cells, it might occur that these classes are both higher than the class of the connecting waterway. This could lead to alternative routes for ships that turn out to be impossible to pass. The colourscheme used to indicate the class is the same as in Figure 1.
The next section in the piechart is the correspondence of operating times between locks. This attribute can only be one of two values; true or false. For corresponding operating times, a green colour is attributed. Otherwise, a red colour is assigned. Important to note is that only a green colour is used for those elements that have a 100% correspondence.
Lastly, the distance between two locks is indicated in the cell. For the distance, the same colourscheme is used as for the utilization on the diagonal. Starting with green for a distance <10 km, then ranging up to 20 km, 35 km, 50 km and red for connections which are >50 km apart.
Notice that for three attributes in the matrix a similar colourscheme is used, ranging from red to green. This is done to enhance readability. A red colour for these attributes indicates suboptimal conditions. Therefore, entries that show predominantly red colours instantly catch the eye, telling the reader that it is better to search for alternatives. Green colours on the other hand, indicate advantageous properties, pointing in the direction of a proper rerouting.

4.2 Alternative route check

The DSM can be used as a tool to search for alternative routes. It is recommended to keep the map from Figure 1 close by. This overview is more intuitive when looking for a specific route. Then, using the DSM, this route can be checked to confirm whether it is the optimal alternative.
For explanatory purposes,a simplified version of the DSM is displayed in Figure 3a. This version only shows the connections between locks, without the colours in the piecharts or on the diagonal. The equivalent part of the map in Figure 1 is shown in Figure 3b. In this mock-up scenario, the lock in Heumen is inaccessible. Therefore, an alternative route has to be found.
The connections to other locks are shown in the rows of the matrix. In Figure 3a the lock in Weurt shows only two connections; Heumen and the Waal. Since Heumen is no longer an option, the Waal is the only other option to travel to. The green arrow can be followed as displayed in the matrix. When a connection is followed, the user ends up at the corresponding cell at the diagonal. From this cell, the corresponding row can be used to see which connections are possible from there. From the Waal, one can follow the waterways to the St. Andries lock. Then the next block on the diagonal is reached. This process can be continued until the desired lock is reached, in this case the lock in Grave.
The complete DSM is more complicated, but the principle stays the same. In the complete DSM, also weighing factors like the distance and CEMT-class can be seen. The DSM is designed in such a way that the most optimal routes show the most green colour.



(a) Schematic demonstration of how to use the DSM in finding an alternative route

(b) When the lock in Heumen is not operational, a different route has to be taken to get to lock Grave

Figure 3: Example of finding an alternative route using the DSM (left) and the map (right)

4.2.1 Automated path finding (using Dijkstra's algorithm)

With all the data structured in the computer a lot of analysis and calculations can be done using algorithms. A useful additional tool could be an program which calculates the optimal route given a range of parameters like: unavailable locks, ship class, amount of ships and other current data. As a proof of concept for this idea a path finding algorithm was implemented in python using the Dijkstra method (Appendix E). The program calculates the shortest possible path between two nodes (locks) in the network given a minimum CEMT-class and a number of unavailable locks.
Dijkstra algorithm is a path finding algorithm which in its simplest form can be used to calculate the shortest route between two nodes in a network by giving weights (distances/travel time) at all the edges between the nodes. The program then calculates the minimal distance and the corresponding path. To make the program more applicable some extra statements have been added so the program can check whether the path has the correct minimal CEMT-class. Next to that some functionality has been added to remove nodes (locks) from the network .
In Figure 4 an example can be seen of the input parameters for a desired route check. In Figure 5 the corresponding solution can be seen on the network map.Three different configurations of the same route have been checked. First with a ship of CEMT III, then of a ship with CEMT IV, then with both the Johan Friso Sluis and the Princes Margrietsluis unavailable and lastly with a ship that is to big. The program will give a different recommendation route depending on these parameters. For the third option the route will go all the way via the Waddenzee and Farmsum. The fourth option will result in a distance of 999999 which means there is no route possible.

```
Solution with CEMT class: 3  and unavailable nodes: []
Vertex                        Distance from Source                  Path

5 Lorentzsluizen Grote Kolk  --> 7 Terherne Sluis             74             5 Lorentzsluizen Grote Kolk , 54 IJsselmeer, 55 Johan Frisosluis, 7 Terherne Sluis,


Solution with CEMT class: 4  and unavailable nodes: []
Vertex                        Distance from Source                  Path

5 Lorentzsluizen Grote Kolk  --> 7 Terherne Sluis             77             5 Lorentzsluizen Grote Kolk , 54 IJsselmeer, 9 Prinses Margrietsluis, 7 Terherne Sluis,


Solution with CEMT class: 4  and unavailable nodes: [55, 9]
Vertex                        Distance from Source                  Path

5 Lorentzsluizen Grote Kolk  --> 7 Terherne Sluis             216            5 Lorentzsluizen Grote Kolk , 66 Waddenzee, 2 Zeesluis Farmsum Grote Kolk, 1 Oostersluis, 0 Gaark
eukensluis, 7 Terherne Sluis,


Solution with CEMT class: 7  and unavailable nodes: []
Vertex                        Distance from Source                  Path

5 Lorentzsluizen Grote Kolk  --> 7 Terherne Sluis             999999         7 Terherne Sluis,
```

Figure 4: Code response for a range of inputs.

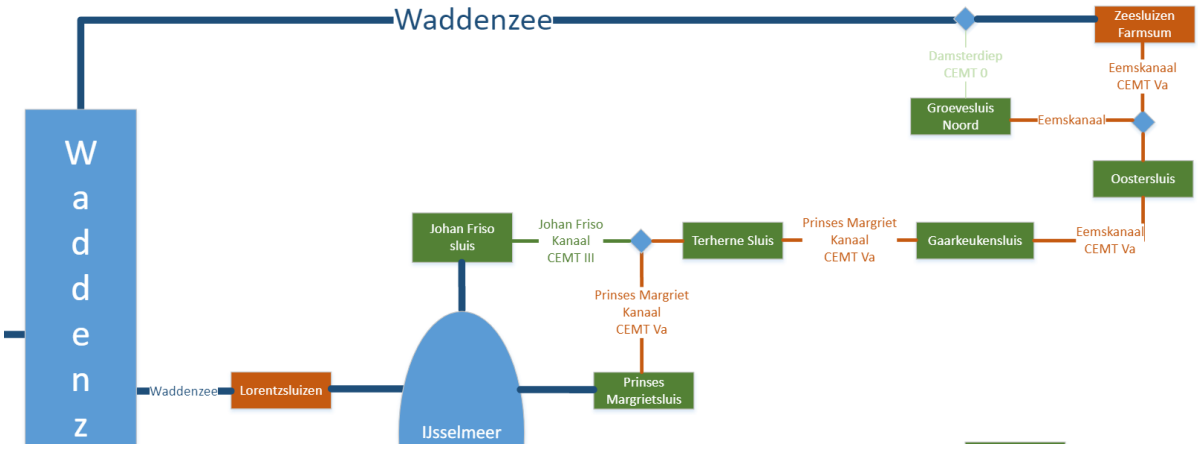


Figure 5: Relevant part of the network map for a route from the Lorentzsluizen to the Terhernesluis.

Although currently in a very rudimentary form, this program could be very useful as a complementary system for investigating possible routes. The program could for example be expanded with data of the capacity of locks and the yearly usage. The program can then automatically calculate whether the proposed trajectory has enough capacity and were possible bottlenecks could be. Next to that,

data of bridge heights, current water depth, one way connections, and exact operating times can all be incorporated fairly easily which could result in a comprehensive algorithm which can propose a range of different routes which can then be checked manually.

5 Conclusion

The goal of this project was to provide a DSM which visualizes the information which is needed to find alternative shipping routes when locks become unavailable. RWS provided several datasheets from which useful data is gained regarding the concerning locks in the Dutch water network. This data is processed to get the most relevant parameters which are used in the DSM. Corresponding to the model purpose, the designed DSM can be used by itself to determine an alternative route in case a lock is unavailable.
Additionally, the corridor map (Figure 1) can also be used to find possibilities for an alternative route. It mainly provides information about the CEMT-class of waterways between the locks but lack information about the distance and capacity. Lastly, the Dijkstra algorithm can also provide a starting point for an alternative route based on the shortest distance of the interconnections. This route should be the best alternative based on distance but does not take capacity and operating times in consideration. The DSM can then be used to check the alternative route on CEMT-class, the distance, operating times and utilization.
The DSM uses the diagonal to provide information about the elements and the entries to provide information about the relation between the elements. For structural reasons the DSM is divided in two different parts: the north of the Dutch water network and the south of the Dutch water network. These networks are separated by a bus which are open waters like the North Sea, IJssel Lake and the Waal. These waterways do not contain any locks but are displayed in the system as a lock with no CEMT-class maximum. The elements in the north and south network are clustered based on its absolute distance. That means that short connections between locks are placed next to each other in the DSM for a clear and logical order. This clustering method motivates looking for close by connections first before looking to locks that are further away, which could aid in finding a short alternative route when using the DSM to analyse the system.
To conclude, the DSM provides the most information about an alternative route if used correctly. However, not everyone is familiar with using a DSM. For that reason, the corridor map and Dijkstra's algorithm can help in providing context for an alternative shipping route.

5.1 Recommendations

There are several aspect of our model which could be improved, most of these are based around improving and expanding the dataset. Below is a list of different aspects that could be improved or expanded:

Travel time
To make the DSM more realistic, it would be better to use travelling times in stead of absolute distance. These are based on the real distance a ship has to travel. It also offers the opportunity to include flow direction and flow velocity. The real distances are measurable from the available maps. But to calculate the actual travel times, it is required to have a mathematical model which also predicts the travel speed dependent on shipspeed, flowspeed and weather conditions. A second way to get the data is to measure the time a ship actually travels between locks.
Secondly, the travel time could also include the estimated time to go through the lock (schuttijd). Together with the traveltime this could make a quite accurate model for the total travelling time of a route.

Improving capacity model (based on Kooman model)
The utilization is now based on calculations with a measured cycle time. As mentioned, the KOOMAN model calculates these times based on certain lock dimensions and can estimate an actual capacity. However, the provided data will not give significant more reliable results when using the KOOMAN model. If there is better data from actual cycle times and better data from the distribution of passing CEMT-classes, a better estimation from the average payload per hour can be made. This can lead to a more reliable factor for the utilization. See Kooman and Bruijn (1975) for more information about the model.

Display possible increase capacity
It might be useful to indicate whether the current capacity is the actual limited capacity. Increasing

operating times and staffing can increase the capacity significantly. The DSM now only shows whether the operating times agree. When there is lack of capacity, the operating times can be extended but only if the lock is not on a 24 hour schedule. Also staffing could be increased when the utilization or capacity will be increased. It might also be an idea to specifically show the fraction of the current operation per day on the diagonal. This way it is easy to see how much the operating times, and thus capacity, can be increased.

Water level, waterway closure and unidirectionality
The DSM is designed without taking the water level into account. Some rivers and canals sometimes are faced with high or low water levels that are not suitable for traffic. This also depends on weather conditions and differs every season. The DSM can therefore be improved by giving the opportunity to close waterways or to select them as unidirectional.
Implementing unidirectionality and closure can be done in a variety of ways. One possibility could be to expand the database by recording all the waterways(possibly in sections) and travelling directions used for each of the connections. Closure can simply be implemented with a function in python that removes all connections that use the specific waterway that is not available. Unidirectionality can be implemented by writing a function that removes all connections in which a certain travelling direction of a waterway is used.

Municipal locks
The study in this report neglects locks that are managed by the municipalities because RWS has little to no data of them. Usually these locks are used for recreational use and have class I, II or III. Therefore they have little added value by means of providing alternative routes. However, to increase the routing possibilities especially for recreational usage, these locks can be taken into account.

Minimum CEMT-class
The North Sea and Wadden Sea are open seas and are faced with waves and turbulence a lot more than canals and rivers are. Ships therefore have to be of a minimum CEMT-class to be allowed on open sea. This study neglects the minimum classes that are required to use an route.

Expanding automatic path planning
A Dijkstra based model could be developed which incorporates all the available data besides the CEMT-class and distance which is currently used. It could for example be possible to project the intensity of a closed lock onto the other locks to automatically calculate the the locks on a route have enough free capacity for it to be a viable route. Next to that the more advanced model for travel time can be used. The program could also be improved by providing a range of possible routes, ranked on travel time and/or minimum lock intensity. Lastly, a User Interface could be very advisable.

References

BV, R. I. (2020). Ragraph 1.8.7 documentation.

database, R. M. (2019). Informatie_sluizen_vaarwegen.xlsx.

Eppinger, S. D. and Browning, T. R. (2012). *Design structure matrix methods and applications*. MIT press, Cambridge.

Kooman, I. C. and Bruijn, P. D. (1975). Lock capacity and traffic resistance of locks. Government publishing office - The Hague.

Passeertijden, R. (2021). Passeertijden.xlsx.

Passsages, R. (2021). Passages.xlsx.

Rijkswaterstaat (2020). Richtlijnen vaarwegen 2020.

Rijkswaterstaat (2021). Geoweb beeheersplan rijkswateren 2016-2021.

Rijkswaterstaat (2022). Bedieningstijden van sluizen en bruggen.

Schuttingen, R. (2021). Schuttingen.xlsx.

Wilschut, T., Etman, L., Rooda, J., and Vogel, J. (2018). DSM modeling and requirement specification in developing a product platform for locks. Lehrstuhl fur produktentwicklung und leichtbau.

A Designed DSMs

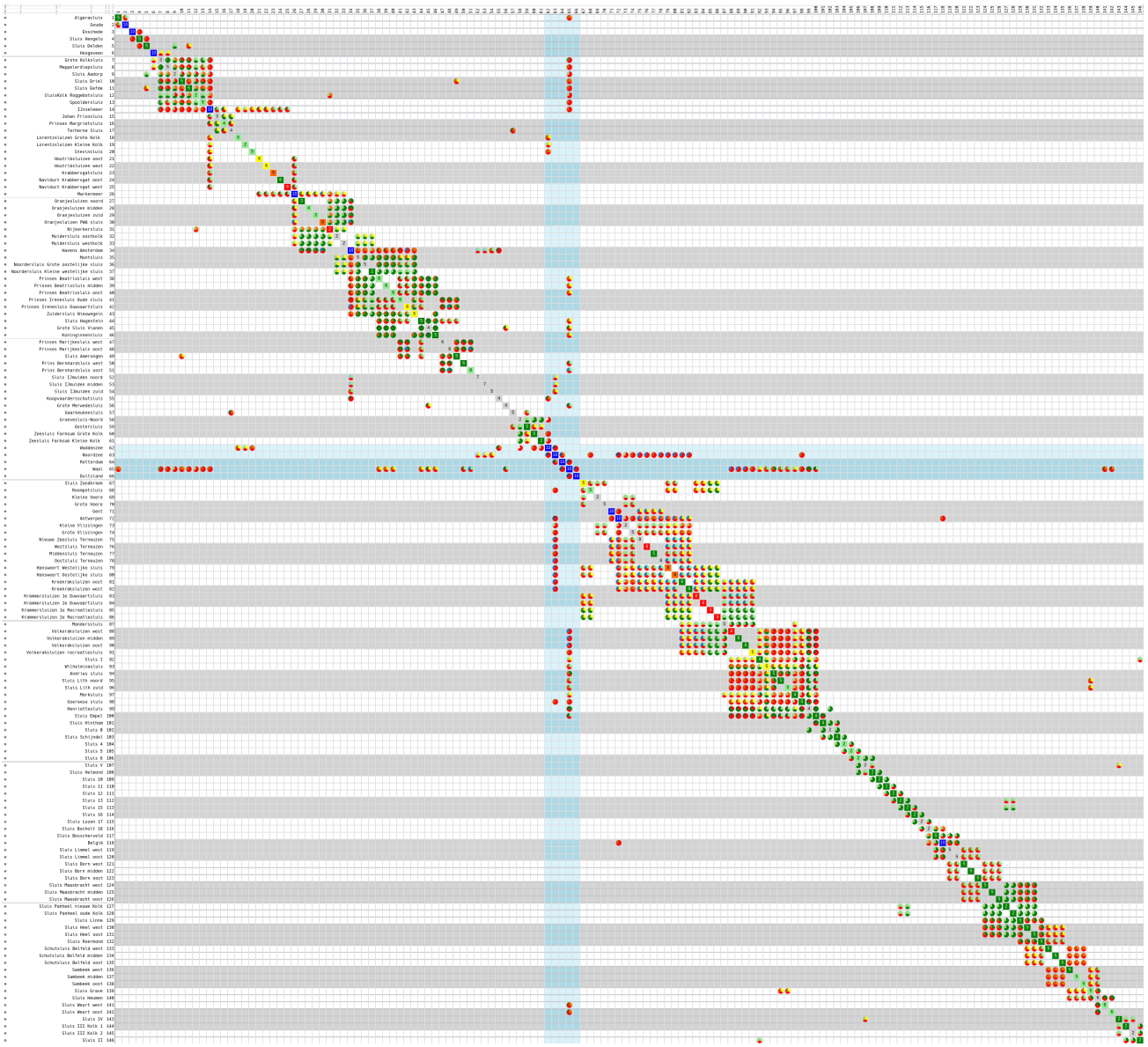


Figure 6: DSM of the main water network of the Netherlands

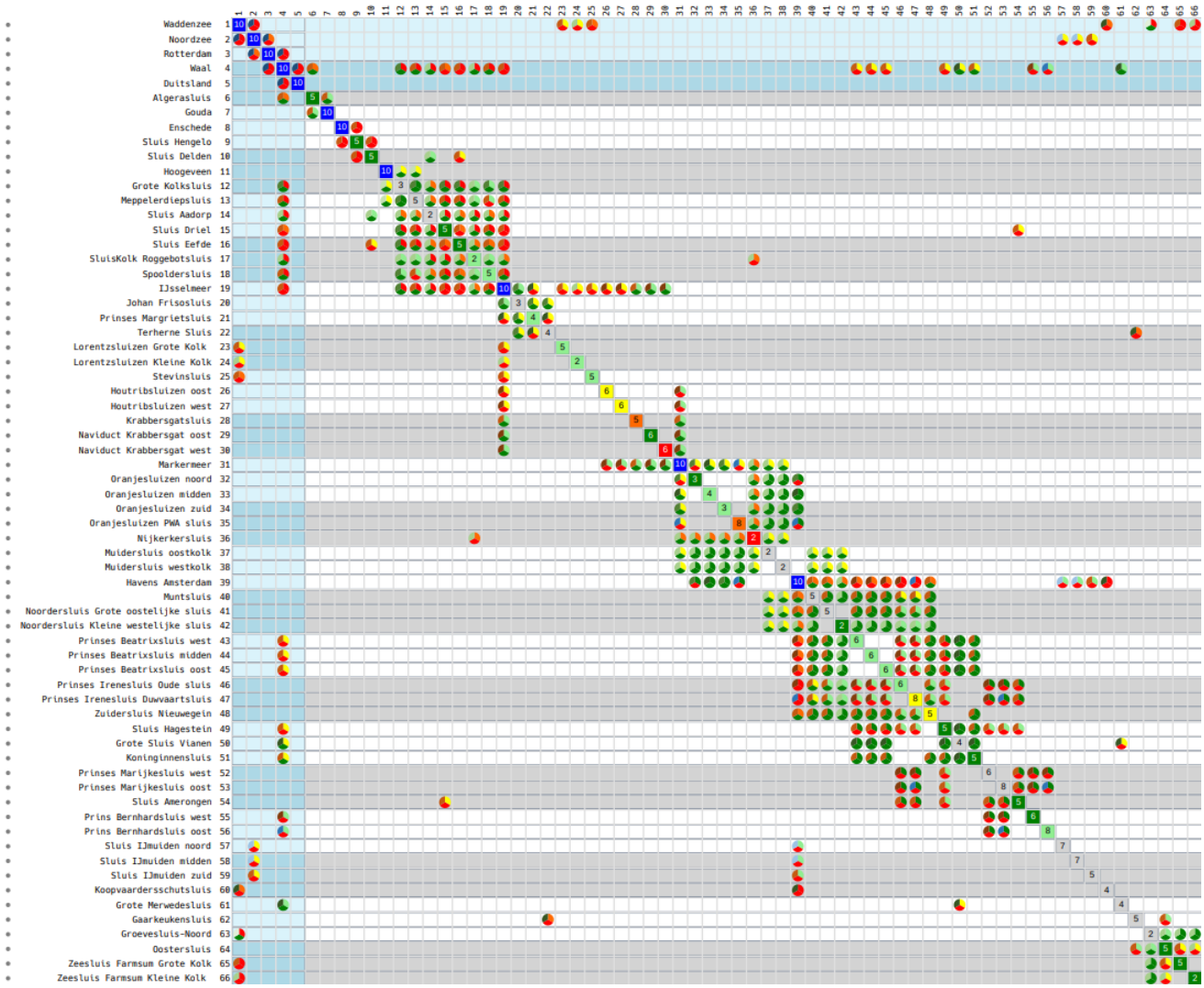


Figure 7: DSM of the main water network of the north of the Netherlands

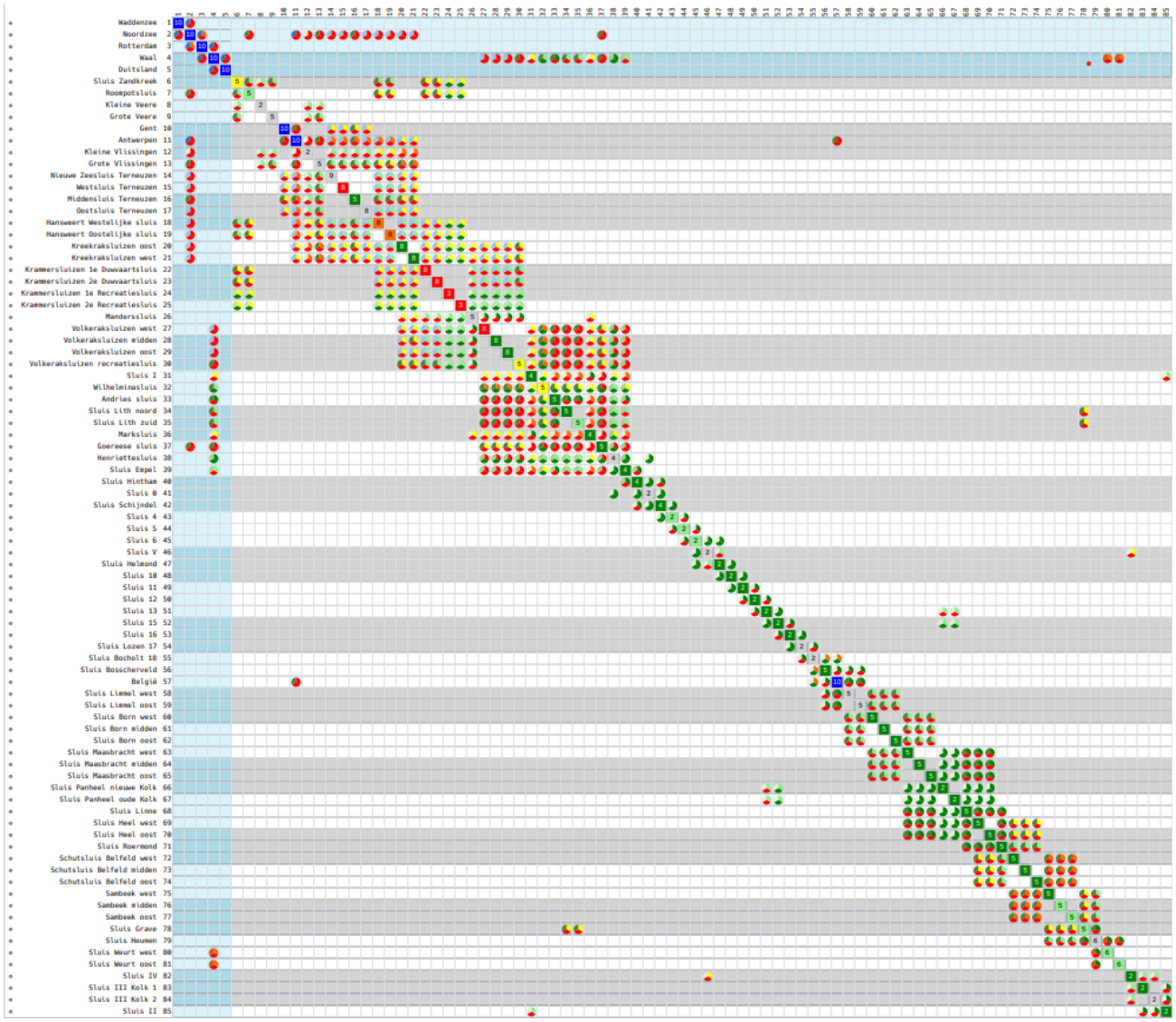


Figure 8: DSM of the main water network of the south of the Netherlands

B Assumptions

- It is not possible to use locks Biesbosch en Grevelingenmeer for an alternative route because they do not lead to another lock or node and therefore are removed from the database.
- Waterways have no minimum CEMT-class, only a maximum.
- Water level has not been taken into consideration. Therefore locks that depend on the water level are considered to be available.
- Locks that are managed by municipalities are neglected.
- In case information about the CEMT-class of locks was not available and could not been found, the lowest class of adjacent waterways or locks was allocated. In case of kleine noordersluis, kleine sluis Vlissingen and kleine sluis Farmsum a CEMT-class has been allocated based on their dimensions. These locks are part of a complex and therefore using previously described method would not yield a realistic result.
- There is no data available of the distribution of CEMT-classes which passes a lock. This distribution has a huge impact on the calculations of the capacity and current intensity. There is chosen to estimate the distribution based on experiences from RWS. This can still deviate from the real case. The distribution is explained in Table 4
- The data regarding the cycle time deviates from the actual time it takes for a ship to pass the lock. The data contains mistakes since some time values are negative or very low. All outliers below five minutes are neglected in calculating the average to get a better approach of the real time values.

C Database

The database contains all information that has been used to create the DSM. The database exist of multiple datasheets: Data alle sluizen (Table 3), capaciteit (Table 4) and verbindingen (Table 5). In the tables all columns of every sheet are discussed to explain what is represented in each column.

The datasheet: data alle sluizen (Table 3) contains the data of all locks and added nodes. This sheet can be seen as the main sheet from which the other sheets import data. It also provides additional information that characterises the locks. The datasheet: capaciteit (Table 4) is used to calculate the utilization-factor that is shown in the diagonal of the DSM. The methodology of this calculation is discussed in Section 3.1.2. The datasheet: verbindingen (Table 5) present all connections from every node in the database. The dependencies presented in the DSM uses the data from this datasheet.

Table 3: Explanation of columns of sheet: Data alle sluizen

Colomn	Parameter	Explanation and source
A	Naam	The name of the locks need to be able to distinguish the different chambers in the same complex. To do this the names of locks with multiple chambers have been given an additional indication. This indication can either be based on location (North, East, South, West, middle) or size (large or small). In some cases just a simple 1 or 2 is added to keep the name similar to the information that has been handed by RWS. As explained in Section 2 extra nodes are added to simplify routing. The name of these nodes correspond with the name of the sea, lake or city.
B/C	ID	To simplify indicating the locks RWS uses id-numbers. In case the locks belong to the same complex only the last part of the id-number will differ. Hence, by using the id-numbers locks with multiple chambers can easily be identified. Instead of given the added nodes a id-number, the name has been used in this column.
D/E	Noord/Bus/Zuid	To cluster the DSM this column indicates the cluster the locks belongs to. The nodes that connect the Northern and Southern part of the Netherlands form the bus. The type of cluster is connected to a number.
F	Corridor	RWS uses corridors to indicate different routes and are therefore also listed in the database. Added nodes are not assigned to a corridor.
G/H/I	Bedieningstijden	These are the times that locks are being operated (also see Section 3.1.4). Some locks are remote-controlled and therefore can always be operated .
J	Uitzonderingen bedieningstijden	Operating times of some locks may differ. This column indicates whether or not this is the case. Exceptions can be locks that are permanently open or locks that are only being operated on request.
K	Percentage bedieningstijden	The percentage of time of an average day that locks are being operated.
L/M	CEMT-klasse, vaarwegklasse nr.	The CEMT-class of the locks (also see Section 3.1.1). Some CEMT-classes are divided into sub-classes (Va,Vb,VIa,VIb en VIc). To be able to indicate these in the DSM they are given an integer that correspond with a CEMT-class. Up to and including IV every class corresponds with its integer. From here on class Va=5, Vb=6, VIa=7, VIb=8 en VIc=9. All added nodes are considered to have larger dimensions than the waterway that connects this node. In the DSM the dependency displays the smallest class of the arrival point, the destination and the waterway between them. Therefore all added nodes are given class nr. 10. Note that doing this affects the diagonal of the DSM but not the interconnections of the DSM. Also note that by doing this it is very easy to spot all the added nodes.
N	Capaciteit	The capacity is calculated with Equation 3 explained in Section 3.1.2. It represents the theoretical maximum possible amount of payload per year in million ton per year.
O	Intensiteit	The intensity is calculated with Equation 1 explained in Section 3.1.2. It represents the current amount of payload per year in million ton per year.
P	Utilization-factor	This factor is used to visualize the utilization in the DSM. It is calculated by $[(C_p - I_p)/C_p]^{-1}$ and represents a relative factor of the 'used space' per year based on payload.
Q/R	X,Y-coordinaat	These coordinates are used to calculate the straight-line distance from all connections (also see Section 3.1.3). Some nodes represent a large area and therefore do not have specific coordinates. These nodes are North Sea, Wadden Sea, IJssel Lake and Marker Lake. To be able to calculate an estimated distance, the most logical (usually most central) point has been used to determine its coordinates. Note that these results therefore may deviate a lot from the distance along the fairway. The coordinates of nodes Belgium and Germany are chosen to be on the border at the border-crossing waterway.

Table 4: Explanation of columns of sheet: capaciteit

Column	**Parameter**	**Explanation and source**
A	Naam	Name of the lock.
B	ID	ID number of the lock.
C	Percentage bediening	Operating times in percentages imported from column K of Table 3.
D	Uren per jaar	The total amount of operational hours per year. Based on general operational hours per day.
E	Intensiteit per jaar	Amount of passing ships per year. database (2019)
F	Intensiteit recreatie	Amount of passing recreational ships. Passsages (2021)
G	Aandeel recreatie	The share of recreational ships.
H	CEMT-class	CEMT-class of the lock imported from column L of Table 3.
I	Vaarwegklasse nr.	The numerical value of CEMT-class imported from column M of Table 3. 1 t/m 4 correspond to CEMT I t/m IV respectively. 5 and 6 correspond to CEMT Va and Vb respectively. 7, 8 and 9 correspond to CEMT VIa, VIb and VIc respectively. 10 corresponds with the added nodes.
J	Oppervlakte vaarwegklasse	Max area a ship of a certain CEMT-class can have.
K	Oppervlakte sluis	Area of the lock. Based on length and width.
L	Aantal schepen per schut	Theoretical amount of ships that can pass a lock. Based on the areas of the ships and locks.
M	Max laadvermogen	Max amount of payload of the max CEMT-class.
N/O/P/Q	$0, x$ Aandeel max-n	Share of a certain CEMT-class which passes the lock. Based on a share of: 50% maximum CEMT-class which can pass the lock 20% max. -1 CEMT-class ,, 20% max. -2 CEMT-class ,, 10% max. -3 CEMT-class ,, With a minimum possible CEMT-class of II.
R	Gem laadvermogen	Average payload which passes the lock. Based on the assumption of the shares.
S	Schuttijd	Cyclus time for a ship to pass the lock. Based on Passeertijden (2021). Defined as an average of 'uitvaar rood'-'invaar groen' .
T	Capaciteit/uur	The theoretical capacity of amount of ships per hour.
U	Capaciteit laadvermogen	The theoretical capacity of payload per year. [million tons per year]
V	Intensiteit laadvermogen	The theoretical intensity of payload per year.
W	Utilization factor	The calculated utilization factor which is used in the DSM.
X	Lengte	Length of the lock. database (2019)
Y	Breedte	Width of the lock. database (2019)

Table 5: Explanation of columns of sheet: Verbindingen

Column	**Parameter**	**Explanation and source**
A	Source-id	ID-number of the source imported from column B of Table 3.
B	Source naam	Name of the source.
C	Target-id	ID-number of the target imported from column B of Table 3.
D	Target naam	Name of the target.
E	Source klasse	Class of the source imported from column M of Table 3
F	Target klasse	Class of the target imported from column M of Table 3
G	Klasse verbinding	Class of the waterway between the source and the target.
H	Edge klasse	This column represents the lowest class of columns E, F and G and is represented in the DSM.
I	Afstand	Distance between the source and the target in kilometers calculated $\sqrt{(x^2_{target} - x^2_{source}) + (y^2_{target} - y^2_{source})}$ (implemented from column Q and R of Table 3).
J	Bedieningstijd source	Operating times of the source imported from column G of Table 3.
K	Bedieningstijd target	Operating times of the target imported from column G of Table 3.
L	Overeenkomst	In case the operating times correspond this column represents 1 and otherwise 0. This column is used to show the operating time in the DSM.
M	Afstand schaal	The distance calculated in column I is divided into multiple scales that are shown in the DSM.

Datasheet **Aandeel recreatie**: Extracted from map 'Passages' Passsages (2021)
Datasheet **Schuttingen met schip**:Extracted from map 'Schuttingen' Schuttingen (2021)
Datasheet **Schuttijd**: Extracted from map 'Passeertijden' Passeertijden (2021)
Datasheet **Oppervlakte**: Extracted from website RWS

D Code for generating the DSM, implemented in python

```
import math
import plotly.io as pio
import plotly.graph_objs as go
import ragraph
import ragraph.plot
from ragraph.colors import get_blue, get_orange, get_green, get_red, get_purple
from ragraph.io.csv import from_csv
from ragraph.analysis import cluster
from ragraph.analysis import sequence

# Define the correct path for the nodes and edges
nodes_file_path = "./Data/Database_sluizen_Nederland - Data alle sluizen.csv"
edges_file_path = "./Data/Database_sluizen_Nederland - Alle verbindingen.csv"

# Define weight columns and initialize the graph
weight_columns = ["edge klasse", "afstand schaal", "bedieningstijd gelijk"]
g = from_csv(nodes_path=nodes_file_path, edges_path=edges_file_path, node_weights=["
    Vaarwegklasse", "IC-factor", "Noord/Knoop/Zuid", "Tekstkleur", "Y coordinaat"],
    edge_weights=weight_columns, csv_delimiter= ',')


# I/C FACTOR: red-orange-yellow-light green-green
colscheme_ic=["#ff0000","#ff6900","#ffff00","#90ee90","#008000"]
# Number color on the diagonal: white, blac
colscheme_text_color_diagonal=['#ffffff', '#000000']
# OPERATIONAL TIMES: green-red
colscheme_optime=['#008000','#ff0000']
# DISTANCE:
colscheme_distance=["#008000","#90ee90","#ffff00","#ff6900","#ff0000"]
# CEMT_CLASSES: same shades used as in the network map of all locks
colscheme_cemt=["#e2efd9","#c5e0b3","#a8d08d","#538135","#375623","#c55a11","#833c0b","
    #9cc3e5","#2e75b5","#1e4e79"]
# Horizontal white and grey lines
grey_colors = ['#FFFFFF', '#FFFFFF', '#FFFFFF', '#D3D3D3', '#D3D3D3', '#D3D3D3']
# Horizontal blue and light blue blocks for the bus
blue_colors = ["#DAF3FB", "#DAF3FB", "#DAF3FB", "#add8e6", "#add8e6", "#add8e6"]

# Define parameters for the graph
fig= ragraph.plot.mdm(
    leafs = g.leafs,
    edges = g.edges,
    style = ragraph.plot.Style(
        piemap=dict(
            display="weights",
            radius = 0.4,
        ),
        palettes=dict(
        fields={
            "bedieningstijd gelijk": colscheme_optime,
            "afstand schaal": colscheme_distance,
            "edge klasse": colscheme_cemt,
        }
        )
    ),
    # Make sure the order of the nodes is unchanged
    sort = False
)

## Put the CEMT class on the diagnonal for all nodes
n_nodes = len([n for n in g.nodes if n.is_leaf])
x = [idx+0.5 for idx in range(n_nodes)]
y = [n_nodes-0.5-idy for idy in range(n_nodes)]
colors = ragraph.colors.get_green(n_nodes)
## Create and add trace for numbers on diagonal.
text_trace = go.Scatter(
    x = x,
    y = y,
    text = [str(int(idx.weights["Vaarwegklasse"])) for idx in g.leafs],
    mode="text",
    showlegend=False,
    textfont=dict(
```

```
        color= [colscheme_text_color_diagonal[int(idx.weights["Tekstkleur"])] for idx in
     g.leafs])
)

# Create a list for all extra shapes
shapes = []

# Create the grey horizontal lines
for idx, color in enumerate(g.leafs):
    shapes.append(
        go.layout.Shape(
            type="rect",
            x0=0,
            y0=y[idx]+.5,
            x1=146,
            y1=y[idx]-.5,
            fillcolor=grey_colors[idx%6],
            xref="x9",
            yref="y9",
            layer='below'
        )
    )

# Create the blue horizontal line for the buss (rows 64, 65 and 66)
# With 146 rows this gives us the coordinates 146 - 66 = 80 and 146 - 63 = 83
shapes.append(
    go.layout.Shape(
        type="rect",
        x0=0, y0=80, x1=146, y1=83,
        fillcolor = "#add8e6",
        xref="x9", yref="y9",
        layer='below',
    )
)

# Create the light blue horizontal line for the buss (rows 62 and 63)
# With 146 rows this gives us the coordinates 146 - 63 = 83 and 146 - 61 = 85
shapes.append(
    go.layout.Shape(
        type="rect",
        x0=0, y0=83, x1=146, y1=85,
        fillcolor = "#DAF3FB",
        xref="x9", yref="y9",
        layer='below',
    )
)

# Create the vertical blue/light blue line for the buss (columns 62, 63, 64, 65 and 66)
for idx, color in enumerate(g.leafs):
    shapes.append(
        go.layout.Shape(
            type="rect",
            x0=61, y0=y[idx]+.5, x1=66, y1=y[idx]-.5,
            fillcolor=blue_colors[idx%6],
            xref="x9", yref="y9",
            layer='below'
        )
    )

# Create colored squares on the diagonal to display the utilization of the node
for idx, node in enumerate(g.leafs):
    if int(round((node.weights["IC-factor"]))) < 0:
        color = '#FF0000'
    elif node.weights["IC-factor"] == 0:
        color = '#0000FF'
    elif node.weights["IC-factor"] != 1:
        color = colscheme_ic[math.ceil(node.weights["IC-factor"]*5)-1]
    else:
        color = "#D3D3D3"
    shapes.append(
        go.layout.Shape(
            type="rect",
            x0=x[idx]-.5,
```

```
            y0=y[idx]+.5,
            x1=x[idx]+.5,
            y1=y[idx]-.5,
            fillcolor = color,
            xref="x9",
            yref="y9",
            layer='below'
        )
    )

fig = fig.update_layout(
    shapes = list(fig.layout.shapes) + shapes
)

fig = fig.add_trace(text_trace, row=2, col=4)

pio.write_image(fig, 'DSM.pdf')        # Save image as pdf (hogere resolutie dan
    downloaden)
```

E Dijkstra's pathfinding algorithm, implementation in python

```
# ######## Python program for Dijkstra's
# single source shortest path algorithm. The program
# is for adjacency matrix representation of the graph
#
# first data is extracted from csv files and converted
# into the matrix representation format.
# then the shortest possible path is calculated.

## inputs:
# class of ship for desired route
# starting node (lock).

## outputs:
# possible routes, distance, path.
## impossible routes return a distance of 1e6

from collections import defaultdict
import pandas as pd

##########################################
# converting csv to matrix representation
##########################################

nodes_file_path = "./Database_sluizen_Nederland_Data_alle_sluizen_2.csv"
edges_file_path = "./Database_sluizen_Nederland_Alle_verbindingen.csv"

edges_data = pd.read_csv(edges_file_path,sep = ",")
nodes_data = pd.read_csv(nodes_file_path,sep = ",")

print("Overview of nodes names and corresponding index: \n")
print("    Lockname")
for i in range(len(nodes_data)):
    print(i, " ", nodes_data.at[i,'name'])

#graph with connections and distances.
graph = [ [] for _ in range(len(nodes_data))]
#graph with CEMT classes of connections
CEMT = [ [] for _ in range(len(nodes_data))]

# fill graphs with zeros for non-connections.
# n x n matrix with n the number of nodes in the network
for i in range(0,len(nodes_data)):
    for j in range(0,len(nodes_data)):
        graph[i].append(0)
        CEMT[i].append(0)

# loop over all nodes to find distances and classes of all connections.
for i in range(0,len(nodes_data)):
    #first some empty lists to store data from csv files
    targets = []
    distances = []
    classes = []

    # loop over data of all the edge connections.
    for j in range(0,len(edges_data)):
        # if source sluis name == current node[i]
        # aka find lines of data with connections for this lock
        if edges_data.loc[j,'source'] == nodes_data.loc[i,'name']:
            # add data of the targets lock, distance and cemt class.
            targets.append(edges_data.loc[j,'target'])
            distances.append(edges_data.loc[j,'afstand'])
            classes.append(edges_data.loc[j,'edge klasse'])

    #store data from targets. distances and classes at correct pos in matrix.
    for k in range(0,len(targets)):
        #find correct index in nodes list
        index_target = nodes_data[nodes_data['name'] == targets[k]].index[0]
        #store distances and classes data in graph en CEMT array.
        graph[i][index_target] = distances[k]
        CEMT[i][index_target] = classes[k]

```

```
72 ###################################
73 # constructing dijkstra algorithm
74 ###################################
75
76 #Class to represent a graph
77 class Graph:
78     # A utility function to find the vertex with minimum dist value,
79     # from the set of vertices still in queue
80
81     def minDistance(self,dist,queue):
82         # Initialize min value and min_index as -1
83         minimum = float("Inf")
84         min_index = -1
85
86         # from the dist array,pick one which
87         # has min value and is till in queue
88         for i in range(len(dist)):
89             if dist[i] < minimum and i in queue:
90                 minimum = dist[i]
91                 min_index = i
92
93         return min_index
94
95
96     # Function to print shortest path
97     # from source to j using parent array
98     def printPath(self, parent, j,nodes_data):
99         name_lock = nodes_data.loc[j,'name']
100        #Base Case : If j is source
101        if parent[j] == -1 :
102            print(j, name_lock, end = ", ")
103            return
104        self.printPath(parent , parent[j],nodes_data)
105        print(j, name_lock, end = ", "),
106
107
108    # A utility function to print
109    # the constructed distance array
110    def printSolution(self, dist, parent, src, nodes_data,ship_class, end_node, show_all
    ):
111        #src = 0
112        print("Solution with CEMT class:", ship_class)
113        print("Vertex \t\t\t\tDistance from Source\t\t\tPath")
114
115        if show_all == 0:
116            name_source_lock = nodes_data.loc[src,'name']
117            name_target_lock = nodes_data.loc[end_node,'name']
118            print("\n%d %s --> %d %s \t\t%d \t\t" % (src, name_source_lock, end_node,
    name_target_lock, dist[end_node]), end = " "),
119            self.printPath(parent,end_node,nodes_data)
120
121        else:
122            for i in range(0, len(dist)):
123                name_source_lock = nodes_data.loc[src,'name']
124                name_target_lock = nodes_data.loc[i,'name']
125                print("\n %d %s --> %d %s \t\t\t%d \t\t\t" % (src, name_source_lock, i,
    name_target_lock, dist[i]), end = " "),
126                self.printPath(parent,i,nodes_data)
127
128
129    '''Function that implements Dijkstra's single source shortest path
130    algorithm for a graph represented using adjacency matrix
131    representation'''
132    def dijkstra(self, graph, CEMT_graph, class_ship, src, end_node, nodes_data,
    show_all):
133
134        row = len(graph)
135        col = len(graph[0])
136
137        # The output array. dist[i] will hold the shortest distance
138        # from src to i Initialize all distances as INFINITE
139        dist = [float("Inf")] * row
140
```

```
        #Parent array to store
        # shortest path tree
        parent = [-1] * row

        # Distance of source vertex
        # from itself is always 0
        dist[src] = 0

        # Add all vertices in queue
        queue = []
        for i in range(row):
            queue.append(i)

        #Find shortest path for all vertices
        while queue:

            # Pick the minimum dist vertex
            # from the set of vertices still in queue
            u = self.minDistance(dist,queue)

            # remove min element
            queue.remove(u)

            # Update dist value and parent
            # index of the adjacent vertices of
            # the picked vertex. Consider only
            # those vertices which are still in
            # queue
            for i in range(col):
                '''Update dist[i] only if it is in queue, there is
                an edge from u to i, and total weight of path from
                src to i through u is smaller than current value of
                dist[i]'''
                if (graph[u][i]) and (i in queue):
                    if ((dist[u] + graph[u][i] < dist[i]) and (class_ship<= CEMT_graph[u
    ][i])):
                        dist[i] = dist[u] + graph[u][i]
                        parent[i] = u
                    elif ((dist[u] + graph[u][i] < dist[i])):
                        dist[i] = float("999999")

         # print the constructed distance array
        self.printSolution(dist,parent,src,nodes_data,class_ship, end_node, show_all)

############################################################
# use above graph to calculate values
############################################################
g = Graph()

class_ship = int(input("\n enter CEMT class of ship: (I = 1, II = 2, III = 3, IV = 4, Va
     = 5, Vb = 6, VIa = 7, VIb = 8, VIc = 9)"))
print("enter starting node:")
start_node = int(input())#62
print("enter end node:")
end_node = int(input())#50
print("show all (1) or not (0):")
show_all = int(input())#0
print("enter unavailable nodes with spaces: x y z")

########
input_unav = input()
print("\n")
unavailable_nodes = input_unav.split()

# convert each item to int type
for i in range(len(unavailable_nodes)):
    # convert each item to int type
    unavailable_nodes[i] = int(unavailable_nodes[i])
##########



for i in range(0,len(nodes_data)):
```

```
212     for j in range(0,len(unavailable_nodes)):
213         CEMT[i][unavailable_nodes[j]] = -1
214 
215 print("")
216 # Print the solution
217 # inputs: matrix_with_connections, matrix_with_CEMT_classes, starting_node, end_node
        list_with_node_names, show_all_solutions
218 g.dijkstra(graph, CEMT, class_ship, start_node,end_node, nodes_data, show_all)
219 
220 ########################################################
221 # Part of this code is contributed by Neelam Yadav
222 # Daan den Hartog, 12/01/2022 - TU Eindhoven
223 ########################################################
```