Institut für Intermodale Transport- und Logistiksysteme

Technische Universität Braunschweig

Multimodal Transport Systems (Multimodale Transportsysteme)

Lecture Notes

Jürgen Pannek

April 8, 2024

Jürgen Pannek
Institute for Intermodal Transport and Logistic Systems
Hermann-Blenck-Str. 42
38519 Braunschweig

FOREWORD

During summer term 2024 I give the lecture to the module *Multimodal Transport Systems (Multimodale Transportsysteme)* at the Technical University of Braunschweig. To structure the lecture and support my students in their learning process, I prepared these lecture notes. As it is the first edition, the notes are still incomplete and are updated in due course of the lecture itself. Moreover, I will integrate remarks and corrections throughout the term.
The aim of the module is to provide an overview on intermodal transport and logistics systems with a particular focus on methods for planning, design and coordination of such systems.
In particular, students shall be able to describe, explain, apply and analyze modes and systems in transport and logistics. Moreover, students can recall, interpret and evaluate key performance indicators for unimodal and intermodal systems. Regarding planning and design, students are able to characterize, apply and differentiate methods with respect to the area of application and assess suitability of these methods. Last, students are able to describe, categorize and evaluate methods of coordination regarding intermodality.
To this end, we address the subject areas

- modes and systems in transport and logistics,
- design and planning of systems, and
- methods of coordination

within the lecture. Additionally, we utilize simulation and hardware to support understanding and application of the discussed methods within the tutorial classes. The module itself is accredited with 5 credits.
An electronic version of this script can be found at

https://www.tu-braunschweig.de/en/itl/teaching/lecture-notes

During the preparation of the lecture, I utilized the book of Gudehus [4] and Neumann [7].

Literature for further reading

- Modes and systems in transport and logistics
  - GUDEHUS, T.: *Logistik: Grundlagen, Strategien, Anwendungen*. 4th edt. Springer, 2010
  - GUDEHUS, T. ; KOTZAB, H.: *Comprehensive Logistics*. Springer, 2012
- Design and planning of systems
  - SARDER, M.D.: *Logistics Transportation Systems*. Elsevier, 2020
  - NEUMANN, K. ; MORLOCK, M.: *Operations Research*. 2nd edt. Hanser, 2004
  - FEICHTINGER, G. ; HARTL, R.F.: *Optimale Kontrolle ökonomischer Prozesse*. deGruyter, 2011
- Methods of coordination
  - VOGT, J.J.: *Business Logistics Management*. 5th edt. Oxford University Press, 2016
  - SCHÖNBERGER, J.: *Model-Based Control of Logistics Processes in Volatile Environments*. Springer, 2011

Contents

List of Tables

List of Figures

List of Definitions and Theorems

LIST OF ALGORITHMS

CHAPTER 1

TRANSPORT AND LOGISTICS SYSTEMS

> Amateurs discuss tactics, professionals discuss logistics.
>
> Napoleon Bonaparte

While modern transport and logistics systems are not primarily driven by military developments but instead by civil needs of people and companies, the tasks, utilities, approaches and methods of such systems remain identical. Within this lecture, we discuss modes and transport and logistics systems with a particular focus on how to intertwine modes. Here, we not only consider different modes as in road, rail, water etc. transport, but also modes of operation such as cooperation and non-cooperation.

Within the present chapter, we introduce the basic terms which we are going to use throughout the lecture. Thereafter, we continue by defining the general concept of a system before stating standard methods to describe such systems in models. Knowing the description of a transport system, we discuss performances measures.

1.1 Modes

The task of logistics systems is provide the right amount of required objects in the right composition at the right location at the right time. In order to fulfill this task, three basic utilities are required: infrastructure, vehicles and operation. Within this section, we introduce basic terms in a descriptive manner. Based on these, in the following section we formalize the latter to utilize computational methods for solving the logistics task.

In order to arrive at a logistics system, we need to go along the task definition of such systems and introduce respective terms. Based on DIN IEC 60050-351 [2], we start by the terms of a system, a process and an object:

Definition 1.1 (System).
A *system* is a set of interrelated elements that are viewed as a whole in a particular context and considered as distinct from their environment.

Building on the description of a system, a process is given as follows

Definition 1.2 (Process).
A *process* is the entirety of relations and interacting elements in a system through which matter, energy or information is transformed, transported or stored.

In logistics, the object to be transformed, transported or stored is defined more precisely in DIN 30781 [1]:

Definition 1.3 (Object).
A *logistics object* may consist of people, goods, energy or information.

We like to stress that an logistics object is not restricted to be of the types mentioned in Definition 1.3 but, e.g., be a combination of good and information. Moreover, while objects may be time dependent, e.g. degradable such as fruits, time itself is not a logistics object. In real life, objects are humans, trading goods, food, raw materials or material, pre-products, semi-finished products, products as well as investment and consumer goods and production and operating resources. Additionally, in reverse logistics also waste and exhausted goods are objects.
Continuing with the task of logistics systems, we next define composition. To this end, we require an auxiliary, which is fundamental in any kind of logistics system.

Definition 1.4 (Load carrier and load unit).
A *load carrier* is a bearing means to accumulate objects to one load unit. A *load unit* consists of one or multiple objects combined by a load carrier together with necessary *safety agents*.

Remark 1.5
Note that even for single objects such as humans, load carriers such as seats or tethers are provided which in that case also serve as safety agents.

[1]Source: https://commons.wikimedia.org/wiki/File:EPAL-Europalette.jpg
[2]Source: https://www.europlanttray.com/de/

(a) Euro pallet[1]

(b) Plant tray[2]

Figure 1.1: Examples of load carriers

While load size one is common in transport of people, the standard case for transporting goods is to accumulate several objects. Following Definition 1.2 of a process, we clarify the terms „transform“, „transport“ and „store“.

Definition 1.6 (De-/commissioning).
Commissioning/decommissioning is the process of assembling and disassembling load units.

Having defined an object, we can introduce the meaning of transport.

Definition 1.7 (Transport).
A *transport* is the possibly time dependent process of moving of a load unit starting at an initial time at an initial location to a target location at a target time.

At this point we observe that time enters into our description. For one, the displacement starts and ends at certain time and certain places like, e.g., the start and end points of a trip, but also the displacement may be time dependent, e.g., caused by traffic. Still we require object to be available at the right time. Therefore, starting/arriving to early may render the task of a transport and logistics system to be infeasible. Hence, storage is required as an option to bridge time gaps.

Definition 1.8 (Storage).
Storage is the time displacement of a load unit in an unchanging location.

Last, it may be necessary to rearrange load units between two transports or between transport and storage.

Definition 1.9 (Handling).
The consecutive decommissioning and commissioning is called *handling*.

(a) Automated storage facility[3] (b) Automated handling facility[4]

Figure 1.2: Examples of storage and handling processes

Remark 1.10
Note that the resulting number of load units may change by handling.

Now we can combine the definitions above and formally introduce a logistics system.

Definition 1.11 (Logistics).
Logistics refers to the combination of the processes transport, de-/commissioning, handling and storage of objects.

Each of the four logistics processes may be executed in different environments. Stemming from transport, these are referred to as modes. Depending on technology, the list of modes may be extended in the future. Currently, the following modes are known:

Definition 1.12 (Modes of logistics processes).
The *modes* of the logistics processes

- for transport and storage include air, land (rail and road), water, cable, pipeline, space and cyberspace, and
- for de-/commissioning and handling include manual, automated, and machine supported.

[3]Source: https://commons.wikimedia.org/wiki/File:TGW-Stingray-Shuttle.jpg
[4]Source: https://commons.wikimedia.org/wiki/File:Float_Glass_Unloading.jpg

Additionally, for logistics process three core components can be identified: For one, passive components provide and consume objects, active components modify time, space and consistency of objects, and process components determine the sequence and composition of passive and active components. To introduce these components, we require the following:

Definition 1.13 (Sinks, sources and nodes).
A point in space is called *source* if it generate objects, *sink* if it consumes objects, and *node* if it may hold objects.

Now, the norm DIN 30781 [1] provides us with respective terms of the components.

Definition 1.14 (Components of logistics processes).
We call

- sinks, sources and nodes of objects *infrastructure*,
- time, space and consistency changes to objects *utilities*, and
- the method to define paths from sinks to sources *operations*.

The terms infrastructure, utilities and operations are called components of logistics systems.

In real life, sources are all kinds of warehouses, manufacturing plants and workshops. Sinks correspond to consumers, shops and markets. Hence, by conservation of energy/material, we directly see the following:

Corollary 1.15 (Identity of sources/sinks).
Sources are always sinks in other logistics systems.

Note that we do not use operations as paths from sources to sinks, but methods to determine such paths. Within DIN 30781 [1], the term transport chain is used for such paths. The term itself is technically misleading as it also includes de-/commissioning, handling and storage:

Definition 1.16 (Transport chain or path).
A *transport chain* or *path* is a sequence of logistics processes.

Based on the latter, we can already derive a necessary condition for a transport and logistics system:

Corollary 1.17 (Necessary components).
Any transport and logistics system requires infrastructure, utilities, and operations.

Considering a specific transport, it is possible to use different modes. These modes may exist in parallel or may be used in sequence. In the parallel case there is a choice between these modes and the object may be displaced using any of these modes. This case is referred to as multimodal:

Definition 1.18 (Multimodality).
Consider an object, for which there exist several transport options with different modes. Then the transport is called *multimodal*. Moreover, any transport and logistics system is called *multimodal* if there exists at least one multimodal transport.

The multimodal case is quite common in transport of humans, which may choose to walk, use a bike or car or public transport. Hence, multimodality provides us with options. In contrast to that, the sequential case means that an object is transported using different modes (and possible intermediate handlings).

Definition 1.19 (Intermodality).
We call a transport chain *intermodal* if more than one transport mode is used.

The typical example of an intermodal transport chain is transportation by ship, rail or aircraft. In any of these cases, a transport chain can be split in three (or more) parts: a preliminary leg, a main leg or main run, and a subsequent leg. In a preliminary leg the respective object is first transported to utilities (ports, railway stations, airports etc.), where they are handled to switch transport mode for the main run. Arriving at end point of the main run, the object is again handled and then fed to the subsequent leg.

Remark 1.20
We like to stress that intermodality refers to transport chains, whereas multimodality refers to a transport, cf. Figure 1.3

Note that the multimodal example in transport of humans is not completely correct, i.e. before taking public transport, one typically has to walk to a respective station. So technically the chosen transport itself is an intermodal one.
Having introduced the basic terms we are dealing with, our next aim is to formalize these terms to use them in an automated way such as, e.g., simulation.

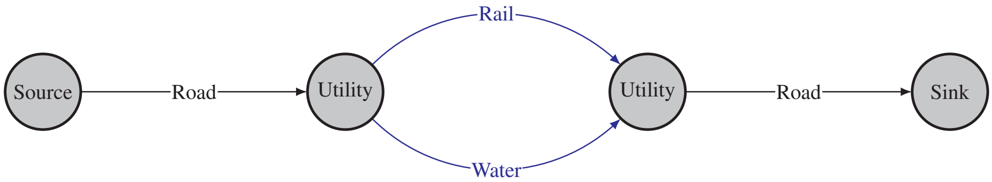


Figure 1.3: Multimodal transport (blue) in an intermodal transport chain

1.2 Modeling

Within modeling, there are several sources which can lead to misunderstandings. These range from industrial sector specific terms or objects to coding languages up to cultural diversity. Yet the aim is always to provide insight into the system at hand. Hence, the overall tasks to be included in concepts of modeling is to provide a holistic view of the information model. For transport and logistics systems, these views may differ depending on their purpose. For example, for real time planning a functional description using differential equation is more appropriate than the graph description used for planning and vice versa.
Modeling itself as cognition method exhibits the components shown in Figure 1.4.

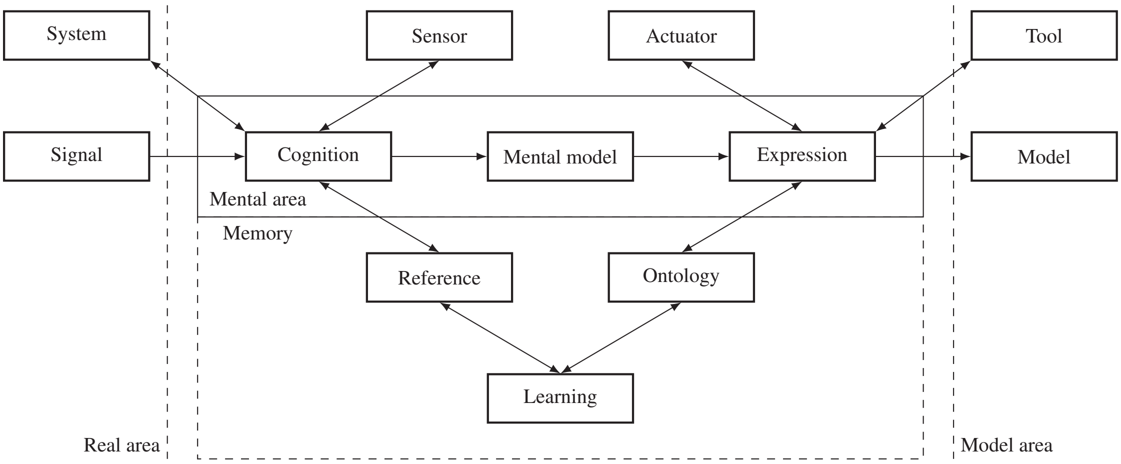
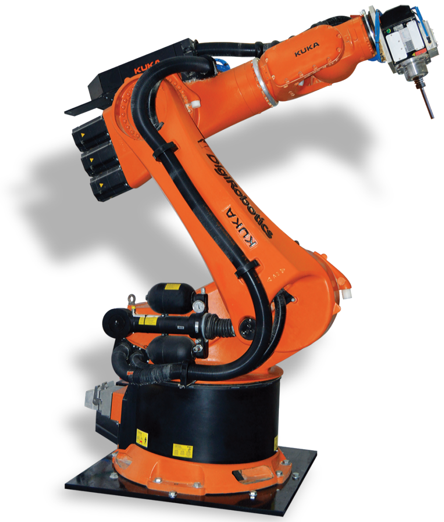


Figure 1.4: Modeling as cognition method

Here, the model can exist on various levels and degrees of detail. An example is given by the robot arm in Figure 1.5, which can be abstracted to its main physical components.
However, these components can be further detailed to satisfy a set of differential equations, which need to be parametrized and identified. The level of detail therefore depends on the purpose of the model and how it shall be used.
Regardless of the usage, there exist process requirements for modeling, that is working principles found within any modeling process. These requirements are a state-of-the-art list, which is



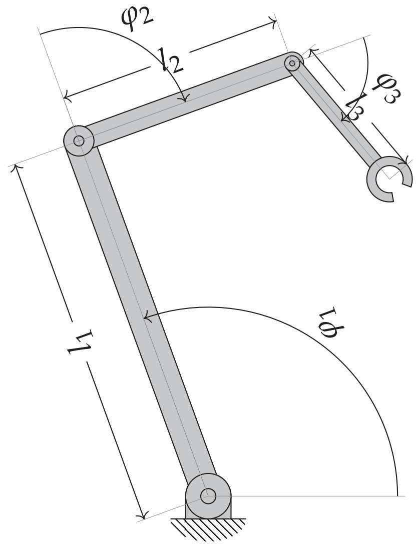


Figure 1.5: Robot arm in real and model area

commonly used as a convention and not as a definition.

Convention 1.21 (Process requirements of modeling)
During the modeling process, six principles need to be met:

1. Principle of Correctness: A model needs to present the facts correctly regarding structure and dynamics (semantics). Specific notation rules have to be considered (syntax).

2. Principle of Relevance: All relevant items have to be modeled. Non-relevant items have to be left out, i.e. the value of the model doesn't decline if these items are removed.

3. Principle of Cost vs. Benefit: The amount of effort to gather the data and produce the model must be balanced against the expected benefit.

4. Principle of Clarity: The model must be understandable and usable. The required knowledge for understanding the model should be as low as possible.

5. Principle of Comparability: A common approach to modeling ensures future comparability of different models that have been created independently from each other.

6. Principle of Systematic Structure: Models produced in different views should be capable of integration. Interfaces need to be designed to ensure interoperability.

Here, we like to stress that the principle of cost vs. benefit already dictates that it does not make sense to model the entirety of any system. Instead, we have to accept that the model is to some (hopefully known) extend not perfect. Figure 1.6 illustrates this idea.

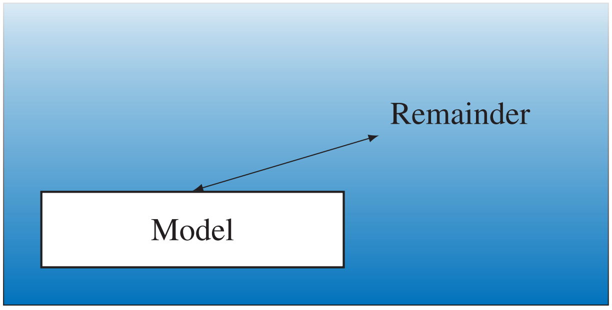


Figure 1.6: Model and remainder

Similarly, any concept of modeling should satisfy the following functional requirements:

Convention 1.22 (Functional requirements of modeling)
For any concept of modeling, the fitness of methods, tools and implementation need to be aligned.

1. Fitness for methods:
   - Consideration of modern software development methods
   - Consideration of development phase
   - Analytical/mathematical properties
   - Theoretical soundness and provability
   - Vertical and horizontal consistency
   - Composability and decomposability
   - Consideration of deterministic and stochastic properties
   - Graphical presentability
   - Ability for simulation
   - Testability, traceability and comprehensibility
2. Fitness for tools
   - Portability
   - Compatibility
   - Usability
3. Fitness for implementation

- Viability in soft- and hardware
- Reverse engineering

Depending on the purpose of the description, a variety of methods are applied. In the following Table 1.1, we structure these methods according to usage. The sequence follows a top down idea:

Table 1.1: List of description methods

Class	**Example**
Abstraction oriented	Verbal description, algebra, proposition logic, predicate logic
Structure oriented	Sequential logic system, combinatorial logic
Implementation oriented	Logic plan, function plan, contact diagram, structure diagram, timing diagram, instruction list, gantt chart
State oriented	Decision table, transition table, state diagram, state graph, Karnaugh-Veitch diagram
Technology oriented	Flow chart, switching plan, computer aided design (CAD)
Method oriented	Network diagram, Nassi-Shneidermann diagram, unified modeling language (UML), structure-analysis-real-time (SA/RT) diagram
Decision/Time oriented	Boolean algebra, differential/difference equations, Markov chains

Note that the methods cannot be considered to stand by themselves, they require specification top down and connection bottum up.
Within this lecture, we focus on methods and decision making. Therefore, we utilize networks and equation systems for our models. Both approaches are suitable for modern computer science concepts such as object orientation, allow for mathematical concepts such as discrete event triggered dynamical systems and integrate both compact description as well as horizontal and hierarchical structures. Generally speaking, the (mathematical) description of models varies depending on the considered time, space and amplitude properties. Figure 1.7 provides a rough overview on these characteristics.

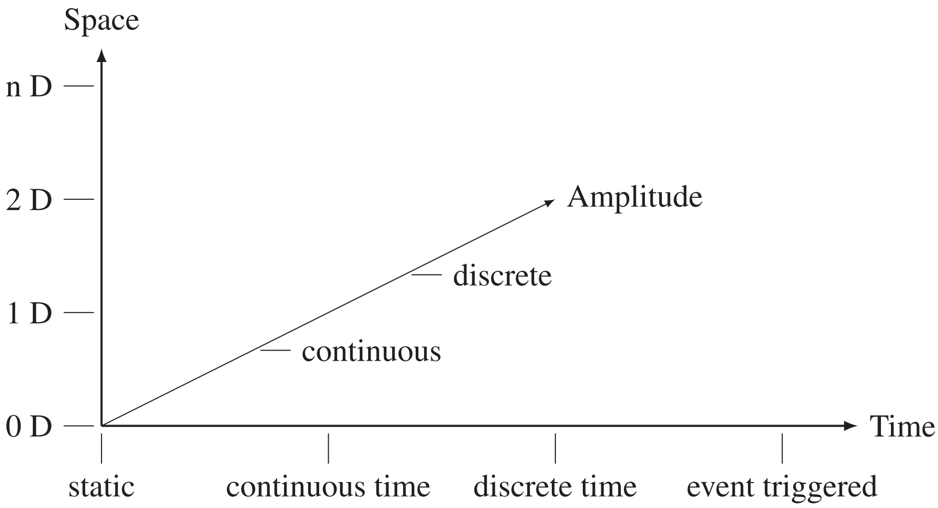


Figure 1.7: Dimensions of model characteristics

Remark 1.23
Regarding time, static models are characterized by the fact that inputs, outputs, and measurements of the system are available. In contrast to that, continuous time models exhibit data streams being received continuously. Discrete-time models differ from that by the availability of data, which is received at certain, not necessarily equidistant time instances. Last, event-triggered models require issues to trigger receiving data.
Regarding space, models may vary from a simple connection to complex systems.
Regarding amplitude, models may differ regarding continuous spaces e.g., mass, and discrete spaces such as gear shifts.

Following Table 1.1 on networks, we more formally define the following:

Definition 1.24 (Network).
Consider a set of $\mathcal{V} = \{v_1, \ldots v_{n_\mathcal{V}}\}$ where $n_\mathcal{V} \in \mathbb{N}$ is the maximal entry of $\mathcal{V}$. Moreover, suppose $\mathcal{E} = \mathcal{V} \times \mathcal{V}$ where $n_\mathcal{E} \in \mathbb{N}$ is the maximal entry of $\mathcal{E}$. Then we call $\mathcal{V}$ the *set of vertexes*, $\mathcal{E}$ the *set of edges* connecting the vertexes, and $\mathcal{N} = (\mathcal{V}, \mathcal{E})$ a *network*.

Regarding our transport and logistics processes (Definition 1.14), we can directly identify the network and logistics components:

Corollary 1.25 (Identification of components and network).
For logistics processes, we identify definable infrastructure as vertexes and definable utilities as edges.

Note that Corollary 1.25 does not require all components to be defined. This is in accordance with our concept of a model, for which certain parts are modeled and the remainder is considered as disturbance.

Remark 1.26

In the computer science or mathematics literature, a network is also called a graph, for which the set of vertexes is typically referred to as set of nodes.

In the process automation literature, vertexes are split into attributes and methods. Attributes are also called places indicating the physical position of an object. Methods, on the other hand, are also called transitions, i.e. transportation from start to destination or modifications from initial to target property.

Task 1.27

Consider the transportation system given by $\mathcal{V} = \{A, B, C, D, E, F\}$*, which is complete. Draw the respective network.*

Solution to Task 1.27: Within a complete network for each pair of vertexes there exists an edge. The network is displayed in Figure 1.8.

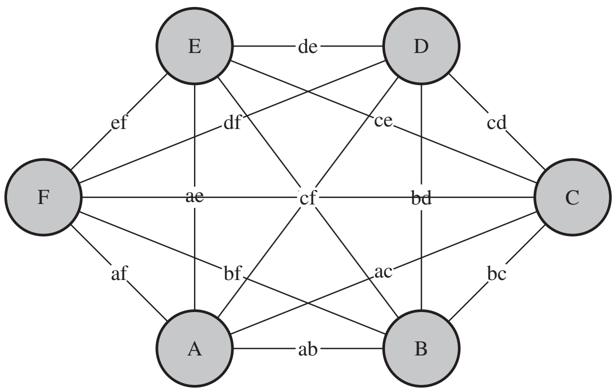


Figure 1.8: Graph of network from Task 1.27

Within these networks, there are value streams between vertexes. Coming back to the special vertexes sources and sinks, cf. Definition 1.13, there is a feed entering the network at the sources and leaving the network at the sinks. More generally, we define the input/output and value stream using the following:

Definition 1.28 (System and process).
Consider two sets $\mathcal{U}$ and $\mathcal{Y}$. Then a map $\Sigma : \mathcal{U} \to \mathcal{Y}$ is called a *system* and the application of this map to an input $\mathbf{u} \in \mathcal{U}$ to obtain an output $\mathbf{y} = \Sigma(\mathbf{u}) \in \mathcal{Y}$ is called a *process*.

The latter definition will be very useful on both the planning / tactical level as well as the control / operational level. In particular, the sets $\mathcal{U}$ and $\mathcal{Y}$ are called input and output sets. An element from the input set $\mathbf{u} \in \mathcal{U}$ is called an input, which acts from the environment to the system and is not dependent on the system itself or its properties, cf. Figure 1.9.

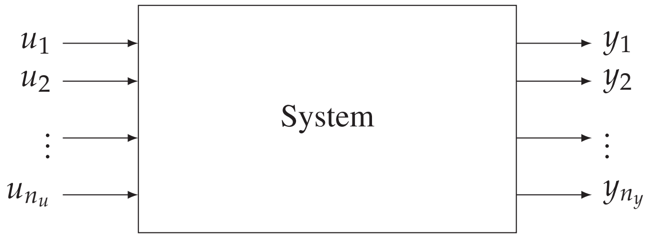


Figure 1.9: Term of a system

We distinguish between inputs, which are used to specifically manipulate (or control) the system, and inputs, which are not manipulated on purpose. We call the first ones *control or manipulation inputs*, and we refer to the second ones as *disturbance inputs*. An element from the output set $\mathbf{y} \in \mathcal{Y}$ is called an output. In contrast to an input, the output is generated by the system and influences the environment.
In practice, we subdivide between three different networks:

Definition 1.29 (Intralog, extralog and interlog).
Consider a transport and logistics system to be modeled by a network. Then we define

- *intralog network* as a system within one infrastructure of one company,
- *extralog network* as a system between the infrastructure of one company, and
- *interlog network* as a system of all companies.

Table 1.2: Atrributes of logistics systems

	Intralog	Extralog	Interlog
Scope	System within one infrastructure	System between the infrastructure of one company	System between all infrastructures
Site	One	Multiple	Multiple
Networking	Low	Medium	High
Paths	Internal	Cross infrastructure	Cross company
Source	Stock receipt Manufacturing plant	Supplier Other sites	Companies Households
Sink	Goods issue Point of consumption	Customer Other sites	Companies Households
Configuration	Machine system Storage system Commissioning system Conveying system	Procurement Distribution Disposal Transport	Intralog Extralog Transport

Algebraically, the network or graph resulting from using the network notion stated above can be summarized in the so called incidence matrix.

Definition 1.30 (Incidence matrix).
For any network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$, we call

$$\mathcal{I} = [\mathcal{I}_{jk}] \qquad \text{where } \mathcal{I}_{jk} := \begin{cases} 1 & \text{vertex } v_j \text{ is incident with edge } e_k \\ 0 & \text{else} \end{cases} \tag{1.1}$$

incidence matrix of the network.

Hence, the incidence matrix is arranged with vertexes as rows and edges as columns.

Task 1.31
Compute the incidence matrix of the network from Task 1.27.

Solution to Task 1.31: The incidence matrix is given by

$$\mathcal{I} := \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}.$$

In many cases, also the direction of edges is of importance, e.g. if a road or pipeline can only be used in one direction. Then the graph is called directed.

Definition 1.32 (Directed network).
Consider a network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$. If the set of edges $\mathcal{E}$ is ordered, then we call $\mathcal{N}$ a *directed network*. The incidence matrix is defined via

$$\mathcal{I} = [\mathcal{I}_{jk}] \qquad \text{where } \mathcal{I}_{jk} := \begin{cases} -1 & \text{edge } e_k \text{ leaves vertex } v_j \\ 1 & \text{edge } e_k \text{ enters vertex } v_j \\ 0 & \text{else} \end{cases} \tag{1.2}$$

Task 1.33
Consider a complete transport system $\mathcal{V} = \{A, B, C\}$. Draw the network and compute the incidence matrix.

Solution to Task 1.33: For the network displayed in Figure 1.10 we obtain

$$\mathcal{I} := \begin{pmatrix} -1 & -1 & 1 & 0 & 1 & 0 \\ 1 & 0 & -1 & -1 & 0 & 1 \\ 0 & 1 & 0 & 1 & -1 & -1 \end{pmatrix}$$

as the respective incidence matrix.

The incidence matrix not only provides us with a compact description of the network, it can also be used for computations. It can be used to identify, e.g., whether certain vertexes are necessary/sufficient or may ever be reached. The core tool for such assessments are configurations. In

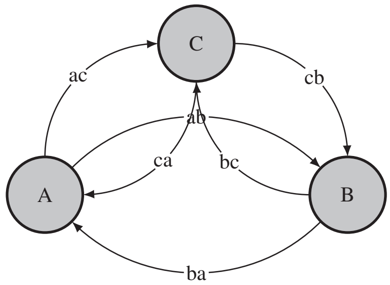


Figure 1.10: Graph of network from Task 1.33

principle, these configuration are nothing else than use cases of a system or process, for which we like to answer certain questions.

Definition 1.34 (Configuration).
Consider a network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$. Then any subset $\mathcal{C} \subseteq \mathcal{V}$ is called a *configuration*. We call the tuple $(\mathcal{N}, \mathcal{C})$ an *elementary network*.

Hence, a configuration is a subnet within a network. As such, it interacts with the rest of the network, yet we are only interested in answers for this specific subset.

Remark 1.35
Loosely speaking, if the entire world would be represented as a network, then a configuration is a model of a process/system, which interacts with its surroundings and is disturbed by it.

We extend the notion of a configuration by introducing markings and multiplicities. Markings can be interpreted as units assigned to a vertex, like load units waiting to be transported. Multiplicities may be used to assign transport costs along an edge.

Definition 1.36 (Marking and multiplicity).
Consider a network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$. Then the maps

$$\mathcal{C}_{\mathcal{V}} : \mathcal{V} \to \mathbb{R}_0^{n_{\mathcal{V}}}, \qquad \mathcal{C}_{\mathcal{V}}(v) = \#\left(v \in \mathcal{C}_{\mathcal{V}}(v)\right) \quad \forall v \in \mathcal{V} \tag{1.3}$$

$$\mathcal{C}_{\mathcal{E}} : \mathcal{E} \to \mathbb{R}_0^{n_{\mathcal{E}}}, \qquad \mathcal{C}_{\mathcal{E}}(e) = \#\left(e \in \mathcal{C}_{\mathcal{E}}(e)\right) \quad \forall e \in \mathcal{E} \tag{1.4}$$

are multisets where $\mathcal{C}_{\mathcal{V}}$ is called *marking* and $\mathcal{C}_{\mathcal{E}}$ is called *multiplicity*. The triple $(\mathcal{N}, \mathcal{C}_{\mathcal{V}}, \mathcal{C}_{\mathcal{E}})$ is called *marked network*.

Remark 1.37
We like to note that multiplicities may also be used to display the required number of units for a utility, i.e. how many load units are required for a specific transport. Such a graph is the more special case of a Petri network.

Within Figure 1.11, multiplicities are added to a network. Similarly, the vertexes can be complemented with values.

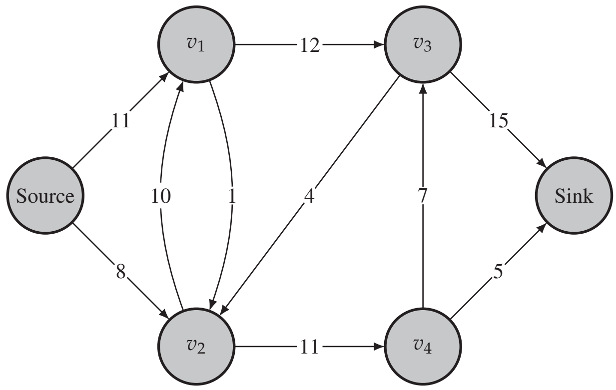


Figure 1.11: Multiplicity within a network

Based on markings and multiplicities, we can introduce bounds of infrastructure and utilities by defining inequalities for the vertexes and edges:

Definition 1.38 (Network constraints).
Consider a network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$. Then we call

$$\underline{v}_j \leq v_j \leq \overline{v}_j \tag{1.5}$$

$$\underline{e}_j \leq e_j \leq \overline{e}_j \tag{1.6}$$

network constraints for all vertexes $v \in \mathcal{V}$ and all edges $e \in \mathcal{E}$.

Figure 1.12 illustrates the network constraints.
Hence, if the constraints are satisfied, we call the respective maps a solution of the network.

Definition 1.39 (Solution of network).
Given a network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$ with constraints. Then we call any $(\mathcal{C}_{\mathcal{V}}, \mathcal{C}_{\mathcal{E}})$ satisfying the constraints a *solution of the network*.

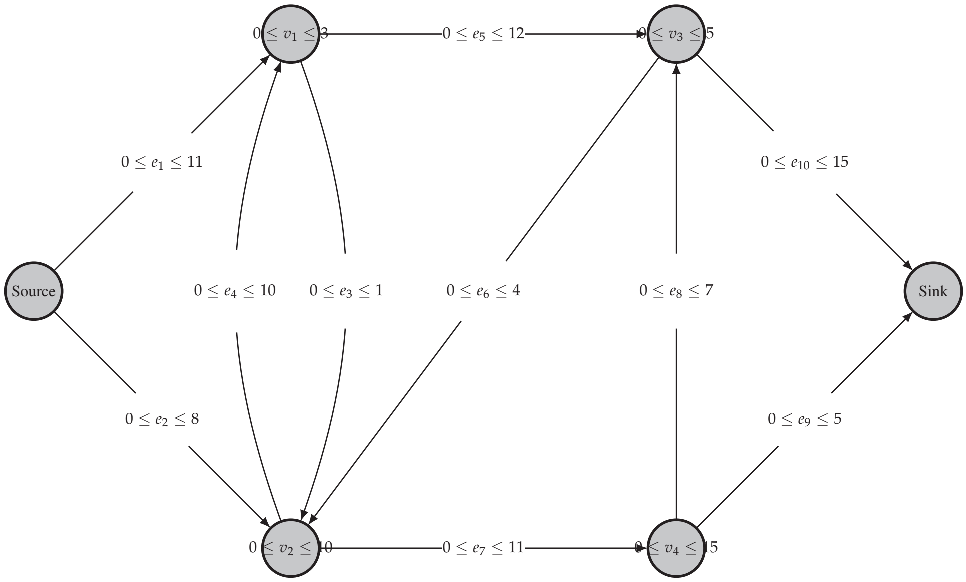


Figure 1.12: Network constraints

At this point, the theory and design of networks splits into two different areas. For one, researchers and practitioners look for answers regarding properties of networks. Some of the most popular questions range from the static ideas of

- reachability: Can all markings be set? Which markings can be set?
- coverability: Can specific markings be set?

to dynamic/time dependent properties such as

- liveness: Is a process/system deadlock free, can all vertexes be marked and unmarked?
- consistency: Will all markings be set uniquely?
- boundedness: Will all markings stay bounded?

In practice, such questions are found in the design phase or on the strategic level of a system, e.g. introduction of new transport modes, a new train station or bus line.
In contrast to topology questions, the second trail deals with optimization regarding – among others – the questions

- minimal cost: Which transport chain shows minimal costs?
- maximal throughput: Which infrastructure or utility provides a bound on the throughput of objects?

- robustness: Is a transport system robust against disturbances/blockages?

Such questions are part of planning on a tactical level. Within this lecture, we include both means in the subsequent chapters.

Remark 1.40
Apart from description via networks, transport and logistics systems can also be modeled using maps from inputs to outputs, that is, e.g., transport requests to changes of load units over time. A basic example is a given transport system where we supply a number of load units to be transported. The system will then show how the load units are transported within the system, e.g., within a simulation.

1.3 Performance indicators

Regarding intent, we are going to use the term a key performance criterion, also called key performance indicator (KPI), throughout the lecture. Formally, the definition follows ISO 22400 [6]:

Definition 1.41 (Key performance criterion).
Given a system/process Σ, a *key performance criterion* is a function $J : \mathcal{Y} \to \mathbb{R}$, which measures defined information retrieved from the system against a standard.

Performance indicators are not to be misunderstood as strategies. A performance indicator is a rating, which allows us to assess aims and strategies. To clarify the difference between KPI and aim, we introduce the following:

Definition 1.42 (Aim).
Consider a system/process Σ and a KPI J to be given. An *aim* is the definition of a desired system/process behavior.

The following Table 1.3 summarizes examples of typical performance measures, which are sorted by strategies. Here, the goals, capabilities and measurements are specified for transport and logistics systems.

Table 1.3: Decomposition and measurement of KPIs

Strategy	Aims	Capability decomposition	Performance measurements
Cost leadership	Productivity	○ Throughput ○ Effectiveness ○ Efficiency	○ Transported units per period ○ Availability, performance ○ Input used for specific output
Differentiation	Agility	○ Response speed ○ On time delivery ○ Fault recovery	○ Response/cycle time ○ Rate to complete and deliver ○ Rate of downtime
	Quality	○ Transport quality ○ Innovation ○ Diversity ○ Service	○ Customer denial/rejection ○ Innovation cycle time ○ Services and personalization ○ Customer's evaluation
	Sustainability	○ Utility/infrastructure ○ Process	○ Energy efficiency, lifetime, reusability ○ Energy use, CO_2 balance

Regarding networks, a cost function depends on the choices of routes along the network only. Therefore, the same notation as for dynamical systems may be used by setting the state set $\mathcal{X} = (\mathcal{C}_{\mathcal{V}}(\mathcal{V}), \mathcal{C}_{\mathcal{E}}(\mathcal{E}))$ and $\mathcal{U} = \emptyset$. For simplicity, we define the following:

Definition 1.43 (Network costs).
We call a key performance criterion given by a function $J : (\mathcal{C}_{\mathcal{V}}(\mathcal{V}), \mathcal{C}_{\mathcal{E}}(\mathcal{E})) \to \mathbb{R}$ *network costs*.

The reference, which is required by Definition 1.41, can be designed by capacity usage of both infrastructure and utilities, or by topology of again both infrastructure and utilities.
Here, we start with the network utilization factor, which provides a reference for the summarized utilization of all logistics components within a network.

Definition 1.44 (Network utilization factor).
Given a network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$ with constraints. Then we call

$$J(\mathcal{C}_{\mathcal{V}}(\mathcal{V}), \mathcal{C}_{\mathcal{E}}(\mathcal{E})) := \underbrace{\frac{1}{n_{\mathcal{V}}} \sum_{j=1}^{n_{\mathcal{V}}} \frac{v_j}{\overline{v}_j}}_{\text{Infrastructure}} + \underbrace{\frac{1}{n_{\mathcal{V}}} \sum_{j=1}^{n_{\mathcal{E}}} \frac{e_j}{\overline{e}_j}}_{\text{Utilities}} \tag{1.7}$$

network utilization factor.

In particular, the utilization factor is an unweighted factor, which looks at percentages of utilization of components. The factor can be split between capacities of infrastructure (storage and handling) and capacities of utilities (transport) and may also be broken down to individual components. One of the outcomes of utilization is to identify the maximal flow or respectively the minimal cut of a network.
Apart from utilization, also topology plays a critical role for transport and logistics systems. The relevant factor regarding topology is the so called detour factor:

Definition 1.45 (Network detour factor).
Consider a network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$. Then we call the ratio

$$J(\mathcal{C}_{\mathcal{V}}(\mathcal{V}), \mathcal{C}_{\mathcal{E}}(\mathcal{E})) := \frac{1}{n_{\mathcal{V}} \cdot (n_{\mathcal{V}} - 1)} \sum_{j=1}^{n_{\mathcal{V}}} \sum_{\substack{k=1 \\ k \neq j}}^{n_{\mathcal{V}}} \frac{d_{\mathcal{N}}(v_j, v_k)}{d(v_j, v_k)} \tag{1.8}$$

network detour factor where $d_{\mathcal{N}}(v_j, v_k)$ is the shortest path and $d(v_j, v_k)$ is the Euclidean distance between the points v_j, v_k

To illustrate the difference between shortest path and Euclidean distance, we utilize the Manhattan norm.

Task 1.46 (Manhattan norm)
Define the Manhattan norm to characterize the distances to be taken if the road network provides a rectangular street grid.

Solution to Task 1.46: Using a rectangular grid, we obtain the shortest path $\|\mathbf{x}\|_1 = \sum_{j=1}^{n_x} |\mathbf{x}_j|$ whereas the Euclidean distance is given by $\|\mathbf{x}\|_2 = \sqrt{\sum_{j=1}^{n_x} \mathbf{x}_j^2}$.

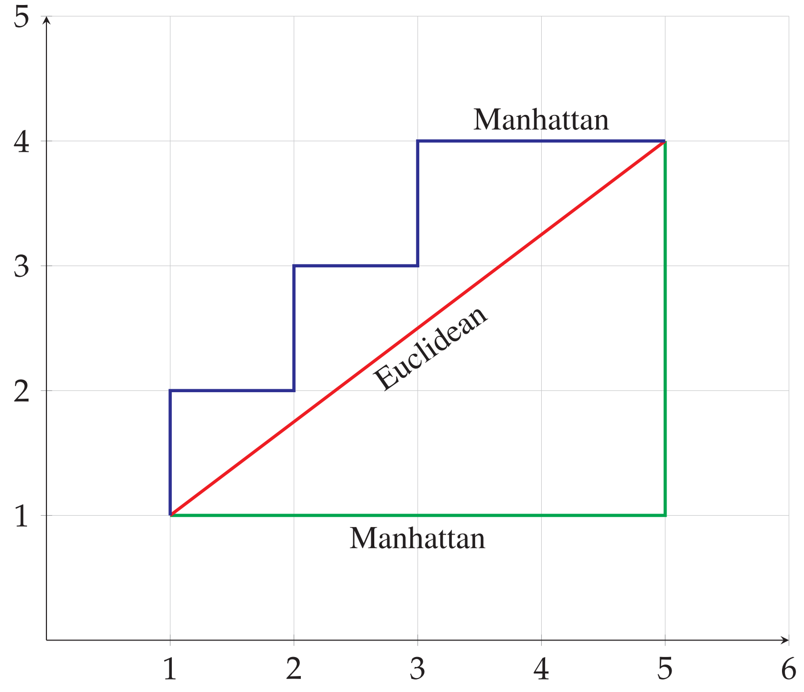


Figure 1.13: Sketch Euclidean and Manhattan norm

An example of the Manhattan distance vs. the Euclidean distance on road networks for the TU Braunschweig is given in Figure 1.14.

As we can directly deduce from the Manhattan norm Task 1.46, the detour factor is limited:

Theorem 1.47 (Limits of detour factor).
Given a network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$. Then the detour factor is bound from below by

$$J(\mathcal{C}_{\mathcal{V}}(\mathcal{V}), \mathcal{C}_{\mathcal{E}}(\mathcal{E})) \geq 1. \tag{1.9}$$

If the network is a grid, then the detour factor is additionally bounded from above by

$$J(\mathcal{C}_{\mathcal{V}}(\mathcal{V}), \mathcal{C}_{\mathcal{E}}(\mathcal{E})) \leq \sqrt{2}. \tag{1.10}$$

From Theorem 1.47 we can directly deduce the optimal topology:

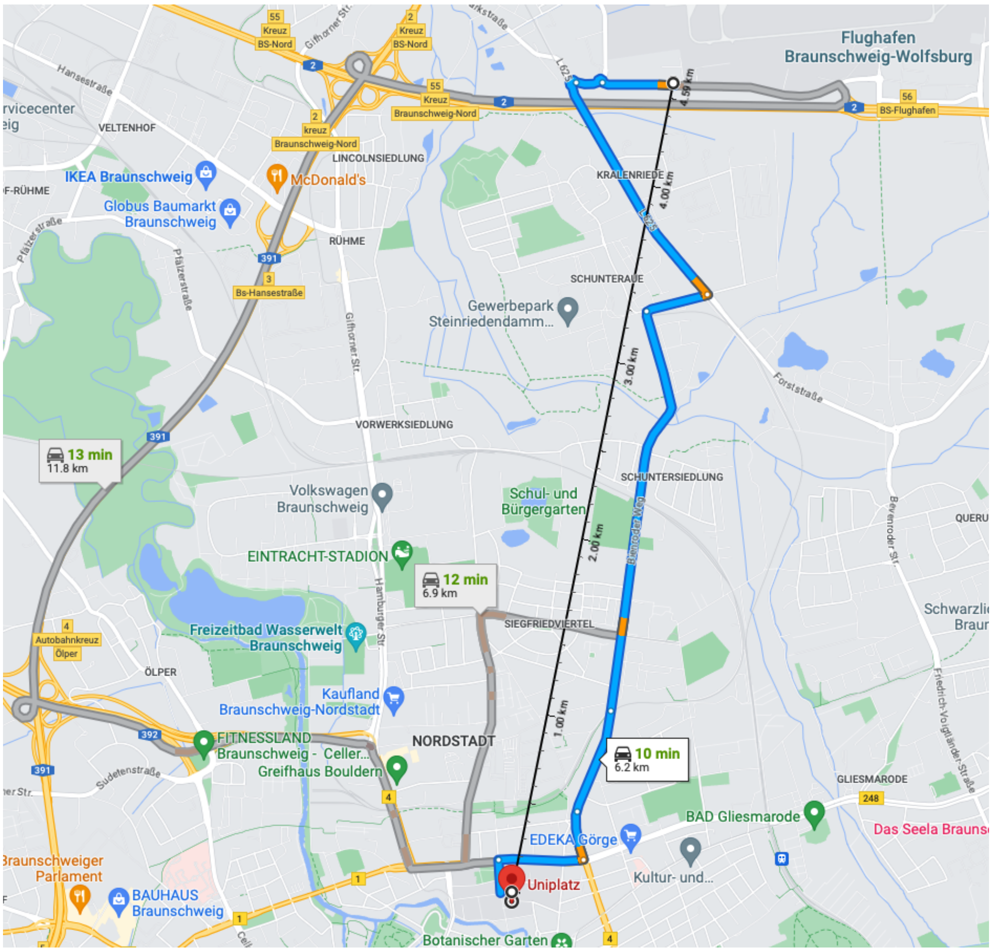


Figure 1.14: Manhattan distance using streets networks at TU Braunschweig

Corollary 1.48 (Detour optimal network).
If a given network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$ is complete and all edges $e \in \mathcal{E}$ are Euclidean connections, then the network is detour optimal, i.e. $\ell(\mathcal{C}_\mathcal{V}(\mathcal{V}), \mathcal{C}_\mathcal{E}(\mathcal{E})) = 1$.

Consequently, any network can be improved regarding the detour factor via direct connections. In general, a detour factor, which is significantly greater than 1.2 is deemed inefficient.

Remark 1.49
Both utilization and detour factor can be generalized by weighting components individually.

In the concluding section, we will link both models and discuss their interrelation.

1.4 Hierarchy of systems

As outlined before, we introduced network models to deal with planning problems on the tactical level, whereas dynamic models are used for control problems on operational level. Figure 1.15 displays the connections between the problems.

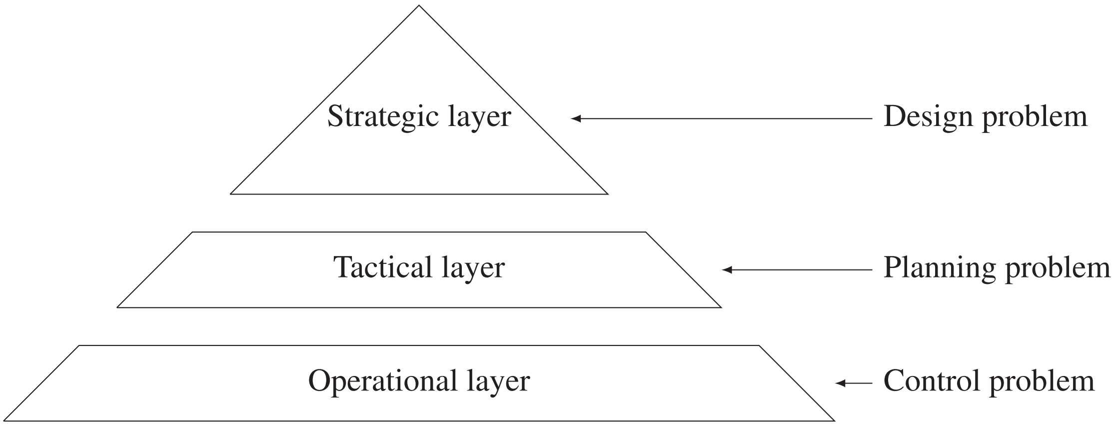


Figure 1.15: Working layers for transport and logistics systems

The solutions of the problems are connected in both directions. Within this lecture, we are particularly interested in the connection between the tactical and operational layer. For this case, the solution of the planning problem reveals the operational point required by the optimal control problem on the operational layer. The solution of the control problem is fed back to the network problem to ensure monitoring.
Between strategic and tactical layer, a similar circle exists. The strategic layer provides network structures, the tactical layer allows evaluation using simulation.

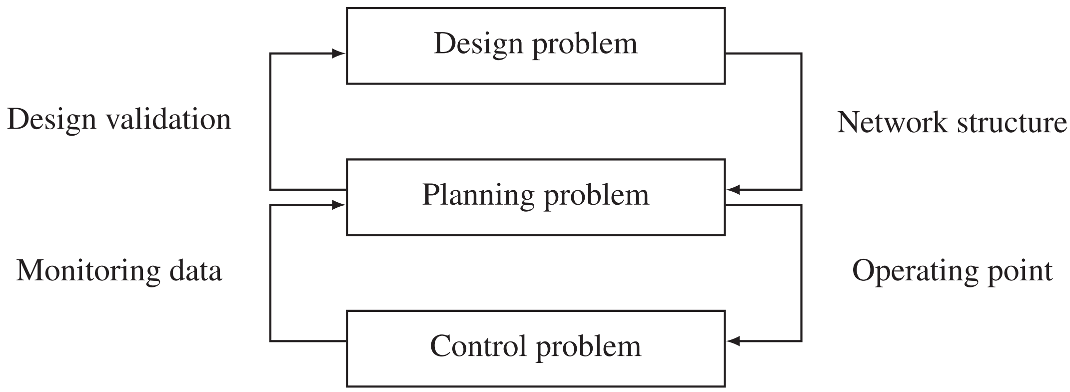
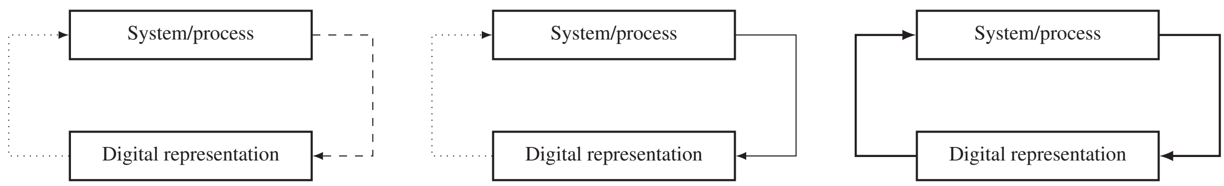


Figure 1.16: Feedback structure between layers/systems

In the literature, such representations are also called digital model/shadow/twin. More specifically, the latter differ as to their purpose:

Definition 1.50 (Digital model/shadow/twin).
Suppose a system/process with inputs and outputs, a digital representation of the same system/process and communication possibility between both to be given.

- If there exists at least a manual data flow from the system/process to the digital representation, then we call the digital respresentation a *digital model*.
- If there exists at least an automated data flow from the system/process to the digital representation, then we call the digital respresentation a *digital shadow*.
- If there exists a bidirectional automated data flow between the system/process and the digital representation, then we call the digital representation a *digital twin*.



(a) Digital model (b) Digital shadow (c) Digital twin

Figure 1.17: Difference between digital model/shadow/twin

Using Definition 1.50, we see that the difference between the three forms exists in the interaction structure. For the digital model, a data transfer is done manually from real system/process to the digital one. The intention of such a structure is to obtain insights into the system/process behavior and its properties. Its applications are on the strategic layer to design transport and logistics networks. Using an automated (and possible real time capable) data stream, the digital shadow can be used for monitoring purposes. Due to the automated nature, it is a reporting tool and can be used for planning. Last, the automated backwards flow to the physical system/process allows the digital twin to be applied on the operational layer.

CHAPTER 2

DESIGN AND PLANNING

> The line between disorder and order lies in logistics.
>
> Sun Tzu

As we have seen in the previous Chapter 1, networks can be utilized for design and planning problems. Within this chapter, we will derive solution methods for both tasks. On the strategic level, we start by the design problem of how a transport and logistics system should look like just in terms of a given KPI for utilities. Thereafter, we address the issue of constraints of utilities.
On the tactical level, we switch perspective from the global view of the entirety of the transport and logistics system towards single transports. In particular, we consider the search of the most efficient (in the sense of a KPI) connection between two infrastructure components within a transport and logistics system. Thereafter, we extend this problem to identify the most efficient connections between all vertexes.

2.1 Strategic level

Considering the design problem, we abstract from load units and consider the entirety of the transportation and logistics network. In this setting, the edges are utilities in the form of street, rail, waterways, air, pipelines or electricity lines. Here, we first ignore the issue of constraints and of the orientation of edges within the network.

2.1.1 Spanning tree problem

In case of a given infrastructure for a transport and logistics system, the problem arises to identify a network between all infrastructure components which is minimal in the sense of a KPI.

> **Remark 2.1**
> *We like to stress that the KPIs network utilization and network detour factors we introduced in the previous chapter are typical, there exists a wide variety of KPIs depending on the stakeholder group currently addressed. For this reason, network utilization and network detour are considered as (secondary) systemic indicators.*

In a general setting, there exist several options to get from one infrastructure component to another one using utilities and/or other infrastructure components. Figure 2.1 shows the realization of the German high speed train network, which represents a choice of realized options.

Figure 2.1: ICE network of Germany[1]

[1]Source: https://commons.wikimedia.org/wiki/File:Germany_-_ICE_line_network,_train_frequencies_and_top_speeds.svg

Examples of such problems may be the transport from production facilities to distribution centers using road, rail or waterways and/or transfer halls and repackaging. In a different setting, the network may represent gas pipelines linking storage areas, inflow manifolds, compressors and end users. A more generic setting is given in Figure 2.2 sketching an abstract transport network.

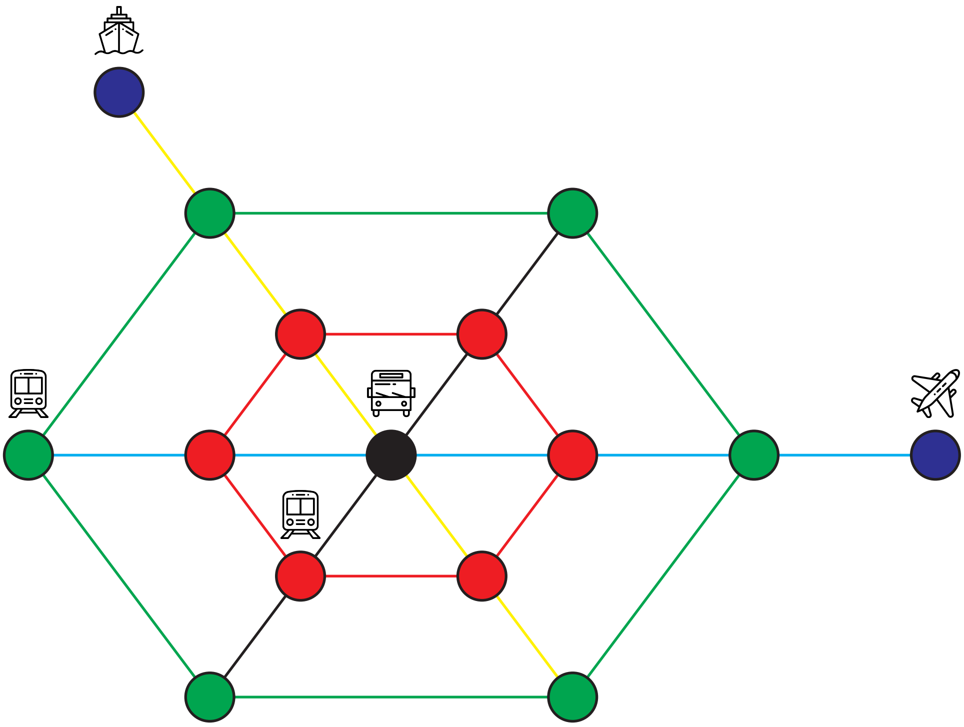


Figure 2.2: Example of spanning tree problem

In all cases, the question arises which utilities should be chosen to connect the infrastructure, or in network terms which edges should be chosen in order to obtain a cost minimal coverage of all vertexes. In order to be cost minimal, we require costs to be assigned to edges. To illustrate the solution approach for such a problem we utilize the example of such a network given in Figure 2.3.

Within this figure, there exists a variety of possibilities to connect the vertexes. Regarding a cost minimal solution, it is directly clear that there cannot exist more than one possibility to connect any two vertexes. To see the latter a simple counterexample can be used: If the solution is cost minimal suppose there exists a second possibility to get from one vertex to another. Since the costs of that edge are greater than zero, the edge can be left out reducing the total costs while still connecting all vertexes. Hence, the solution was not cost minimal.

To formalize this idea, we first introduce the concept of connecting two vertexes formally:

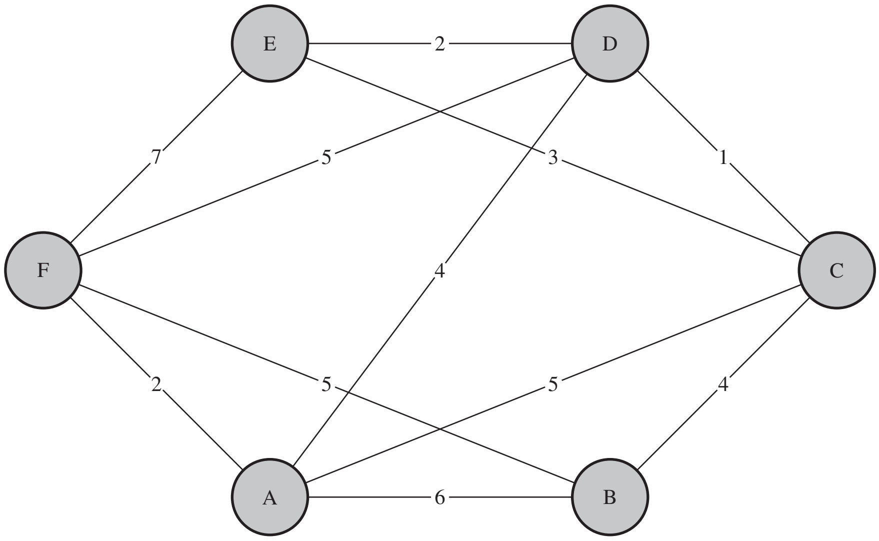


Figure 2.3: Example of a cost assigned network

Definition 2.2 (Path).
Suppose a network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$ to be given. Then we call a sequence of edges $p := \{e_j\}$ given by $e_j = (v_{j_1}, v_{j_2})$ a *path* if

$$v_{j_2} = v_{k_1} \tag{2.1}$$

holds for all j, k satisfying $k = j + 1$.

Task 2.3 (Path)
Consider the network given in Figure 2.3. Highlight a path between vertex A and vertex E.

Solution to Task 2.3: A path connecting A and E may include the edges AD and DE, cf. Figure 2.4.

Secondly, we introduce the so called cycle:

Definition 2.4 (Cycle).
Suppose a network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$ to be given. A path with sequence $p = (e_1, \ldots, e_j)$ is called a *cycle* if it is nonempty and the vertex sequence is of the form $(v_1, v_2, \ldots, v_j, v_1)$.

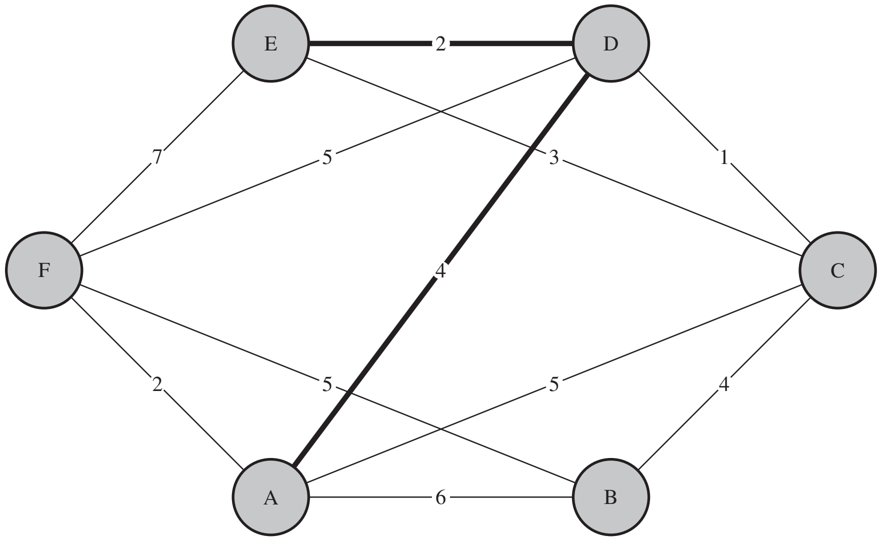


Figure 2.4: Example path within a cost assigned network

Task 2.5 (Path)
Consider the network given in Figure 2.3 and insert a cycle between vertex A and vertex E.

Solution to Task 2.5: A path connecting A and E may include the edges AD and DE, cf. Figure 2.5.

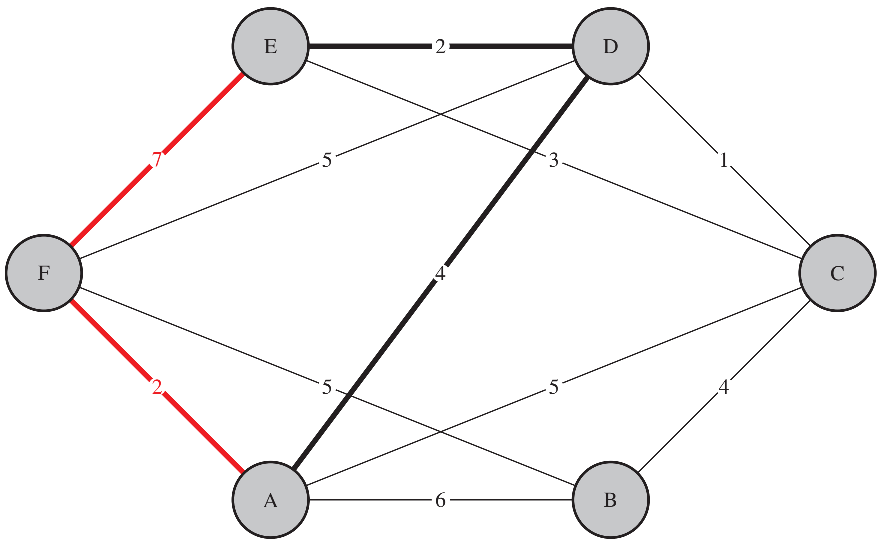


Figure 2.5: Example cycle within a cost assigned network

Using the picture of a cycle, we can now formally state the result we discussed beforehand:

Theorem 2.6 (Cost minimal solution).
For a given network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$ with multiplicities $\mathcal{C}_\mathcal{E} : \mathcal{E} \to \mathbb{R}_0^{n_\mathcal{E}}$ a cost minimal solution is cycle free.

The central question now is, how does such a solution look like and how can we compute it. Since the solution must be cycle free, it must look like a tree:

Definition 2.7 (Tree).
Given a network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$ a *tree* is a subset $\overline{\mathcal{E}} \subset \mathcal{E}$ such that

$$\forall v_j, v_k \in \mathcal{V} : \exists_1 \text{ path connecting } v_j, v_k. \tag{2.2}$$

Since our aim was to connect the entirety of infrastructure, we need to make sure that all vertexes are connected. To include this property, we require a so called spanning tree:

Definition 2.8 (Spanning tree).
Consider a network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$. Suppose $\overline{\mathcal{E}} \subset \mathcal{E}$ to be a tree. If

$$\forall v_j \in \mathcal{V} : \; \exists e \in \overline{\mathcal{E}} \wedge v_k \in \mathcal{V} : \; e = (v_j, v_k) \tag{2.3}$$

holds, then $\overline{\mathcal{E}}$ is called a *spanning tree*.

Task 2.9 (Spanning tree)
Given the network from Figure 2.3 insert two possible spanning trees.

Solution to Task 2.9: The two spanning trees are indicated using black and red in Figure 2.6.

As we have seen, there are several possibilities for spanning trees. To decide which of these trees is the best one considering the KPI of costs assigned to the edges, we directly obtain the following:

Definition 2.10 (Minimal spanning tree).
For a network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$ with multiplicities $\mathcal{C}_\mathcal{E} : \mathcal{E} \to \mathbb{R}_0^{n_\mathcal{E}}$ we call a tree a *minimal spanning*

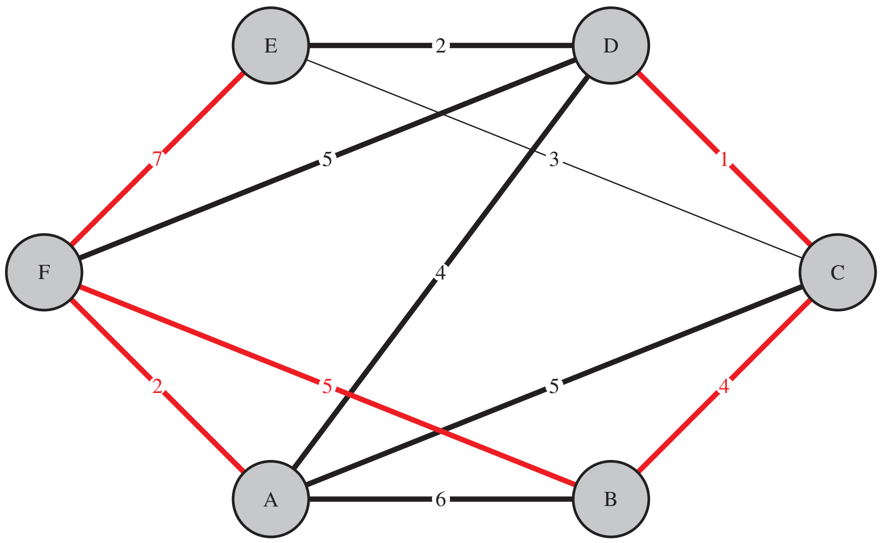


Figure 2.6: Example spanning trees within a cost assigned network

tree if it spans all vertexes $\mathcal{V}$ and minimizes the multiplicities within the tree, i.e.

$$\overline{\mathcal{E}} = \underset{\overline{\mathcal{E}}}{argmin} \sum_{e \in \overline{\mathcal{E}}} \mathcal{C}_{\mathcal{E}}(e). \tag{2.4}$$

Having answered the question which properties a solution to cost minimal connection of vertexes has, we now require a method to compute such a solution. Since the property (2.4) is linear and the network itself is also linear, we can obtain a solution using a so called greedy approach.

For our specific problem of identifying a minimal spanning tree, two different methods exist. The first one is called *Prim's algorithm* and aims to complete a spanning tree in the cost minimal way possible.

Algorithm 1 Prim algorithm for minimal spanning tree

Input: Connected network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$
Input: Multiplicities $\mathcal{C}_{\mathcal{E}} : \mathcal{E} \to \mathbb{R}_0^{n_{\mathcal{E}}}$

procedure CLASS PRIM($\mathcal{N}, \mathcal{C}_{\mathcal{E}}$)
  $\overline{\mathcal{E}} \leftarrow \emptyset$
  $\overline{\mathcal{E}} \leftarrow \overline{\mathcal{E}} \cup \{e = argmin_{e \in \mathcal{E}} \mathcal{C}_{\mathcal{E}}(e)\}$
  for $j = 2, \ldots, n_{\mathcal{V}} - 1$ **do**
    $\overline{\mathcal{E}} \leftarrow \overline{\mathcal{E}} \cup \{e = argmin_{e \in \overline{\mathcal{E}} \cup \{e\}} \mathcal{C}_{\mathcal{E}}(e) \mid \overline{\mathcal{E}} \cup \{e\}$ is a tree$\}$
  end for
end procedure

Output: Minimal spanning tree $\overline{\mathcal{E}}$

The idea of Prim's Algorithm 1 is to start with a cost minimal edge defining an initial tree. Based on this tree, the algorithm considers only edges which are adjacent to the current tree and choose the cost minimal one of them. Continuing this way, a total of $n_{\mathcal{V}} - 1$ edges are chosen resulting in a spanning tree.

Task 2.11 (Minimal spanning tree)
Given the network from Figure 2.3 compute a minimal spanning tree using Algorithm 1.

Solution to Task 2.11: The result is given in Figure 2.7. The minimal cost is 14.

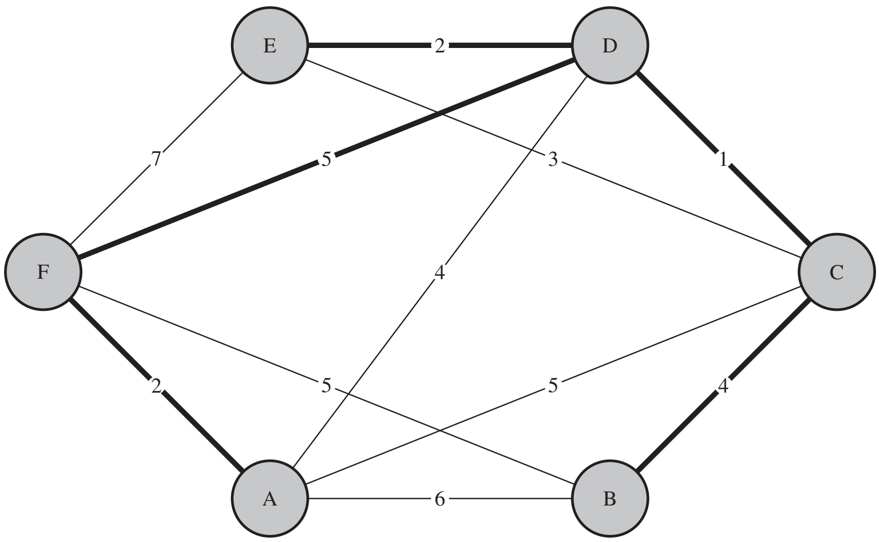


Figure 2.7: Minimal spanning tree using Prim's Algorithm 1

Remark 2.12
Note that the result of Prim's algorithm is not unique. In fact, neither the choice of the initial edge nor the intermediate choices may be unique if two edges with identical costs exist.

A different approach is the so called Kruskal's Algorithm 2. In contrast to Prim, Kruskal always uses the cost minimal edge and adds it to a set.
As a consequence of this strategy, Kruskal's Algorithm considers the entirety of vertexes as a forest of several trees. In each step, it fuses two trees to a bigger one. Hence, after $n_{\mathcal{V}} - 1$ steps only one tree remains, which is then also a spanning tree.

Algorithm 2 Kruskal algorithm for minimal spanning tree

Input: Connected network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$
Input: Multiplicities $\mathcal{C}_{\mathcal{E}} : \mathcal{E} \to \mathbb{R}_0^{n_{\mathcal{E}}}$

procedure CLASS KRUSKAL($\mathcal{N}, \mathcal{C}_{\mathcal{E}}$)
  $\overline{\mathcal{E}} \leftarrow \emptyset$
  $\overline{\mathcal{E}} \leftarrow \overline{\mathcal{E}} \cup \{e = argmin_{e \in \mathcal{E}} \mathcal{C}_{\mathcal{E}}(e)\}$
  for $j = 1, \ldots, n_{\mathcal{V}}$ **do**
    $\overline{\mathcal{E}} \leftarrow \overline{\mathcal{E}} \cup \{e = argmin_{e \in \mathcal{E}} \mathcal{C}_{\mathcal{E}}(e) \mid \overline{\mathcal{E}} \cup \{e\}$ is free of circles$\}$
  end for
end procedure
Output: Minimal spanning tree $\overline{\mathcal{E}}$

Task 2.13 (Minimal spanning tree)
Given the network from Figure 2.3 compute a minimal spanning tree using Algorithm 2.

Solution to Task 2.13: The result is given in Figure 2.8. Similar to Task 2.11 the minimal costs are 14.

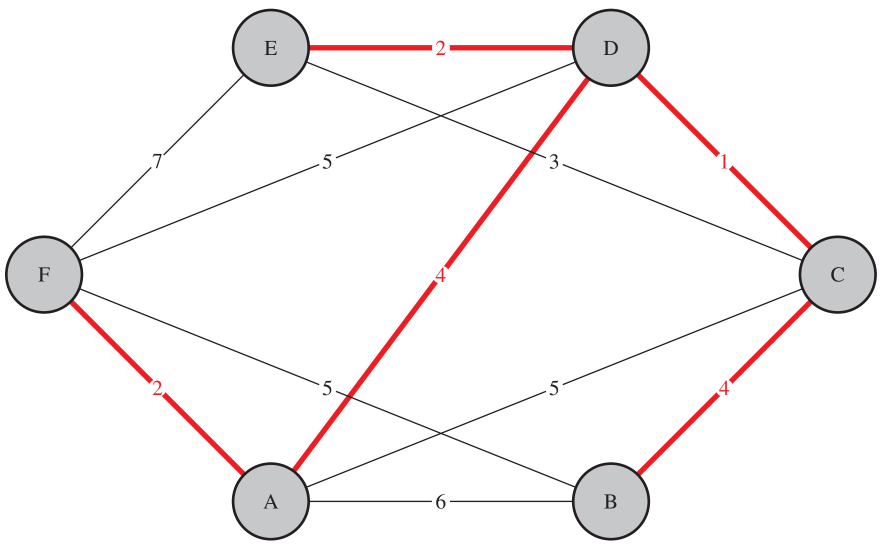


Figure 2.8: Minimal spanning tree using Kruskal's Algorithm 2

Remark 2.14
We like to note that both Prim's and Kruskal's algorithm require the network to be connected. If this is not the case, no spanning tree exists. If these algorithms are applied to such a network, both algorithms are capable to identify that no spanning tree exists.

Remark 2.15
Kruskal's algorithm can be implemented very efficiently if the set of edges $\mathcal{E}$ is stored as a heap sorted by the multiplicities. This is not possible for Prim's algorithm.

Table 2.1: Advantages and disadvantages of Prim/Kruskal algorithms

Advantage	**Disadvantage**
✓ Compute basic network	✗ Neglects directions
✓ Includes any KPI	✗ Neglects constraints

As we have seen, we can use Prim's and Kruskal's algorithms to compute basic supply network. Up till now, we supposed that the edges may be used in both directions and did not worry about capacities. In the following, we will dig deeper into networks using constraints and orientations.

2.1.2 Flow problem

The flow problem is coming up in case when a network consisting of vertexes and edges, which are limited in capacity and also exhibit a direction. In transportation systems, these correspond to limits of load carriers per edge. Here, one central question is to assess how many load carriers can be transported from one vertex to another one. The flow problem is also termed *transportation problem*.
To answer this question, we first need to clarify how a transport can be put onto a network. In that regard, if a transport is executed, the remaining capacities of edges are modified. In order to calculate on such a network, we first introduce some basic terms to properly describe a flow within a network.

Definition 2.16 (Source, sink, predecessor and successor set).
Consider a directed network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$. For any vertex $v_j \in \mathcal{V}$ we call

$$\mathcal{S}(v_j) := \{v \in \mathcal{V} \mid \exists (v_j, v) \in \mathcal{E}\} \tag{2.5}$$
$$\mathcal{P}(v_j) := \{v \in \mathcal{V} \mid \exists (v, v_j) \in \mathcal{E}\} \tag{2.6}$$

successor and predecessor set of v_j. Moreover, we identify vertexes as *sinks* if the condition

$$s = \{v \in \mathcal{V} \mid \mathcal{S}(v) = \emptyset\} \tag{2.7}$$

and *sources* if

$$r = \{v \in \mathcal{V} \mid \mathcal{P}(v) = \emptyset\} \tag{2.8}$$

holds.

Task 2.17 (Source, sink, predecessor and successor set)
Given the network from Figure 2.9 mark sources r and sinks s as well as the predecessor set $\mathcal{P}(s)$ and the successor set $\mathcal{S}(r)$.

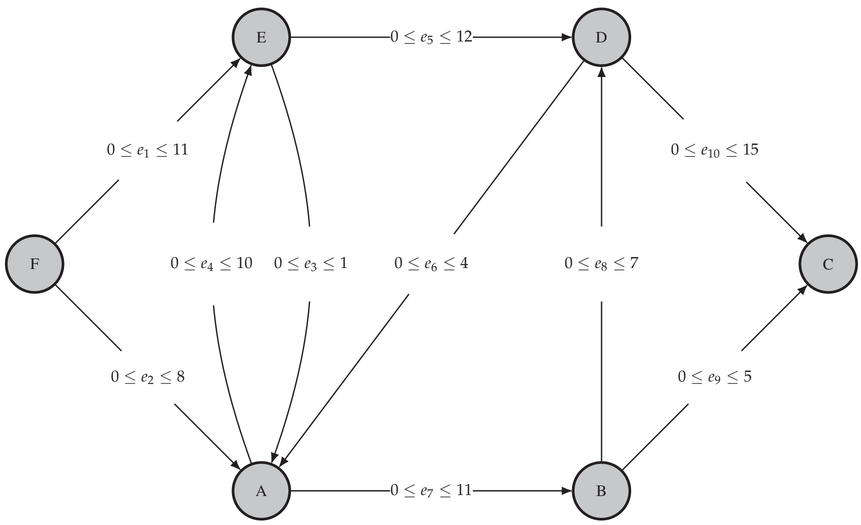


Figure 2.9: Example of a directed network

Solution to Task 2.17: Source and its successor set are displayed in red and sink and its predecessor set are shown in blue in Figure 2.10.

Using sources and sinks, we can introduce the concept of a flow:

Definition 2.18 (Flow).
Suppose a directed network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$ to be given. Then we call a function $\mathcal{F} : \mathcal{E} \to$

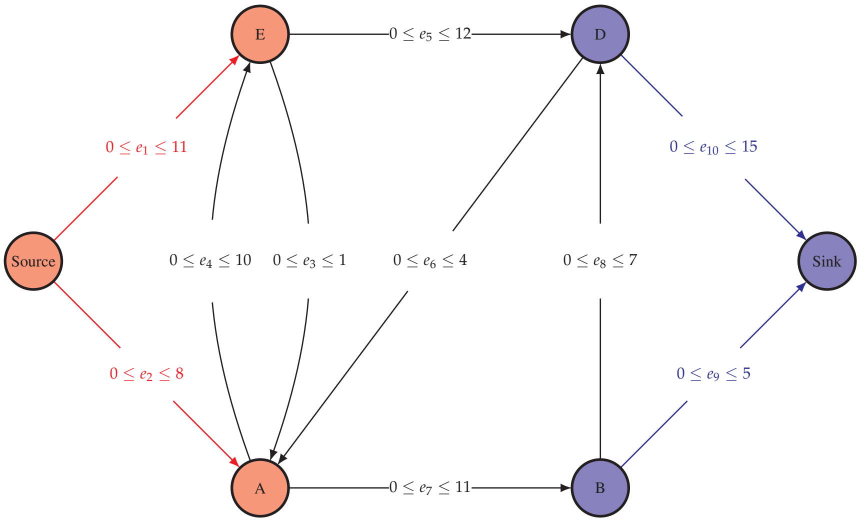


Figure 2.10: Sources, sinks as well as predecessor and successor set of example network from Figure 2.9

$\mathbb{R}^+ \cup \{\infty\}$ a *flow relation* if it satisfies the *flow condition*

$$\sum_{v_k \in \mathcal{S}(v_j)} \mathcal{F}(e_{jk}) - \sum_{v_l \in \mathcal{P}(v_j)} \mathcal{F}(e_{lj}) = \begin{cases} \omega, & \text{if } v_j = r \\ -\omega, & \text{if } v_j = s \\ 0, & \text{if } v_j \in \mathcal{V} \setminus \{r, s\} \end{cases} \tag{2.9}$$

for all vertexes $v_j \in \mathcal{V}$ where $r, s \in \mathcal{V}$ denote source and sink of the network. We refer to ω as the *flow strength*.

For intermodal systems, the latter condition (2.9)

$$\sum_{v_k \in \mathcal{S}(v_j)} \mathcal{F}(e_{jk}) - \sum_{v_l \in \mathcal{P}(v_j)} \mathcal{F}(e_{lj}) = 0, \quad \text{if } v_j \in \mathcal{V} \setminus \{r, s\}$$

is of particular importance: It refers to the handling of load units from one edge to another without storage, i.e. commissioning/decommissioning. Hence, the vertexes satisfying this condition are possible vertex for intermodal transport changes.

It is worth mentioning that a flow relation is not the same as a path. While both represent connections between two vertexes, a path is a line whereas a flow can be a line or multiple lines between the vertexes.

Task 2.19 (Flow)
Given the network from Figure 2.10 insert a flow from source to sink.

Solution to Task 2.19: A flow relation is shown in Figure 2.11.

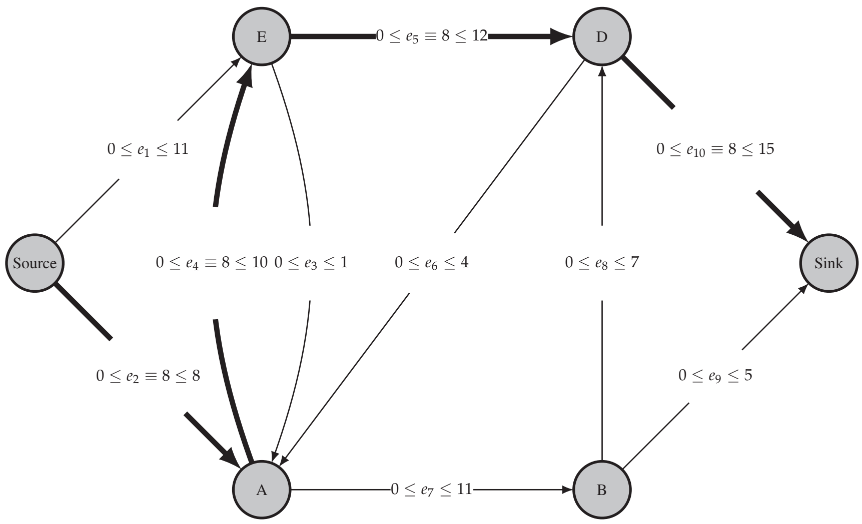


Figure 2.11: Flow relation within the example network from Figure 2.9

Remark 2.20
Note that the flow condition (2.9) *resembles Kirchhoff's law for electrics, i.e. the inflow and outflow at each vertex of an electrical network are identical.*

Since we are dealing with capacities on the edges, we need to include these into the definition of the flow.

Definition 2.21 (Feasible flow).
Given a directed network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$ with constraints

$$\underline{e}_{kl} \le e_{kl} \le \overline{e}_{kl} \tag{2.10}$$

we call a flow relation $\mathcal{F} : \mathcal{E} \to \mathbb{R}^+ \cup \{\infty\}$ *feasible* if

$$\underline{e}_{kl} \le \mathcal{F}(e_{kl}) \le \overline{e}_{kl} \tag{2.11}$$

holds for all $e_{kl} = (v_k, v_l) \in \mathcal{E}$.

One special case of a feasible flow is the so called zero flow. In practice, a zero flow is quite common as it refers to the case of no transportation within the network.

Definition 2.22 (Zero flow).
Consider a directed network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$ with constraints (2.10) and a flow relation $\mathcal{F} : \mathcal{E} \to \mathbb{R}^+ \cup \{\infty\}$. If the condition

$$\mathcal{F}(e_{kl}) = 0 \tag{2.12}$$

holds for all $e_{kl} = (v_k, v_l) \in \mathcal{E}$, then it is called *zero flow*.

Note that a zero flow is only feasible if $\underline{e}_j = 0$ holds for all $e_j \in \mathcal{E}$. As such, it is also a prime candidate to start any iterative algorithm to compute the best possible flow. The latter already points us in the direction to use a KPI and assess possible solutions respectively. As a flow relation is typically not single valued, it is difficult to compare flow relations with one another. To render flow relations comparable, we introduce the concept of a flow order.

Definition 2.23 (Flow order).
Given a directed and connected network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$ with constraints (2.10) and flow relation $\mathcal{F}^1, \mathcal{F}^2 : \mathcal{E} \to \mathbb{R}^+ \cup \{\infty\}$. Then we define the order of flows via the flow strength

$$\omega(\mathcal{F}^1) > \omega(\mathcal{F}^2) \tag{2.13}$$

resembling the natural order $>$ in $\mathbb{R}$.

Task 2.24 (Flow order)
Consider the flow relation from Figure 2.11 suggest a improved flow relation.

Solution to Task 2.24: An improved flow relation is shown in Figure 2.12.

Based on this order, it is straight forward to define the problem of finding a maximal flow within a network.

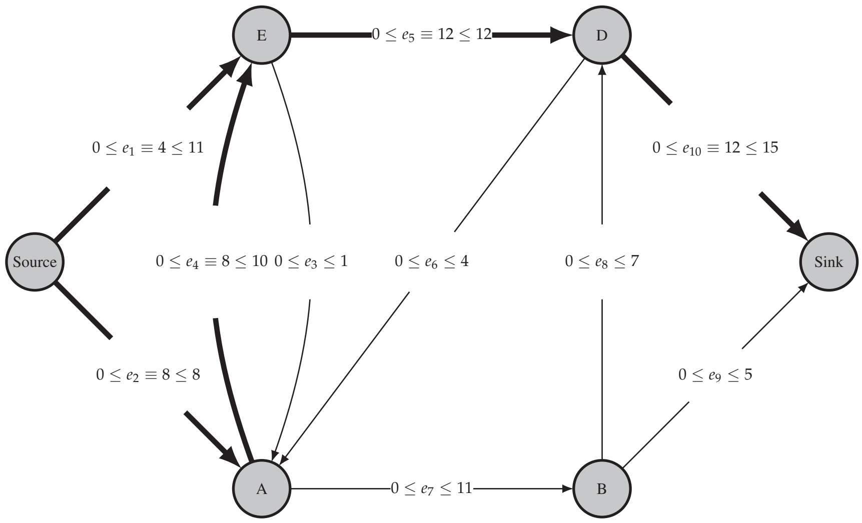


Figure 2.12: Improved flow relation within example network from Figure 2.9

Definition 2.25 (Maximal flow problem).
For a directed and connected network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$ with constraints (2.10) we call a flow relation $\mathcal{F} : \mathcal{E} \to \mathbb{R}^+ \cup \{\infty\}$ maximal if it solves the *maximal flow problem*

$$\max_{\mathcal{F}} \omega(\mathcal{F}) \tag{2.14}$$

$$\text{such that} \quad \sum_{v_k \in \mathcal{S}(v_j)} \mathcal{F}(e_{jk}) - \sum_{v_l \in \mathcal{P}(v_j)} \mathcal{F}(e_{lj}) = \begin{cases} \omega, & \text{if } v_j = r \\ -\omega, & \text{if } v_j = s \\ 0, & \text{if } v_j \in \mathcal{V} \setminus \{r, s\} \end{cases}$$

$$\underline{e}_{kl} \leq \mathcal{F}(e_{kl}) \leq \bar{e}_{kl} \qquad \forall e_{kl} = (v_k, v_l) \in \mathcal{E}.$$

Before we start to design a method or algorithm to compute a maximal flow, we need to make sure that such a flow actually exists. Fortunately, existence can be guaranteed very easily.

Theorem 2.26 (Existence of maximal flow).
Suppose a directed and connected network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$ with constraints (2.10) *to be given. If there exists a feasible flow relation $\mathcal{F} : \mathcal{E} \to \mathbb{R}^+ \cup \{\infty\}$ and $\bar{e}_{kl} < \infty$ for all $e_{kl} = (v_k, v_l) \in \mathcal{E}$, then there exists a solution to maximal flow problem* (2.14).

Task 2.27 (Existence of maximal flow)
Given the network from Figure 2.10. Show that the conditions of Theorem 2.26 hold.

Solution to Task 2.27: For all edges we can directly observe that $\overline{e}_j < \infty$ holds for all $e_j \in \mathcal{E}$. Moreover, as all lower bounds satisfy $\underline{e}_j = 0$ for all $e_j \in \mathcal{E}$, there exists a zero flow. Hence, existence of a feasible flow relation is shown and the conditions of Theorem 2.26 hold.

As the existence theorem already indicates, we require a feasible solution. Based on the latter, we can derive improvements. As we are dealing with flows, such an improvement is characterized by a flow increase:

Definition 2.28 (Flow increasing path).
Suppose a directed and connected network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$ with constraints (2.10) to be given. Let $r, s \in \mathcal{V}$ be source and sink of the network and suppose $p(r, s)$ to be a path connecting $r \in \mathcal{V}$ and $s \in \mathcal{V}$. Furthermore let $\mathcal{F} : \mathcal{E} \to \mathbb{R}^+ \cup \{\infty\}$ be a feasible flow relation. If there exists $p > 0$ such that

$$p \leq p(v_k, v_l) := \begin{cases} \overline{e}_{kl} - \mathcal{F}(e_{kl}), & \text{if } e_{kl} = (v_k, v_l) \text{ and } e_{kl} \text{ is oriented along } p(r, s) \\ \mathcal{F}(e_{kl}) - \underline{e}_{kl}, & \text{if } e_{kl} = (v_k, v_l) \text{ and } e_{kl} \text{ is oriented against } p(r, s) \end{cases} \tag{2.15}$$

holds, then the path $p(r, s)$ is called *flow increasing*.

While we could manually searched for a flow increasing path, we require a function to do so using a computer. In order to start such a calculation, we require knowledge on which edges still provide open capacities in both forward or backward flow. A respective list can be generated using Algorithm 3.
The flow capacity calculation now allows us to identify a flow increasing path. A respective function is given by Algorithm 4.
Again, we can use our ongoing example to highlight the computation of a flow increasing path.

Task 2.29 (Flow increasing path)
Consider the flow relation from Figure 2.12. Insert a flow increasing path.

Algorithm 3 Flow capacity calculation

Input: Feasible flow relation $\mathcal{F} : \mathcal{E} \to \mathbb{R}^+ \cup \{\infty\}$
Input: Constraints $\underline{e}_{jk}, \overline{e}_{jk}$ for all $e_{jk} = (v_j, v_k) \in \mathcal{E}$

function CALCULATEFLOWCAPACITIES($\mathcal{N}, \mathcal{C}_{\mathcal{E}}, \mathcal{F}$)
    for $j = 1, \ldots, n_{\mathcal{V}}$ **do**
        $\mathcal{M}(v_j) \leftarrow \emptyset$ ▷ $\mathcal{M}(v_j)$ will hold all markable vertexes connected to v_j
    end for
    for $j = 1, \ldots, n_{\mathcal{V}}$ **do**
        for all $v_k \in \mathcal{S}(v_j)$ **do**
            Identify $e_{jk} = (v_j, v_k)$
            if $\mathcal{F}(e_{jk}) < \overline{e}_{jk}$ **then**
                $\mathcal{M}(v_j) \leftarrow \mathcal{M}(v_j) \cup \{v_k\}$ ▷ Allows forward increase of capacity
            else if $\mathcal{F}(e_{jk}) > \underline{e}_{jk}$ **then**
                $\mathcal{M}(v_k) \leftarrow \mathcal{M}(v_k) \cup \{v_j\}$ ▷ Allows backward decrease of capacity
            end if
        end for
    end for
end function
Output: Flow capacities $\mathcal{M}$

Solution to Task 2.29: As a path is simply connected, we consider edge e_{10} in Figure 2.12 to be improved. One (and in this case the only) remaining path with unused capacities from source to sink is given by $(e_1, -e_4, e_7, e_8, e_{10})$. In this case, e_{10} is the limiting factor as it allows for a maximum increase of 3 units. The result is shown in Figure 2.13.

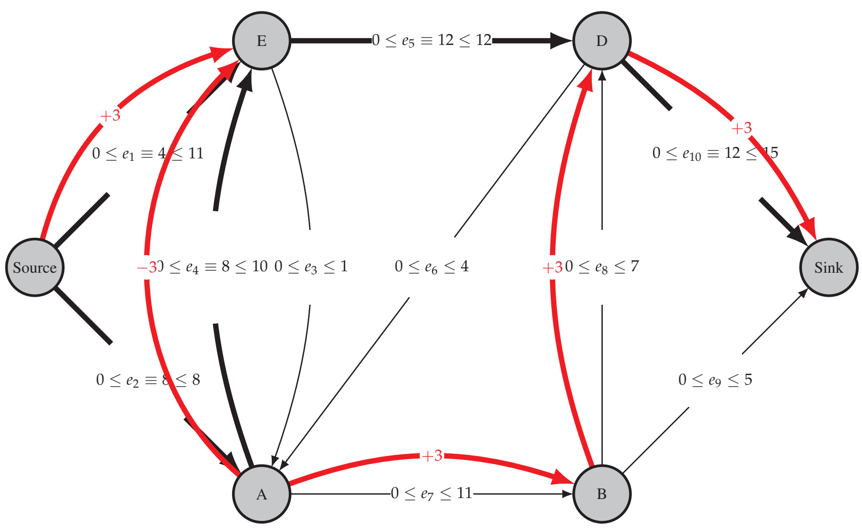


Figure 2.13: Flow increasing path within example network from Figure 2.12

Algorithm 4 Algorithm to compute a flow increasing path

Input: Connected network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$
Input: Feasible flow relation $\mathcal{F} : \mathcal{E} \to \mathbb{R}^{+} \cup \{\infty\}$
Input: Constraints $\underline{e}_{jk}, \overline{e}_{jk}$ for all $e_{jk} = (v_j, v_k) \in \mathcal{E}$
Input: Flow capacities $\mathcal{M}$

function CALCULATEFLOWINCREASINGPATH($\mathcal{N}, \mathcal{F}, (\underline{e}_{jk}, \overline{e}_{jk}), \mathcal{M}$)
  $b \leftarrow$ false, $p_r \leftarrow r$, $\varepsilon_r \leftarrow \infty$, $Q \leftarrow \{r\}$, $L \leftarrow \{r\}$ ▷ Q and L are snakes
  while $Q \neq \emptyset$ **do**
    Obtain v_k from $Q(1)$, remove head of Q ▷ $Q(1)$ is head of snake
    for all $v_k \in \mathcal{M}(v_j) \setminus L$ **do**
      Insert v_k at end of Q and L
      if $v_k \in \mathcal{S}(v_j)$ **then**
        $p_{v_k} \leftarrow v_j$, $\varepsilon_{v_k} \leftarrow \min\{\varepsilon_{v_j}, \overline{e}_{jk} - \mathcal{F}(e_{jk})\}$ ▷ Forward marking
      else
        $p_{v_k} \leftarrow -v_j$, $\varepsilon_{v_k} \leftarrow \min\{\varepsilon_{v_j}, \mathcal{F}(e_{jk}) - \underline{e}_{jk}\}$ ▷ Backward marking
      end if
      if $v_k = s$ **then**
        Terminate ▷ Sink s is marked
      end if
    end for
  end while
  $b \leftarrow$ true ▷ No connection to sink s
end function

Output: Stopping criterion b
Output: Flow increase ε_s
Output: Flow direction list p

Here, we like to point out that Algorithm 4 is able to identify whether or not a flow increasing path exists. Hence, if no flow increasing path exists, this knowledge can be used as a breaking criterion in finding a maximal flow. Theorem 2.30 formalizes this finding.

Theorem 2.30 (Maximal flow).
Given a directed and connected network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$ *with constraints* (2.10) *and a feasible flow relation* $\mathcal{F} : \mathcal{E} \to \mathbb{R}^{+} \cup \{\infty\}$. *If there exists no flow increasing path, then the flow relation is maximal.*

Task 2.31 (Flow increasing path)
Given the network from Figure 2.9 insert the maximal flow.

Solution to Task 2.31: The maximal flow is displayed in Figure 2.14.

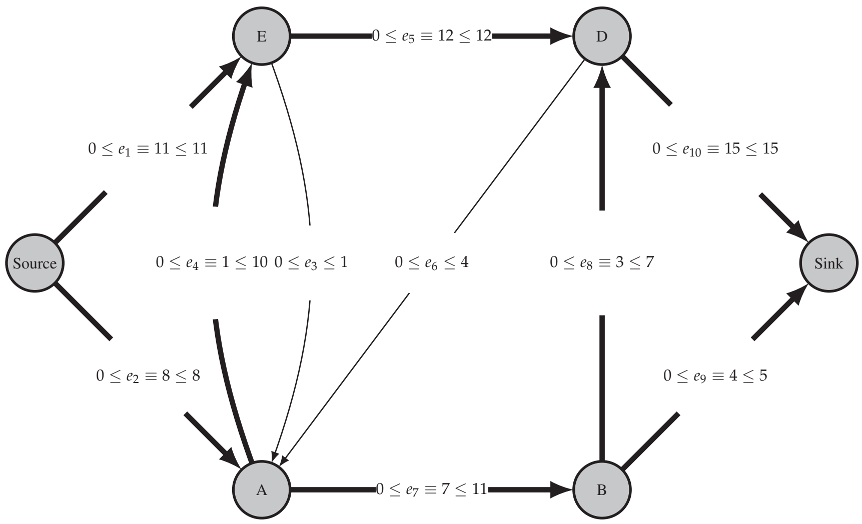


Figure 2.14: Maximal flow for example network from Figure 2.9

Note that due to the property of nonexistence of a flow increasing path, the latter results is identical to a cut in the network, i.e. the network is split into two disconnected parts. To this end, only the remaining capacities for each edge are shown.

Theorem 2.32 (Max flow – min cut).
Given a directed and connected network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$ *with constraints* (2.10). *Then the maximal flow from a source* $r \in \mathcal{V}$ *to a sink* $s \in \mathcal{V}$ *is identical to the capacity of the minimal cut in* $\mathcal{N}$.

Task 2.33 (Minimal cut within a network)
Given the maximal flow relation from Figure 2.14 show the minimum cut within the network.

Solution to Task 2.33: The minimum cut is displayed in Figure 2.15. In our case, the minimal cut separated the source from the rest of the network.

Before coming to an overall algorithm to compute maximal flows, we first need to be able to increment a given flow using a flow increasing path. A respective function is outlined in Algorithm 5.

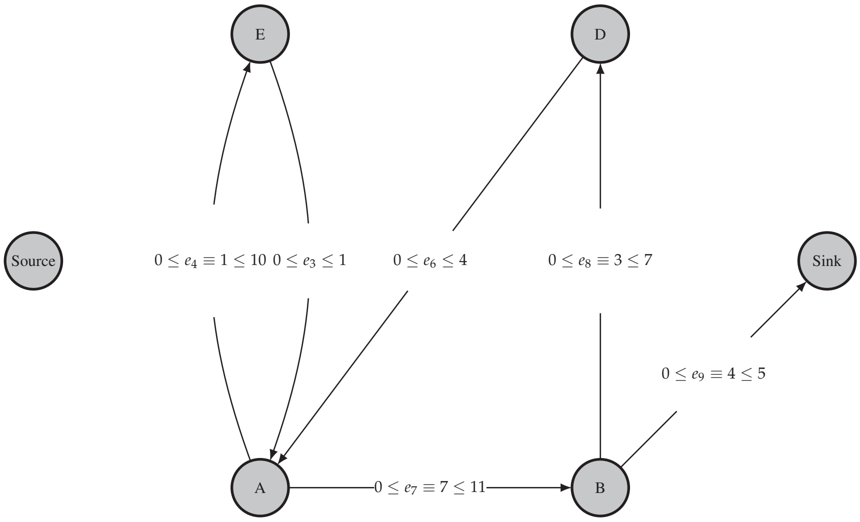


Figure 2.15: Minimal cut for example network from Figure 2.9

Algorithm 5 Algorithm to add an increasing path to a flow relation

---

Input: Connected network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$
Input: Feasible flow relation $\mathcal{F} : \mathcal{E} \to \mathbb{R}^+ \cup \{\infty\}$
Input: Constraints $\underline{e}_{jk}, \overline{e}_{jk}$ for all $e_{jk} = (v_j, v_k) \in \mathcal{E}$
Input: Flow capacities $\mathcal{M}$, flow increase ε_s, flow direction list p, flow strength ω

function FLOWINCREASE($\mathcal{N}, \mathcal{F}, (\underline{e}_{jk}, \overline{e}_{jk}), \mathcal{M}, \varepsilon_s, p, \omega$)
    $\omega \leftarrow \omega + \varepsilon_s$, set j indicator of sink s
    while $v_j \neq r$ **do** ▷ Until source is reached
        $k \leftarrow j$, set j indicator of p_{v_j}
        $\mathcal{M}(v_k) \leftarrow \mathcal{M}(v_k) \cup \{v_j\}$
        if $p_{v_k} > 0$ **then**
            $\mathcal{F}(e_{jk}) \leftarrow \mathcal{F}(e_{jk}) + \varepsilon_s$ ▷ Forward mark increases flow
            if $\mathcal{F}(e_{jk}) = \overline{e}_{jk}$ **then**
                $\mathcal{M}(v_j) \leftarrow \mathcal{M}(v_j) \setminus \{v_k\}$
            end if
        else
            $\mathcal{F}(e_{jk}) \leftarrow \mathcal{F}(e_{jk}) - \varepsilon_s$ ▷ Backward mark decreases flow
            if $\mathcal{F}(e_{jk}) = \underline{e}_{jk}$ **then**
                $\mathcal{M}(v_j) \leftarrow \mathcal{M}(v_j) \setminus \{v_k\}$
            end if
        end if
    end while
end function

Output: Flow relation $\mathcal{F}$
Output: Flow capacities $\mathcal{M}$
Output: Flow strength ω

The result we obtain from Algorithm 5 is twofold: For one, the algorithm adds the flow increasing path to the already existing flow relation. At the same time, the algorithm also updates the remaining flow capacities for forward and backward flows. Hence, we are not required to recompute these using Algorithm 3.

Now, we can combine Algorithms 3–5 to obtain the so called Ford-Fulkerson algorithm for computing maximal flows. The basic requirement of this algorithm is that there exists a zero flow, which is also used as initialization of the algorithm. Algorithm 6 resembles the respective pseudocode.

Algorithm 6 Ford-Fulkerson algorithm for maximal flows

Input: Connected network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$
Input: Feasible flow relation $\mathcal{F} : \mathcal{E} \to \mathbb{R}^+ \cup \{\infty\}$
Input: Constraints $\underline{e}_{jk}, \overline{e}_{jk}$ for all $e_{jk} = (v_j, v_k) \in \mathcal{E}$

```
procedure CLASS FORD-FULKERSON(𝒩, ℱ, (e̲_jk, ē_jk))
    ω ← 0
    ℳ ← INITIALIZEFLOWCAPACITIES(ℱ, (e̲_jk, ē_jk))
    [b, ε_s, p] ← CALCULATEFLOWINCREASINGPATH(𝒩, ℱ, (e̲_jk, ē_jk), ℳ)
    while b = false do
        [ℱ, ℳ, ω] ← FLOWINCREASE(𝒩, ℱ, (e̲_jk, ē_jk), ℳ, ε_s, p, ω)
        [b, ε_s, p] ← CALCULATEFLOWINCREASINGPATH(𝒩, ℱ, (e̲_jk, ē_jk), ℳ)
    end while
end procedure
```

Output: Maximal flow $\mathcal{F}$

Table 2.2: Advantages and disadvantages of Ford-Fulkerson algorithm

Advantage	Disadvantage
✓ Addresses constraints	✗ Neglects costs
✓ Includes directions	✗ Results in NP problem with costs

The result of the maximal flow problem (2.14) in a second step be used to find a cost minimal flow of given flow strength. This reveals the so called cost minimal flow or transshipment problem.

Definition 2.34 (Cost minimal flow problem).
Consider a network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$ with multiplicities $\mathcal{C}_{\mathcal{E}} : \mathcal{E} \to \mathbb{N}_0^{\mathcal{E}}$ and constraints (2.10) and suppose a flow strength $\omega \in \mathbb{R}^+$ to be given. Then we call

$$\min_{\mathcal{F}} \sum_{e \in \mathcal{E}} \mathcal{C}_{\mathcal{E}}(e) \cdot \mathcal{F}(e) \tag{2.16}$$

$$\text{such that} \sum_{v_k \in \mathcal{S}(v_j)} \mathcal{F}(e_{jk}) - \sum_{v_l \in \mathcal{P}(v_j)} \mathcal{F}(e_{lj}) = \begin{cases} \omega, & \text{if } v_j = r \\ -\omega, & \text{if } v_j = s \\ 0, & \text{if } v_j \in \mathcal{V} \setminus \{r, s\} \end{cases}$$

$$\underline{e}_{jk} \leq \mathcal{F}(e_{jk}) \leq \overline{e}_{jk} \qquad \forall e_{jk} \in \mathcal{E}.$$

cost minimal flow problem.

Technically, the latter problem can be reformulated using the incidence matrix to obtain a linear optimization problem of the form

$$\begin{aligned} &\min \mathcal{C}_{\mathcal{E}}^{\top} \cdot \mathcal{F} \\ &\text{such that } H \cdot \mathcal{F} = \omega \\ &\qquad \underline{\mathbf{e}} \leq \mathcal{F} \leq \overline{\mathbf{e}} \end{aligned} \qquad \Longleftrightarrow \qquad \begin{aligned} &\min c^{\top} \cdot x \\ &\text{such that } A \cdot x = b \\ &\qquad \underline{x} \leq x \leq \overline{x} \end{aligned}$$

which can be solved using the simplex method. This is outside the scope of this lecture.

Remark 2.35
We like to note that the combination of cost minimal maximal flow is possible by redefining the KPI in problem (2.16) *to*

$$\min_{\mathcal{F}} \max_{\omega} \sum_{e \in \mathcal{E}} \mathcal{C}_{\mathcal{E}}(e) \cdot \mathcal{F}(e). \tag{2.17}$$

Such a min-max problem is typically NP hard, yet an efficient solution for this particular problem can be found using the Busacker-Gowen algorithm, cf. [7] for details.

2.2 Tactical level

On the tactical level, planning of tours for load units is the most prominent task. Similar to the cost minimal flow problem (2.16), the network consists of a directed graph with constraints and costs. In contrast to this problem, our aim is to find a tour, which optimizes the KPI $\mathcal{C}_{\mathcal{E}}$. In the literature, one typically refers to the KPI as distance, which leads to respective methods being called shortest path methods. Here, we will not follow this denomination but instead utilize the term of costs per edge.

Remark 2.36
Note that KPIs other than distance, e.g. energy, transportation time and mode, are typically more important for transport and logistic systems. Most of the latter can be transformed into costs, yet also multi-KPI systems are possible but beyond the scope of this lecture.

For the planning process, there are four typical problem we can formulate to optimize the KPI:

- Single-pair shortest path problem: Find the optimal path between an initial vertex r and a terminal vertex s.
- Single-source shortest path problem: Find the optimal paths between an initial vertex r and all other vertexes of the network.
- Single-destination shortest path problem: Find the optimal paths between any vertex of the network and the terminal vertex s.
- All-pairs shortest path problem: Find the optimal paths between any pair of vertexes of the network.

Here, we first focus on the single-source shortest path problem as illustrated in Figure 2.16 connecting the Central Campus and the Airport Campus at TU Braunschweig.

2.2.1 Shortest path problem

The shortest path problem addresses the issue finding a shortest path from a given starting point to any reachable destination. Within this section, we utilize the example network given in Figure 2.17 to illustrate definitions and methods.
To formulate the so called transshipment problem, we first define the so called reachable set, i.e. those vertexes $v \in \mathcal{V}$ which can be reached from the initial one.

Definition 2.37 (Reachable set).
Consider a network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$. For any vertex $v_j \in \mathcal{V}$ we call

$$\mathcal{R}(v_j) := \{v \in \mathcal{V} \mid \text{there exists a path connecting } v_j \text{ and } v\} \tag{2.18}$$

the *reachable set* of v_j.

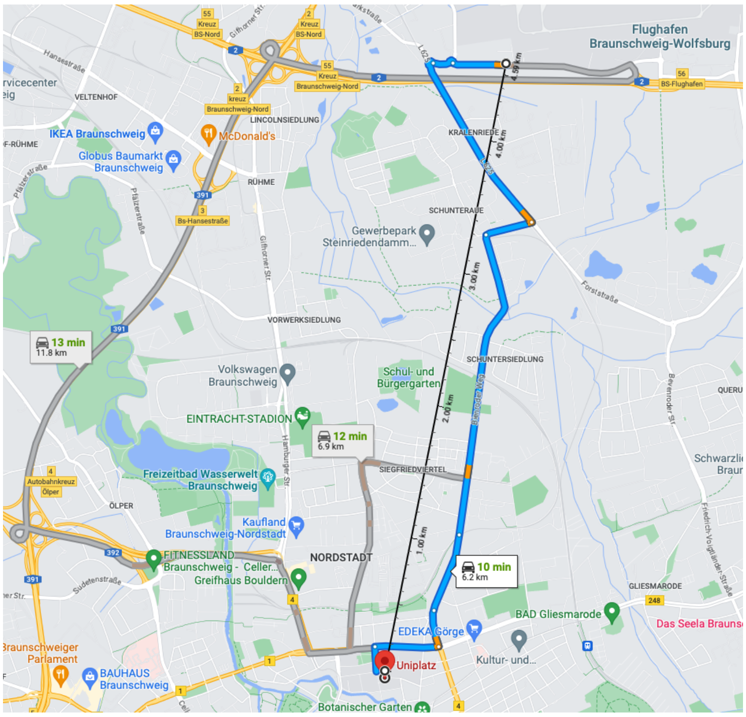


Figure 2.16: Manhattan distance using streets networks at TU Braunschweig

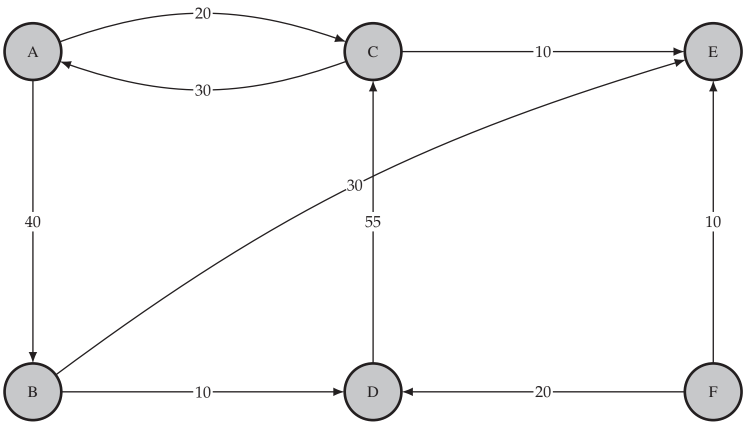


Figure 2.17: Example network with multiplicities

Task 2.38

Compute the reachable set of vertex A from the network in Figure 2.17.

Solution to Task 2.38: From A all vertexes except F can be reached. In fact, we have

$$\begin{aligned} \mathcal{S}(A) &= \{B, C\}, \\ \mathcal{S}(B) &= \{D, E\}, \\ \mathcal{S}(C) &= \{A, E\}, \\ \mathcal{S}(D) &= \{C\}, \text{ and} \\ \mathcal{S}(E) &= \emptyset. \end{aligned}$$

Hence, we obtain

$$\mathcal{R}(A) = \mathcal{S}^{n_\mathcal{E}}(A) = \{B, C, D, E\}.$$

The solution is also highlighted in Figure 2.18.

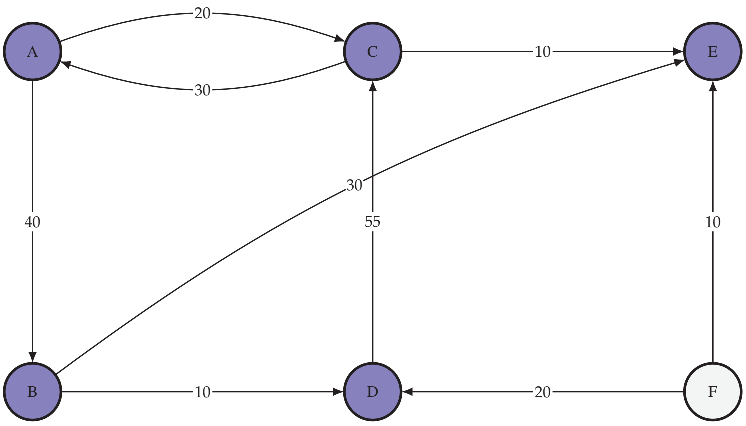


Figure 2.18: Reachable set of vertex A from example network in Figure2.17

We can directly observe that the reachable set is an extension of the successor set $\mathcal{S}$, cf. Definition 2.16, by allowing for paths containing more than one edge. Moreover, if we combine the adjacent edges of all nodes in the reachable set, then by Definition 2.7 the result is a tree:

Theorem 2.39 (Reachable set as tree).
Suppose a network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$ to be given. Then for any vertex $v \in \mathcal{V}$ the adjacent edges of the reachable set $\mathcal{R}(v)$ are a tree.

Task 2.40
Highlight the tree induced by the reachable set $\mathcal{R}(A)$ for the network in Figure 2.17.

Solution to Task 2.40: The solution is highlighted in Figure 2.19.

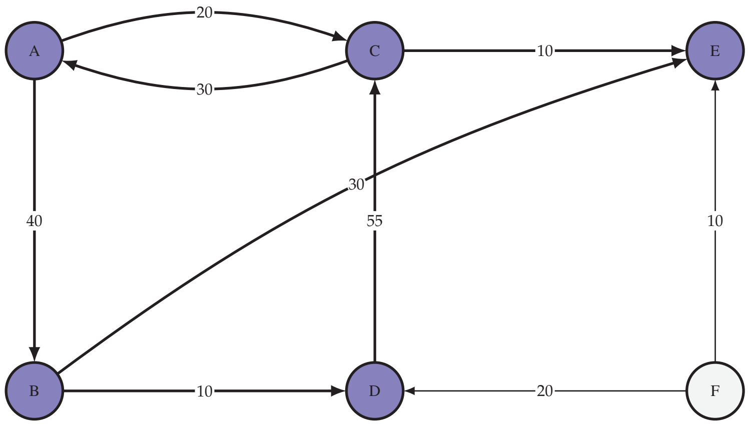


Figure 2.19: Tree induced by reachable set $\mathcal{R}(A)$ from example network in Figure 2.17

Remark 2.41
Note that we are operating on directed graphs. Hence, even if the graph is fully connected, there is no guarantee that each vertex can be reached from any other vertex. As a result, the reachable set may not cover the entirety of $\mathcal{V}$. This is exactly the case for our example network, cf. Figure 2.18 where $F \notin \mathcal{R}(A)$.

Since we want to compute optimal paths starting at $r \in \mathcal{V}$, we have to extend the KPI from a single edge to an entire path.

Definition 2.42 (Minimal path).
Consider a network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$ with multiplicities $\mathcal{C}_{\mathcal{E}} : \mathcal{E} \to \mathbb{R}_0^{n_{\mathcal{E}}}$. Then we call

$$d_{v_j} := \begin{cases} 0, & \text{if } v_j = r \\ d_{v_k} + \mathcal{C}_{\mathcal{E}}(e_{kj}), & \text{if } v_k \in \mathcal{P}(v_j) \\ \infty, & \text{else} \end{cases} \tag{2.19}$$

path value. Moreover, we call

$$d_{v_j} := \min_{v_k \in \mathcal{P}(v_j)} \left(d_{v_k} + \mathcal{C}_{\mathcal{E}}(e_{kj}) \right) \tag{2.20}$$

for all $v_j \in \mathcal{R}(r)$ *minimal path value*.

Task 2.43

Compute the minimal path from vertex A to vertex E for the example network from Figure 2.17.

Solution to Task 2.43: We obtain

$$\begin{aligned}
d_E &= \min_{v_k \in \mathcal{P}(E)} \{d_B + 30, d_C + 10, d_F + 10\}\,, \\
d_B &= \min_{v_k \in \mathcal{P}(B)} \{d_A + 40\} = 40, \\
d_C &= \min_{v_k \in \mathcal{P}(C)} \{d_A + 20, d_D + 10\} = \min_{v_k \in \mathcal{P}(C)} \{20, d_D + 10\}\,, \\
d_D &= \min_{v_k \in \mathcal{P}(D)} \{d_B + 10, d_F + 20\} = \min_{v_k \in \mathcal{P}(D)} \{50, d_F + 20\}\,.
\end{aligned}$$

Hence, we obtain $d_F = \infty$ and therefore $d_D = 50$, $d_C = 20$ and $d_E = 30$. The solution is highlighted in Figure 2.20.

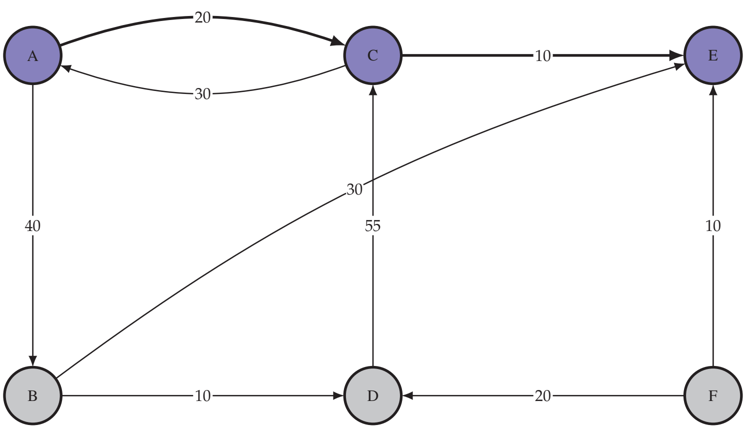


Figure 2.20: Minimal path from vertex A to vertex E for network from Figure 2.17

Note that by construction, the so called Bellman's principle of optimality holds:

Theorem 2.44 (Bellman's principle of optimality).
Consider a network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$ with multiplicities $\mathcal{C}_{\mathcal{E}} : \mathcal{E} \to \mathbb{R}_0^{n_{\mathcal{E}}}$. Furthermore suppose an initial vertex r and a terminal vertex s to be given. If v_j is an element along the minimal path from r to s, then the path from v_j to s is also minimal.

Task 2.45
Argue why any endpiece of an optimal path is again optimal.

Solution to Task 2.45: Suppose the endpiece of an optimal path is not optimal. Then there exists a different path exhibiting an improved path value. Hence the entire path was not optimal contradicting the assumption.

Generically speaking, Bellman's principle states that the tails of optimal solutions are again optimal.

Remark 2.46
Bellman's principle also holds true for very general nonlinear systems and forms the foundation of the so called dynamic programming approach.
For our setting, we additionally obtain that also the starting tails are optimal. This result, however, does not hold true for arbitrary systems.

Unfortunately, the latter approach only allows us to compute a minimal path for one terminal vertex. For planning transport and logistics systems, we are interested to generate minimal paths to all possible vertexes and want such an algorithm to avoid any double computations. This gives us the minimal path reachable set problem:

Definition 2.47 (Transshipment problem).
Suppose a network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$ with multiplicities $\mathcal{C}_{\mathcal{E}} : \mathcal{E} \to \mathbb{R}_0^{n_{\mathcal{E}}}$ and an initial vertex $r \in \mathcal{V}$ to be given. Then we call

$$\min_{v \in \mathcal{R}(r)} \sum_{v \in \mathcal{R}(r)} d_v \tag{2.21}$$

$$\text{such that } d_{v_j} := \begin{cases} 0, & \text{if } v_j = r \\ d_{v_k} + \mathcal{C}_{\mathcal{E}}(e_{kj}), & \text{if } v_k \in \mathcal{P}(v_j) \\ \infty, & \text{else} \end{cases}$$

transshipment problem.

To solve the latter efficiently, we utilize another insight we obtain from Bellman:

Corollary 2.48 (Spanning tree of optimal paths).
Consider a network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$ with multiplicities $\mathcal{C}_{\mathcal{E}} : \mathcal{E} \to \mathbb{R}_0^{n_{\mathcal{E}}}$ and minimal paths from r to be given. Then minimal paths are a spanning tree of the reachable set.

Based on Bellman's principle, we can not only construct one solution in a backwards manner as we did in Task 2.43, but instead apply it in a forward manner to construct a spanning tree of the reachable set. A respective construction algorithm is shown in Algorithm 7.

Algorithm 7 Floyd-Warshall algorithm for minimal paths

Input: Connected network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$
Input: Multiplicities $\mathcal{C}_{\mathcal{E}} : \mathcal{E} \to \mathbb{R}_0^{n_{\mathcal{E}}}$
Input: Initial vertex $r \in \mathcal{V}$

procedure CLASS TREE ALGORITHM($\mathcal{N}, \mathcal{C}_{\mathcal{E}}, r$)
    $d_r \leftarrow 0$, $p_r \leftarrow 0$, $Q \leftarrow \{r\}$
    for all $j \in \{1, \ldots, n_{\mathcal{V}}\} \setminus \{r\}$ **do**
        $d_{v_j} \leftarrow \infty$, $p_{v_j} \leftarrow \infty$
    end for
    while $Q \neq \emptyset$ **do**
        Select v_j from Q ▷ Select arbitrary end of minimal path
        $Q \leftarrow Q \setminus \{v_j\}$
        for all $v_k \in \mathcal{S}(v_j)$ **do**
            if $d_{v_k} > d_{v_j} + \mathcal{C}_{\mathcal{E}}(e_{jk})$ **then** ▷ Check for improvement of successor vertex
                $d_{v_k} \leftarrow d_{v_j} + \mathcal{C}_{\mathcal{E}}(e_{jk})$
                $p_{v_k} \leftarrow v_j$ ▷ Label path predecessor
                if $v_k \notin Q$ **then**
                    $Q \leftarrow Q \cup \{v_k\}$
                end if
            end if
        end for
    end while
end procedure

Output: Minimal path values d_v for all $v \in \mathcal{R}(r)$
Output: Path sequences p_v for all $v \in \mathcal{R}(r)$

For these tree algorithms, there exist two possible technical outcomes, the so called label-setting and the label-correcting methods.

Definition 2.49 (Label setting and label correcting).
The Tree Algorithm 7 is called *label setting* if any vertex $v \in \mathcal{V}$ is added to the queue Q only once. If any vertex is added more than once, the algorithm is called *label correcting*.

The major difference regarding setting and correcting occurs in line 7 of Algorithm 7 where the next candidate vertex is selected. If an arbitrary vertex is selected, we cannot guarantee that the vertex will not reenter the queue Q. However, we may utilize the following assumption:

Assumption 2.50 (Nonnegative multiplicities)
The multiplicities $\mathcal{C}_{\mathcal{E}} : \mathcal{E} \to \mathbb{R}_0^{n_{\mathcal{E}}}$ satisfy

$$\mathcal{C}_{\mathcal{E}}(e) \geq 0 \tag{2.22}$$

for all $e \in \mathcal{E}$.

Based on the latter assumption, we can apply a greedy heuristic and simply consider that vertex in the queue Q which exhibits the lowest minimal path value. This greedy idea leads to the so called Dijkstra Algorithm 8.

Remark 2.51
Note that due to the nonnegativity assumption (2.22)*, we obtain that the vertex will never reenter the queue Q. Again, the argumentation is built on a contradiction assumption: Suppose a vertex will reenter the queue, then its value must be reduced before reentering. As it is already the minimal value in the queue and all multiplicities that can be applied are positive, it can only increase. Hence, this case cannot occur.*

Task 2.52
Apply Dijkstra's algorithm to the example network from Figure 2.17.

Solution to Task 2.52: We obtain the steps given in Table 2.3. The solution is visualized in Figure 2.21.

Algorithm 8 Dijkstra's algorithm for minimal paths

Input: Connected network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$
Input: Multiplicities $\mathcal{C}_{\mathcal{E}} : \mathcal{E} \to \mathbb{R}_0^{n_{\mathcal{E}}}$
Input: Initial vertex $r \in \mathcal{V}$

procedure CLASS DIJKSTRA($\mathcal{N}, \mathcal{C}_{\mathcal{E}}, r$)
  $d_r \leftarrow 0$, $p_r \leftarrow 0$, $Q \leftarrow \{r\}$
  for all $j \in \{1, \ldots, n_{\mathcal{V}}\} \setminus \{r\}$ **do**
    $d_{v_j} \leftarrow \infty$, $p_{v_j} \leftarrow \infty$
  end for
  while $Q \neq \emptyset$ **do**
    $v_j \leftarrow argmin_{v_j \in Q}\, d_{v_j}$ ▷ Select end of minimal path
    $Q \leftarrow Q \setminus \{v_j\}$
    for all $v_k \in \mathcal{S}(v_j)$ **do**
      if $d_{v_k} > d_{v_j} + \mathcal{C}_{\mathcal{E}}(e_{jk})$ **then** ▷ Check for improvement of successor vertex
        $d_{v_k} \leftarrow d_{v_j} + \mathcal{C}_{\mathcal{E}}(e_{jk})$
        $p_{v_k} \leftarrow v_j$ ▷ Label path predecessor
        if $v_k \notin Q$ **then**
          $Q \leftarrow Q \cup \{v_k\}$
        end if
      end if
    end for
  end while
end procedure

Output: Minimal path values d_v for all $v \in \mathcal{R}(r)$
Output: Path sequences p_v for all $v \in \mathcal{R}(r)$

Table 2.3: Dijkstra table for example from Figure 2.17

Iteration	1		2		3		4		5		6	
Queue Q	$\{A\}$		$\{B, C\}$		$\{B, E\}$		$\{B\}$		$\{D\}$		$\emptyset$	
Vertex j	d_{v_j}	p_{v_j}	d_{v_j}	p_{v_j}	d_{v_j}	p_{v_j}	d_{v_j}	p_{v_j}	d_{v_j}	p_{v_j}	d_{v_j}	p_{v_j}
A	0											
B	∞		40	A								
C	∞		20	A								
D	∞								50	B		
E	∞				30	C						

Continued on next page

Table 2.3 – continued from previous page

Iteration	1		2		3		4		5		6	
Queue Q	$\{A\}$		$\{B,C\}$		$\{B,E\}$		$\{B\}$		$\{D\}$		$\emptyset$	
Vertex j	d_{v_j}	p_{v_j}	d_{v_j}	p_{v_j}	d_{v_j}	p_{v_j}	d_{v_j}	p_{v_j}	d_{v_j}	p_{v_j}	d_{v_j}	p_{v_j}
F	∞											

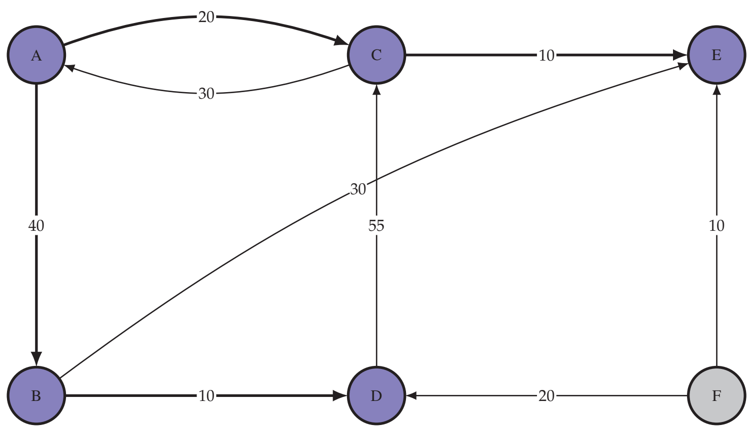


Figure 2.21: Minimal paths and path sequences for $\mathcal{R}(A)$ for network from Figure 2.17

Remark 2.53

Dijkstra's algorithm can be extended to the so called Bellman-Dijkstra algorithm. For this version, Assumption 2.50 is replaced by a cycle-free assumption. This version is beyond the scope of the lecture and can be found, e.g., in [7].

Table 2.4: Advantages and disadvantages of the Dijkstra algorithm

Advantage	Disadvantage
✓ Computes reachable set	✗ Requires symmetry
✓ Derives shortest paths	✗ Neglects constraints

Similar to the cost minimal flow problem (2.16) the transshipment problem problem (2.21) can be reformulated using the linear model for directed graphs. To this end, we denominate minimal path value for vertex v_j by $\mathbf{x}_j$ and design one constraint for each multiplicity of an edge revealing

$$\max \sum_{j=1}^{n_{\mathcal{V}}} \mathbf{x}_j \tag{2.23}$$

$$\text{such that } \mathbf{x}_k - \mathbf{x}_j \leq \mathcal{C}_{\mathcal{E}}(e_{jk}) \qquad \forall \underline{e}_{jk} \in \mathcal{E} \tag{2.24}$$

$$\mathbf{x}_r = 0 \tag{2.25}$$

$$\mathbf{x}_j \geq 0 \qquad \forall j \in \{1, \ldots, n_{\mathcal{V}}\} \tag{2.26}$$

Again, the latter problem can be addressed using the simplex method (or more accurately a network version of it), which is outside the scope of this lecture.

2.2.2 Vehicle routing problem

Different from the shortest path problem, the vehicle routing problem does not only consider the shortest paths to destinations, but aims to design a route for a vehicle/utility. As it addresses more than one destination but still requires point-to-point operation, it is an extension of the single-source shortest path problem. Here, a route always starts and ends at the same vertex. Vehicle routing problems may be found in various fields of transport and logistics, i.e. in the delivery of goods to final destinations or the collection of goods from destinations, hence in both forward and reverse logistics. In both cases, it is often called milk run and requires to identify which destinations should be combined to form a tour, cf. Figure 2.22.

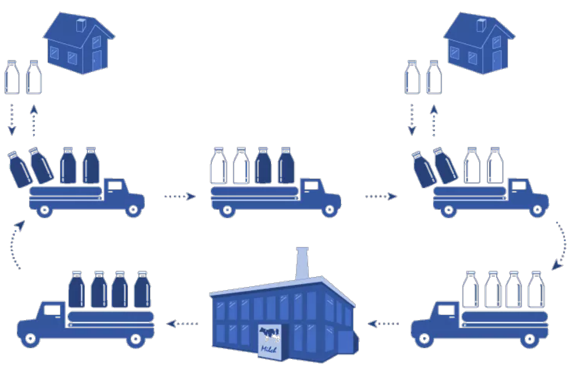


Figure 2.22: Milk run in logistics[2]

[2]Source: https://www.hellmann-east-europe.com

To introduce the problem formally, we need to define what we mean by a tour. The core of a tour is the so called depot, i.e. the start and ending point.

Definition 2.54 (Depot).
Given a network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$ a depot is a fixed vertex $v_0 \in \mathcal{V}$.

To illustrate and accompany the introduced terms, we consider the example given in Figure 2.23. To separate the depot from other vertexes, we utilize a second layer in Figure 2.23.

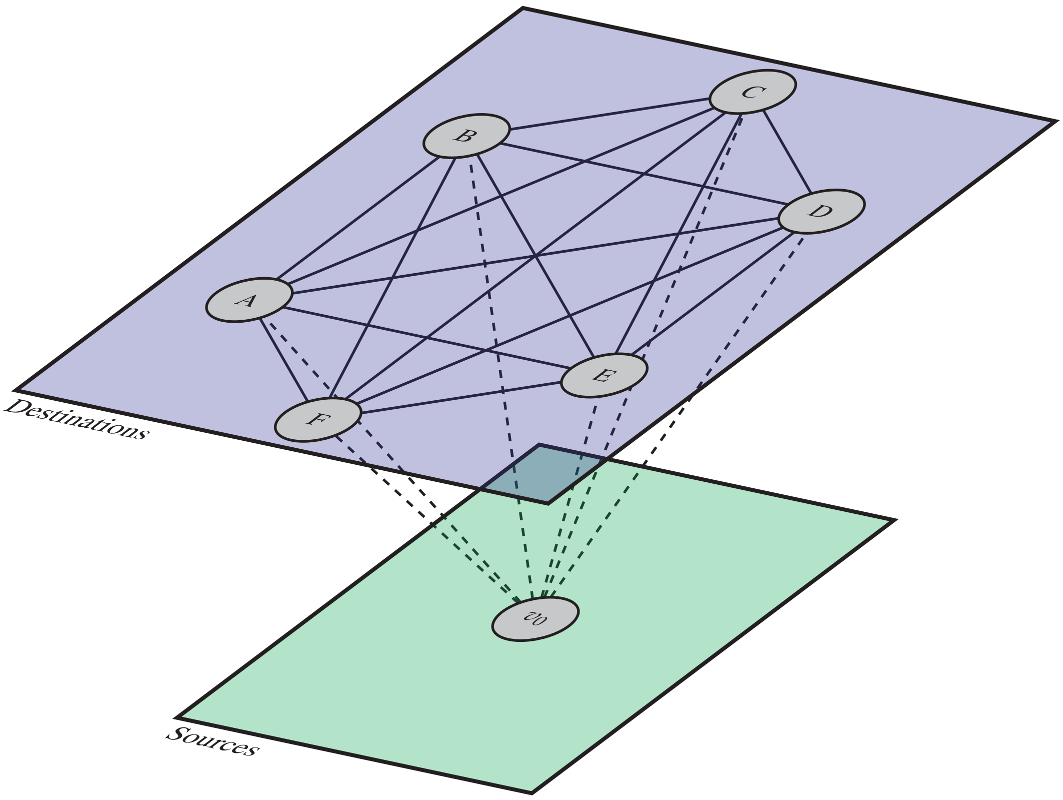


Figure 2.23: Example of a vehicle routing problem

Based on the depot, we introduce the concept of a tour.

Definition 2.55 (Tour).
Consider a network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$ and suppose $v_0 \in \mathcal{V}$ to be depot. Then the set of vertexes $\mathcal{T} := \{v_0\} \cup \{v_j\}_{j \in \mathcal{I}} \subset \mathcal{V}$ that may be connected via a path is called a *tour*.

Note that a tour does not state in which sequence vertexes occur within a path, only that these vertexes shall be contained in one path. To derive the sequence, we build up on the shortest path concept from the previous section and suppose that each edge is associated with a certain cost. Within this section, we adopt the following standard assumption:

Assumption 2.56 (Symmetry)
Given a network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$ with multiplicity $\mathcal{C}_{\mathcal{E}} : \mathcal{E} \to \mathbb{R}_0^{n_\mathcal{E}}$ we have that

$$\forall e_{jk} = (v_j, v_k) \in \mathcal{E} : \ \exists e_{kj} = (v_k, v_j) \in \mathcal{E} \tag{2.27}$$

and the multiplicities satisfy

$$\mathcal{C}_{\mathcal{E}}(e_{jk}) = \mathcal{C}_{\mathcal{E}}(e_{kj}). \tag{2.28}$$

Mapped into reality, Assumption 2.56 requires all edges to be bidirectionally usable (equation (2.27)) and the respective costs to be identical (equation (2.28)).

Remark 2.57
Assumption 2.56 is a purely simplifying and may be disregarded for the upcoming concept.

For our example problem, these multiplicities/KPI costs per edge are given in Table 2.5.

Table 2.5: Distance table for example from Figure 2.23

Vertex	v_0	A	B	C	D	E	F	$\mathcal{C}_{\mathcal{V}}(v)$
v_0	–	20	30	30	20	50	35	
A		–	30	45	35	65	45	5
B			–	30	45	75	55	2
C				–	35	70	60	5
D					–	35	25	8
E						–	25	4
F							–	6

Remark 2.58
Due to symmetry, the lower left quadrant of the cost matrix in Table 2.5 is identical to the upper right and therefore left out.

Utilizing the multiplicities, we can introduce an order to assess tours. Each of such candidates is called a route:

Definition 2.59 (Route).
Consider a network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$ and suppose $v_0 \in \mathcal{V}$ to be depot and $\mathcal{T} \subset \mathcal{V}$ to be a tour. Then any path with initial and terminal vertex v_0 and intermediate vertexes $v_j \in \mathcal{T}$ is called a *route* $\mathcal{R} \subset \mathcal{E}$.

Remark 2.60
We like to stress that tours are sets of vertexes whereas routes are sets of edges.

In practice, the utilities servicing a route are limited regarding their capacity of load units. At the same time, each vertex requires a defined number of load units, which we can model using markings $\mathcal{C}_{\mathcal{V}} : \mathcal{V} \to \mathbb{R}_0^{n_{\mathcal{V}}}$. Combining markings, capacity of utilities and route gives us route the capacity constraint.

Definition 2.61 (Capacity constraint).
Suppose a network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$ with marking $\mathcal{C}_{\mathcal{V}} : \mathcal{V} \to \mathbb{R}_0^{n_{\mathcal{V}}}$ and a route $\mathcal{R} \subset \mathcal{E}$ to be given. Furthermore suppose a utility to exhibit the maximal capacity $C \in \mathbb{R}^+$. Then the inequality

$$\sum_{e=(v_j,v_k)\in\mathcal{R}} \mathcal{C}_{\mathcal{V}}(v_j) \leq C \tag{2.29}$$

is called *capacity constraint*.

Now the aim of the vehicle routing problem is to tackle four issues at the same time, that is

1. to calculate how tours shall be defined and
2. which utility shall be matched to which tour,
3. in which sequence the elements of tours shall be brought to form a route and
4. how an efficient plan considering the KPI can be obtained.

In contrast to the cost minimal maximal flow problem we discussed on the strategic level, here all problems exhibit the same nature of minimizing cost. Hence, no two level problem arises.

Definition 2.62 (Capacitated vehicle routing problem).
Consider a network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$ with marking $\mathcal{C}_{\mathcal{V}} : \mathcal{V} \to \mathbb{R}_0^{n_\mathcal{V}}$ and multiplicity $\mathcal{C}_{\mathcal{E}} : \mathcal{E} \to \mathbb{R}_0^{n_\mathcal{E}}$ as well as capacities of utilities C_j, $j = 1, \ldots, n_u$ to be given where $n_u \in \mathbb{N}$ is the the number of utilities. Then we call

$$\min_{\mathcal{R}_j} \sum_{j=1}^{n_u} \sum_{e \in \mathcal{R}_j} \mathcal{C}_{\mathcal{E}}(e) \tag{2.30}$$

$$\begin{aligned} \text{such that } \mathcal{R}_j \text{ is a route} \quad & \forall j \in \{1, \ldots, n_u\} \\ \bigcup_{j=1}^{n_u} \{v | v \in \mathcal{R}_j\} &= \mathcal{V} \\ \mathcal{R}_j \cap \mathcal{R}_k = \emptyset \quad & \forall j,k \in \{1, \ldots, n_u\} \text{ with } j \neq k \\ \sum_{e=(v_k,v_l) \in \mathcal{R}_j} \mathcal{C}_{\mathcal{V}}(v_k) \leq C \quad & \forall j \in \{1, \ldots, n_u\} \end{aligned}$$

capacitated vehicle routing problem. The minimizing set of routes is called *routing plan*.

Remark 2.63
There are many extension of the capacitated vehicle routing problem that can be found in theory and practice. These include, among others,

- *heterogeneous utilities,*
- *time windows for delivery/collection,*
- *simultaneous pickup-and-delivery,*
- *preferred right turn,*
- *minimal energy and*
- *working hour limitations of drivers as well as driving and resting periods.*

The capacity vehicle routing problem can be solved using the branch-and-bound method. The downside of this method, however, is its complexity which leads to long runtimes. An alternative to branch-and-bound as deterministic approach, a heuristic solution approach can be used.

Remark 2.64
In practice, heuristics are quite common. Reasons for using heuristics are – apart from reduced complexity – that problems are simplifications with mostly not exact parameters. For such problems near optimal but quickly avaialbe solutions are sufficient and allow for a quicker reaction.

Applying heuristics, we first need to identify how many utilities may be needed. To get an educated guess, the so called Bin Packing Algorithm 9 can be used.

Algorithm 9 Bin packing algorithm

Input: Connected network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$
Input: Markings $\mathcal{C}_{\mathcal{V}} : \mathcal{V} \to \mathbb{R}_0^{n_{\mathcal{V}}}$
Input: Capacity of utilities C
procedure CLASS BIN PACKING($\mathcal{N}, \mathcal{C}_{\mathcal{V}}, C$)
  $Q \leftarrow \mathcal{V}$, $n_u \leftarrow 1$, $C(\mathcal{T}) = 0$
  while $Q \neq \emptyset$ **do**
    $v \leftarrow \underset{v \in Q}{argmax}\, \mathcal{C}_{\mathcal{V}}(v)$
    $Q \leftarrow Q \setminus \{v\}$
    if $C_{n_u} + \mathcal{C}_{\mathcal{V}}(v) \leq C$ **then**
      $\mathcal{T}_{n_u} \leftarrow \mathcal{T}_{n_u} \cup \{v\}$
      $C_{n_u} \leftarrow C_{n_u} + \mathcal{C}_{\mathcal{V}}(v)$
    else
      $n_u \leftarrow n_u + 1$, $\mathcal{T}_{n_u} \leftarrow \{v\}$
      $C_{n_u} \leftarrow \mathcal{C}_{\mathcal{V}}(v)$
    end if
  end while
end procedure
Output: Number of utilities $n_u \in \mathbb{N}$
Output: Used capacities per utility C_{n_u}
Output: Tours $\mathcal{T}_{n_u}$

Task 2.65
Apply the bin packing algorithm to our example problem from Figure 2.23 with $C = 10$.

Solution to Task 2.65: From the markings we obtain the sequence order indicated in Figure 2.24

$$\mathcal{C}_{\mathcal{V}}(D) \geq \mathcal{C}_{\mathcal{V}}(F) \geq \mathcal{C}_{\mathcal{V}}(A) \geq \mathcal{C}_{\mathcal{V}}(C) \geq \mathcal{C}_{\mathcal{V}}(E) \geq \mathcal{C}_{\mathcal{V}}(B).$$

We start by inserting D into utility 1.
Since $\mathcal{C}_{\mathcal{V}}(D) + \mathcal{C}_{\mathcal{V}}(F) > C$, we add F to utility 2.
Again we have $\mathcal{C}_{\mathcal{V}}(F) + \mathcal{C}_{\mathcal{V}}(A) \geq C$ and hence add A to utility 3.
Now we have $\mathcal{C}_{\mathcal{V}}(A) + \mathcal{C}_{\mathcal{V}}(C) = C$ and add C to utility 3.
Since $\mathcal{C}_{\mathcal{V}}(A) + \mathcal{C}_{\mathcal{V}}(C) + \mathcal{C}_{\mathcal{V}}(E) > C$, we add E to utility 4.
Last, we have $\mathcal{C}_{\mathcal{V}}(E) + \mathcal{C}_{\mathcal{V}}(B) \leq C$ and add B to utility 4.

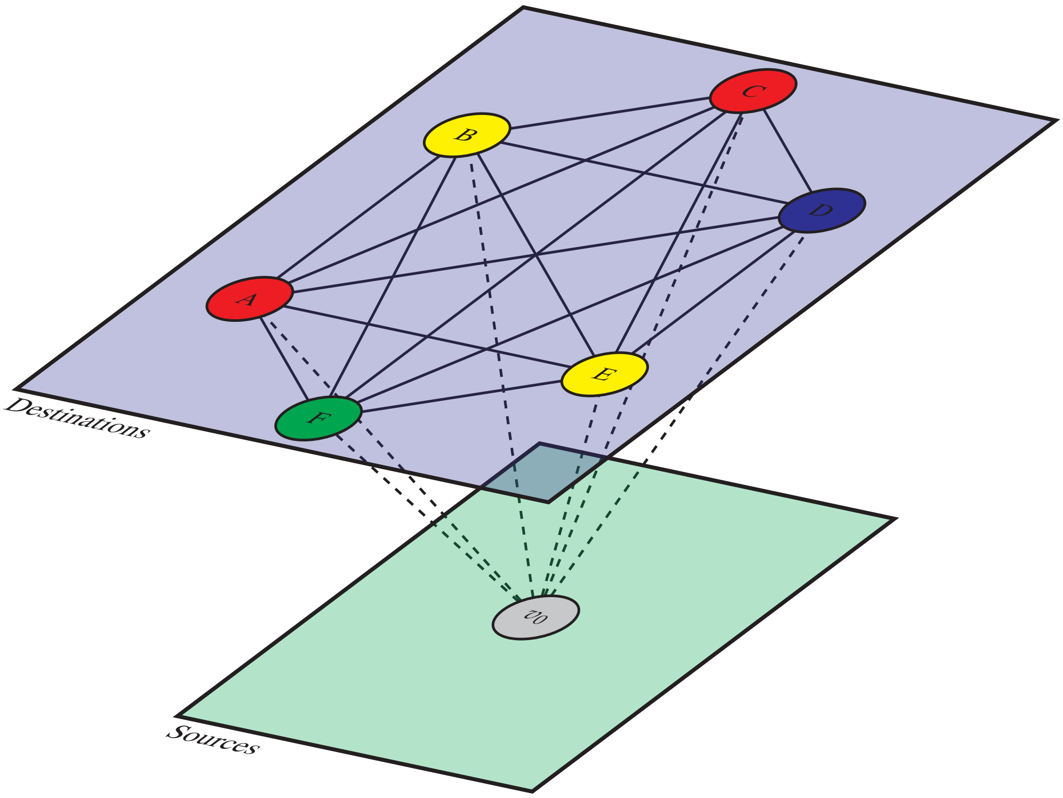


Figure 2.24: Bin packing for example from Figure 2.23

While the Algorithm 9 additionally reveals tours, these tours are purely based on the markings and not on the multiplicities used for minimization. To address multiplicities, the so called nearest neighbor Algorithm 10 can be applied to generate a route based on a given tour.
The nearest neighbor algorithm is operates on the greedy heuristic that utilizes the end vertex of a route and adds that edge to the route which exhibits the least additional costs.

Task 2.66
Given the tours from Task 2.65 use Algorithm 10 to derive respective routes.

Algorithm 10 Nearest neighbor algorithm

Input: Connected network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$
Input: Multiplicities $\mathcal{C}_{\mathcal{E}} : \mathcal{E} \to \mathbb{R}_0^{n_{\mathcal{E}}}$
Input: Tour $\mathcal{T}$ and depot $v_0 \in \mathcal{V}$

procedure CLASS NEAREST NEIGHBOR($\mathcal{N}, \mathcal{C}_{\mathcal{E}}, \mathcal{T}$)
    for $j = 1, \ldots, \sharp\mathcal{R}$ **do**
        $v \leftarrow \underset{v \in \mathcal{S}(v_{j-1}) \cap \mathcal{T}}{argmin} \; \mathcal{C}_{\mathcal{E}}((v_{j-1}, v))$
        $\mathcal{T} \leftarrow \mathcal{T} \setminus \{v\}$
        $\mathcal{R} \leftarrow \mathcal{R} \cup \{(v_{j-1}, v)\}$
    end for
    $\mathcal{R} \leftarrow \mathcal{R} \cup \{(v, v_0)\}$
end procedure
Output: Route $\mathcal{R}$

Solution to Task 2.66: We directly obtain

$$\begin{aligned} &\{(v_0, D), (D, v_0)\} && \text{for utility 1} \\ &\{(v_0, F), (F, v_0)\} && \text{for utility 2.} \end{aligned}$$

For utility 3, we have the tour (A, C). Since $\mathcal{C}_{\mathcal{E}}((v_0, A)) < \mathcal{C}_{\mathcal{E}}((v_0, C))$ we directly obtain the route

$$\{(v_0, A), (A, C), (C, v_0)\} \qquad \text{for utility 3.}$$

Similarly, for utility 4 we have the tour (B, E) and observe $\mathcal{C}_{\mathcal{E}}((v_0, B)) < \mathcal{C}_{\mathcal{E}}((v_0, E))$ to conclude

$$\{(v_0, B), (B, E), (E, v_0)\} \qquad \text{for utility 4.}$$

The result is sketched in Figure 2.25 and reveals the cost

$$J(\mathcal{R}) = \underbrace{20 + 45 + 30}_{\text{utility 3}} + \underbrace{30 + 75 + 50}_{\text{utility 4}} + \underbrace{20 + 20}_{\text{utility 1}} + \underbrace{35 + 35}_{\text{utility 2}} = 360.$$

Remark 2.67
Alternatively to nearest neighbor, successive insertion may be used. Here, the greedy heuristic utilizes the vertex, for which the minimal cost of a connecting edge is maximal and inserts the

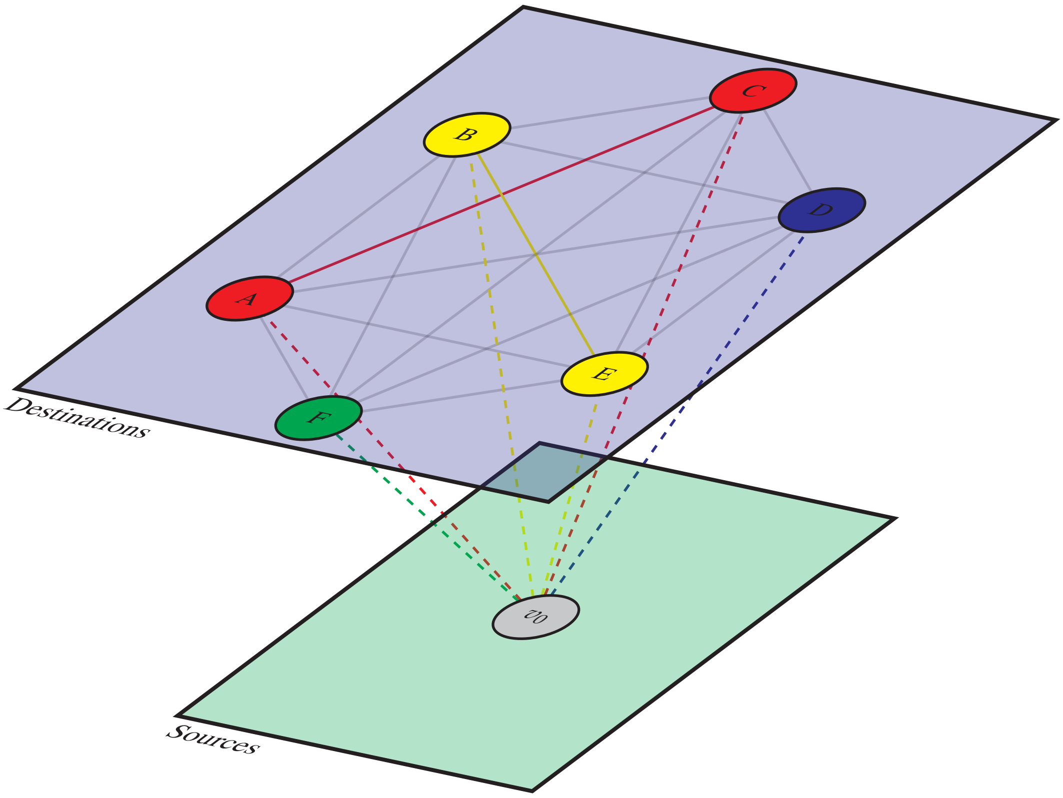


Figure 2.25: Nearest neighbor based on bin packing for example from Figure 2.23

respective minimal edge.

From Task 2.66, we observe that the tours computed by the bin packing algorithm are not optimal. In fact, there are two different ways to improve such a solution:

- Swap sequence of vertexes within a tour (neighborhood search)
- Unite tours (savings search)

Here, we discuss the so called Savings Algorithm 11. The idea of the algorithm is to start with one route per vertex. Then, routes are united using the greedy heuristic of maximal savings of return trips while maintaining the capacity constraint.

Remark 2.68
Note that Algorithm 11 in the displayed form does not assume an initial route to be given. Yet it can be applied to given routes as well by removing the first ForAll-loop.

Algorithm 11 Savings algorithm

Input: Connected network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$
Input: Multiplicities $\mathcal{C}_\mathcal{E} : \mathcal{E} \to \mathbb{R}_0^{n_\mathcal{E}}$
Input: Markings $\mathcal{C}_\mathcal{V} : \mathcal{V} \to \mathbb{R}_0^{n_\mathcal{V}}$
Input: Capacity of utilities C

```
procedure CLASS SAVINGS(𝒩, 𝒞_ℰ, 𝒞_𝒱, C)
    for all e_jk = (v_j, v_k) ∈ ℰ do
        s_jk ← 𝒞_ℰ(e_j0) + 𝒞_ℰ(e_0k) − 𝒞_ℰ(e_jk)
        if s_jk > 0 then
            𝒮 ← 𝒮 ∪ {s_jk}
        end if
    end for
    for all j = 1, ..., n_𝒱 do
        ℛ_j ← {e_0j, e_j0}, C_j ← 𝒞_𝒱(v_j)
    end for
    while S ≠ ∅ do
        e ← argmax_{e_jk ∈ ℰ} s_jk, 𝒮 ← 𝒮 \ {e}
        if 𝒞_𝒱(v_j) + 𝒞_𝒱(v_k) ≤ C then
            ℛ_j ← ℛ_j ∪ {e_jk} ∪ ℛ_k \ {e_j0, e_0k}
            Delete ℛ_k
        end if
    end while
end procedure
```

Output: Routes $\mathcal{R}_j$, $j = 1, \ldots, n_u$

Task 2.69

Apply the savings algorithm to example problem from Figure 2.23 with $C = 10$.

Solution to Task 2.69: From Table 2.5 we obtain the savings

$$[s_{jk}] = \begin{pmatrix} - & 20+30-30 & 20+30-45 & 20+20-35 & 20+50-65 & 20+35-45 \\ & - & 30+30-30 & 30+20-45 & 30+50-75 & 30+35-55 \\ & & - & 30+20-35 & 30+50-70 & 30+35-60 \\ & & & - & 20+50-35 & 20+35-25 \\ & & & & - & 50+35-25 \\ & & & & & - \end{pmatrix}$$

$$= \begin{pmatrix} - & 20 & 5 & 5 & 5 & 10 \\ & - & 30 & 5 & 5 & 10 \\ & & - & 15 & 10 & 5 \\ & & & - & 35 & 30 \\ & & & & - & 60 \\ & & & & & - \end{pmatrix}$$

Then we identify the maximum for s_{EF}. Since the markings reveal required capacities $\mathcal{C}_{\mathcal{V}}(EF) = \mathcal{C}_{\mathcal{V}}(E) + \mathcal{C}_{\mathcal{V}}(F) = 4 + 6 = 10 \leq C$ we can unite the routes.
The next maximum is given by s_{DE}. Yet we have $\mathcal{C}_{\mathcal{V}}(D) + \mathcal{C}_{\mathcal{V}}(EF) = 8 + 10 = 18 > C$ and cannot combine the routes. The same holds for s_{DF}.
Following, we consider s_{BC} and see $\mathcal{C}_{\mathcal{V}}(BC) = \mathcal{C}_{\mathcal{V}}(B) + \mathcal{C}_{\mathcal{V}}(C) = 5 + 2 = 7 \leq C$, which allows us to unite the routes.
Next we consider s_{AB} which gives us $\mathcal{C}_{\mathcal{V}}(ABC) = \mathcal{C}_{\mathcal{V}}(A) + \mathcal{C}_{\mathcal{V}}(BC) = 5 + 7 = 12 > C$ and we cannot combine the routes.
Based on the algorithm, we would have to continue with all positive combinations. Based on the markings, however, we can already state that no further unions are possible. Figure 2.26 shows the result. The respective costs sum up to

$$J(\mathcal{R}) = \underbrace{20 + 20}_{\text{utility 1}} + \underbrace{30 + 30 + 30}_{\text{utility 2}} + \underbrace{20 + 20}_{\text{utility 3}} + \underbrace{50 + 25 + 35}_{\text{utility 4}} = 280,$$

which is a clear improvement over the bin packing / nearest neighbor approach.

Remark 2.70
As an alternative to the saving algorithm the so called sweep algorithm can be used. Instead of uniting routes using saved costs, the sweep utilizes a geometric approach. The idea is to unite routes via a (counter-)clockwise logic, i.e. neighboring routes are combined. The approach, however, requires the vertexes to be positioned as in carthesian coordinates and the costs to be defined via distances.

Different from the savings algorithm, the following so called 2-opt Algorithm 12 aims to identify improvements within a route, i.e. not between routes. To this end, the algorithm stochastically crawls routes to find possible improvements.

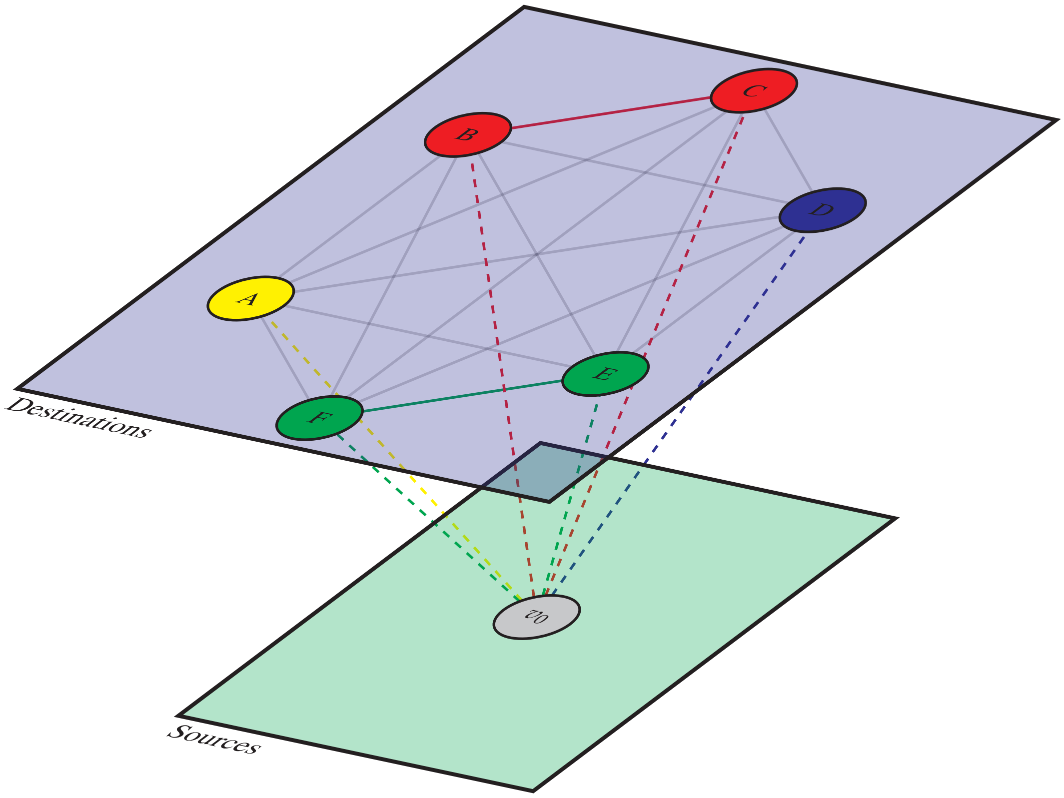


Figure 2.26: Result of savings algorithm for example from Figure 2.23

Algorithm 12 2-opt algorithm

Input: Connected network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$
Input: Multiplicities $\mathcal{C}_{\mathcal{E}} : \mathcal{E} \to \mathbb{R}_0^{n_{\mathcal{E}}}$
Input: Route $\mathcal{R}$

procedure Class 2-opt($\mathcal{N}, \mathcal{C}_{\mathcal{E}}, \mathcal{R}$)
  Select $e_{jj+1}, e_{kk+1} \in \mathcal{R}_l$
  if $\mathcal{C}_{\mathcal{E}}(e_{jj+1}) + \mathcal{C}_{\mathcal{E}}(e_{kk+1}) > \mathcal{C}_{\mathcal{E}}(e_{jk}) + \mathcal{C}_{\mathcal{E}}(e_{j+1k+1})$ **then**
    $\mathcal{R}_l \leftarrow \mathcal{R}_l \setminus \{e_{jj+1}, e_{kk+1}\} \cup \{e_{jk}, e_{j+1k+1}\}$
  end if
end procedure
Output: Route $\mathcal{R}$

Task 2.71

Utilize the 2-opt algorithm to improve the solution obtained for Task 2.69.

Solution to Task 2.71: For our example from Figure 2.23 it does not make sense to apply the 2-opt improvement. The reason is that 2-opt at minimum requires 5 vertexes (including the depot) within a route to be applicable.

To conclude this section, we can combine the above mentioned algorithms to obtain a heuristic solution of the capacitated vehicle routing problem (2.30).

Algorithm 13 Heuristic for capacitated vehicle routing problem

Input: Connected network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$
Input: Multiplicities $\mathcal{C}_\mathcal{E} : \mathcal{E} \to \mathbb{R}_0^{n_\mathcal{E}}$
Input: Markings $\mathcal{C}_\mathcal{V} : \mathcal{V} \to \mathbb{R}_0^{n_\mathcal{V}}$
Input: Capacity of utilities C

```
1: procedure CLASS HEURISTIC CVRP(N, C_E, C_V, C, R)
2:     (R, n_u) ← CLASS SAVINGS(N, C_E, C_V, C)
3:     for j = 1, ..., n_u do
4:         T_j ← CLASS 2-OPT(N, C_E, R_j)
5:     end for
6: end procedure
```

Output: Number of utilities $n_u \in \mathbb{N}$
Output: Routes $\mathcal{R}_j$, $j = 1, \dots, n_u$

Table 2.6: Advantages and disadvantages of heuristics for CVRP

Advantage	Disadvantage
✓ Includes costs and constraints	✗ Computes suboptimal solution
✓ Computes usage of utilities	✗ Neglects robustness
✓ Matches routes and utilities	✗ Disregards structure

Remark 2.72
Algorithm 13 is only an example, where we applied the choice of the Savings Algorithm 11 for designing tours and initializing/uniting routes. Alternatively, the Bin packing Algorithm 9 can be used to design tours and Nearest Neighbor Algorithm 10 for initializing/uniting routes. Regarding a neighborhood search for improving the routes we imposed the 2-opt Algorithm 12.

Having established basic methods for strategic design and tactical planning from an operations research perspective, in the following chapter we concentrate on learning patterns for people operating, learning, analyzing and improving intermodal transport and logistic systems on a large scale.

CHAPTER 3

COORDINATION

> In structural gamification the content does not become game-like: only the structure around the content does.
>
> Karl Kapp

In the previous chapter, we discussed several problems arising in transport and logistic systems with a particular focus on deriving an optimal solution or a characteristical insight. In practical applications, the required conditions are rarely fulfilled and the provided models are typically not very good. Hence, the solutions/insights may serve as benchmark but in many cases require different ideas regarding implementation.

In the present chapter, we focus on ideas to abstract from optimality but include the complex and distributed nature of transport and logistics systems. The first approach outlined in Section 3.1 utilizes gamification, that is the idea to formulate a problem as a game and cherish the anticipativity of human interaction. While typically not optimal, such an approach allows to train personal and question existing processes. In the following Section 3.2, we discuss the idea of a leader follower principle, which allows us to split the problem into subproblems and introduce a coordination structure.

3.1 Serious gaming

One of the most basic approaches to design or improve intermodal transport and logistic processes is called serious gaming. Instead of „truely optimizing“ the system, the approach applies interactive and immersive gaming experiences to address challenges, train professionals, optimize operations, and enhance decision-making processes. As such, it addresses

- training and education for individuals to improve understanding,
- process optimization for companies to design improved processes,
- collaboration and coordination for stakeholders,
- risk management for individuals to enhance preparedness, and
- sustainability practices to highlight ecologic impacts.

Here, we particularly focus on the first and second point.
In the context of transport and logistics processes, we formulate a serious game as an approach to find solutions for problems such as our minimal spanning tree problem (Definition 2.10), the max flow problem (Definition 2.25), the transshipment problem (Definition 2.47), the spanning tree problem (Definition 2.48) or the capacitated vehicle routing problem (Definition 2.62). In order to be applicable, we require that for any user input the respective solution can be evaluated, i.e. simulation.

Definition 3.1 (Serious game).
Consider a network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$ with marking $\mathcal{C}_{\mathcal{V}} : \mathcal{V} \to \mathbb{R}_0^{n_{\mathcal{V}}}$ and multiplicity $\mathcal{C}_{\mathcal{E}} : \mathcal{E} \to \mathbb{R}_0^{n_{\mathcal{E}}}$ and possible further constraints regarding infrastructure and utilities $C : \mathcal{V} \times \mathcal{E} \to \mathbb{R}_0^{n_{\mathcal{E}}}$. Furthermore suppose a method to complement inputs to a solution and evaluate the respective output of the latter to be given. Then we call an approach of iteratively defining inputs by a user a serious game.

The intent of a serious game is to utilize the learning capability of users, cf. Figure 3.1 regarding the conceptional context.
From Definition 3.1 we directly obtain that a respective method is not a solution method as we discussed so far, but instead requires inputs from a user. More formally, we can describe the latter in Algorithm 14.
As described in Algorithm 14 and Definition 3.1 a serious game is of iterative nature. In this context, the iterations are seen as time whereas the goal of serious gaming is to derive an optimal solution via convergence over time.
To continue with a system defined via such a map, we need to introduce the concept of time:

Definition 3.2 (Time set).
A *time set* $\mathcal{T}$ is a subgroup of $(\mathbb{R}, +)$.

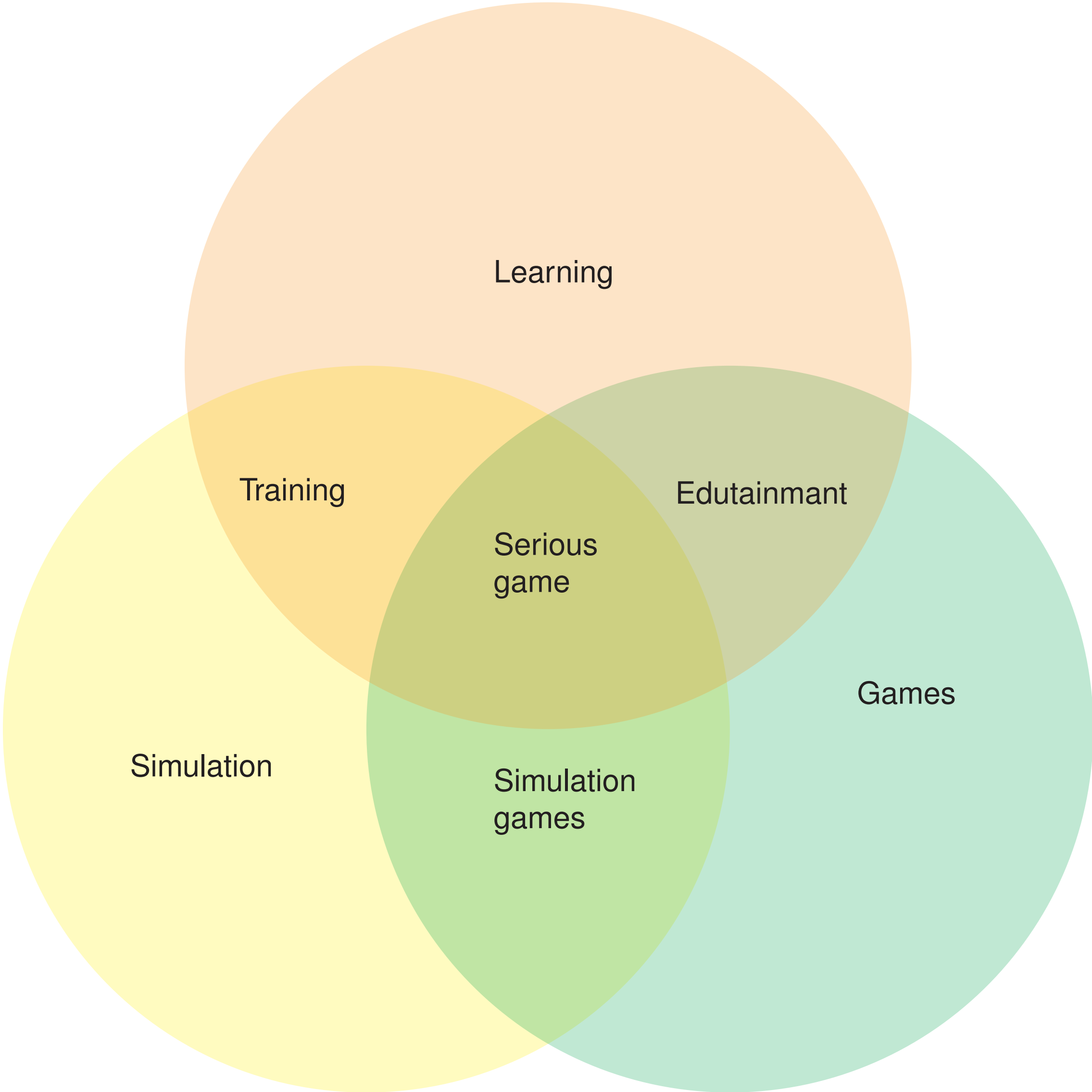


Figure 3.1: Connection of serious games to working methods

Remark 3.3
We like to note that the definition of time in Definition 3.2 allows for continuous time, discrete time and event time. Continuous time is used to model systems varying continuously whereas discrete and event time are sampled.

For our serious game, we apply the discrete time idea, i.e. a user can choose an input at any $t_j \in \mathcal{T}$ where j indicates the time choices in Algorithm 14.

Algorithm 14 Serious game

Input: Connected network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$
Input: Multiplicities $\mathcal{C}_{\mathcal{E}} : \mathcal{E} \to \mathbb{R}_0^{n_{\mathcal{E}}}$
Input: Markings $\mathcal{C}_{\mathcal{V}} : \mathcal{V} \to \mathbb{R}_0^{n_{\mathcal{V}}}$
Input: Constraints on infrastructure and utilities $C : \mathcal{V} \times \mathcal{E} \to \mathbb{R}_0^{n_{\mathcal{E}}}$

```
procedure SERIOUS GAME(N, C_E, C_V, C)
    while Not stopped do
        Get input e_j ∈ E, v_j ∈ V
        Complement input to solution
        Evaluate network problem
    end while
end procedure
```

Output: Solution statistics

Task 3.4 (Time set)
Give an example of a continuous time, discrete time and event time set.

Solution to Task 3.4: The continuous time set is $\mathcal{T} = \mathbb{R}$. Introducing a time discretization by a factor $T \in \mathbb{R}^+$, we obtain the discrete time set $\mathcal{T} = \{t \in \mathbb{R} \mid t = t_0 + k \cdot T;\ k \in \mathbb{Q}, t_0 \in \mathbb{R}\}$. In contrast to the equidistant nature of the discrete time set, the event time set is given by $\mathcal{T} = \{t_j \mid \forall j,k:\ t_j \neq t_k\ \wedge t_j, t_k \in \mathbb{R}\}$.

In order to discuss about convergence of results for our serious game, we first require an internal notion for the condition of the game. To this end, we introduce the so called *state of a system*.

Definition 3.5 (State).
Consider a system $\Sigma : \mathcal{U} \to \mathcal{Y}$. If the output $\mathbf{y}(t)$ uniquely depends on the history of inputs $\mathbf{u}(\tau)$ for $t_0 \leq \tau \leq t$ with $t_0, \tau, t \in \mathcal{T}$ and some $\mathbf{x}(t_0)$, then the variable $\mathbf{x}(t)$ is called *state* of the system and the corresponding set $\mathcal{X}$ is called *state set*.

Task 3.6
Reconsider the example network sketched in Figure 2.9. Sketch a possible state representing a feasible solution.

Solution to Task 3.6: A possible state is given in Figure 3.2.

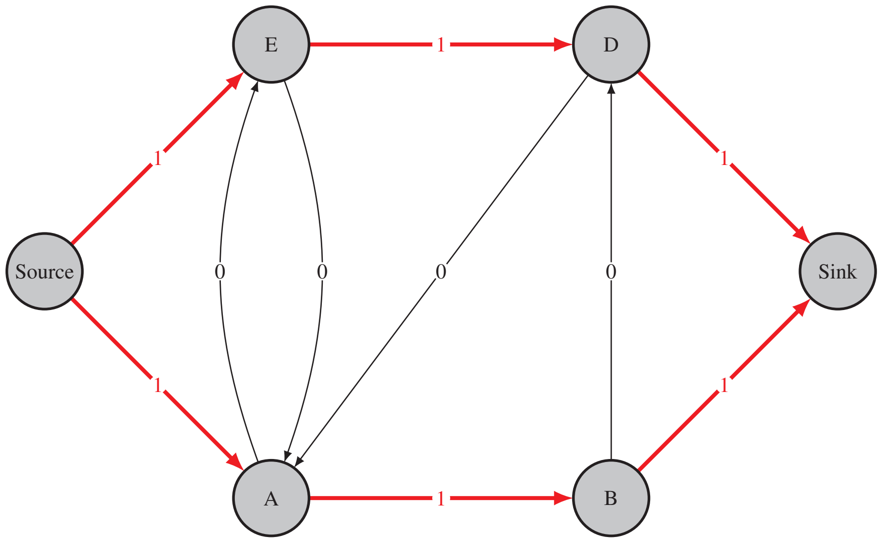


Figure 3.2: State of example network from Figure 2.9

Remark 3.7
Given Task 3.6, the aim of serious gaming is to obtain the optimal solution for the network by guessing. To this end, improvements taking, e.g., flow increasing steps are suitable and are then identified by the users.

In discrete time, that is $\mathcal{T} = \mathbb{Z}$, we obtain the standard description of a state space system:

Definition 3.8 (Discrete time system).
Consider a system $\Sigma : \mathcal{U} \to \mathcal{Y}$ in discrete time $\mathcal{T} = \mathbb{Z}$ satisfying the property from Definition 3.5. If $\mathcal{X}$ is a vector space, then we call it *state space* and refer to

$$\mathbf{x}(t_{j+1}) = f(\mathbf{x}(t_j), \mathbf{u}(t_j), t_j), \quad \mathbf{x}(t_0) = \mathbf{x}_0 \tag{3.1a}$$
$$\mathbf{y}(t_j) = h(\mathbf{x}(t_j), \mathbf{u}(t_j), t_j). \tag{3.1b}$$

as *discrete time system*. Moreover, $\mathbf{u}$, $\mathbf{y}$ and $\mathbf{x}$ are called *input*, *output* and *state* of the system.

Remark 3.9
Similar to the discrete time case, descriptions for continuous time and event driven systems exist.

For convergence of our serious game and many other practical applications, so-called operating points are of interest. These points exhibit the property that the dynamics comes to a stop which is practially relevant to have steady operations, e.g. constant transports or other logistics processes.

Definition 3.10 (Operation point).
Consider system (3.1). Then the pairs $(\mathbf{x}^\star, \mathbf{u}^\star)$ satisfying

$$f(\mathbf{x}^\star, \mathbf{u}^\star) = \mathbf{x}^\star \tag{3.2}$$

are called *operating points* of the system. If (3.2) holds true for any $\mathbf{u}^\star$, then the operating point is called strong or robust operating point.

The dynamic reveals a *flow* of the system at hand, whereas a *trajectory* is bound to a specific initial value and input sequence. The following Figure 3.3 illustrates the idea of flow and trajectory. In this case, the flow is colored to mark its intensity whereas the arrows point into its direction. The trajectory is evaluated for a specific initial value and „follows" the flow accordingly.

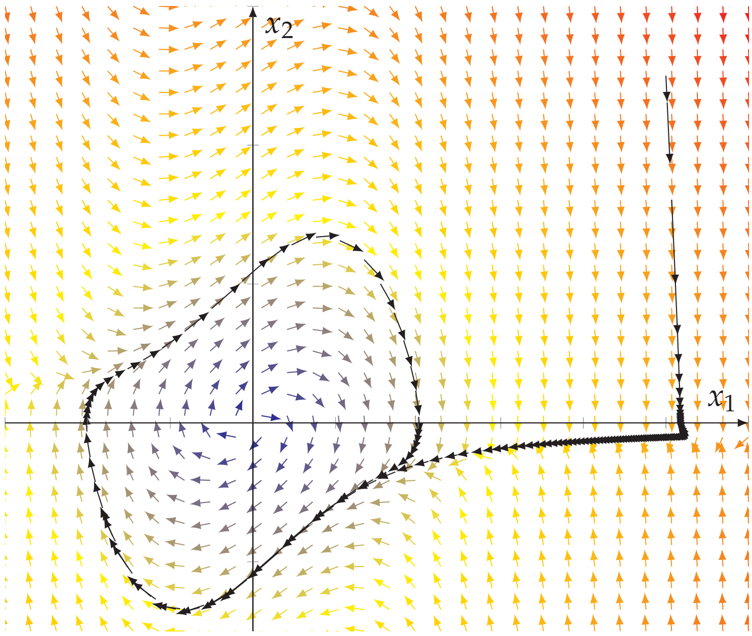


Figure 3.3: Sketch of a dynamic flow and a trajectory

Remark 3.11
Figure 3.3 additionally marks the point that an operating point is not necessarily a point but may be an orbit. In system theory, the property of such an orbit we are looking for is attraction, i.e. any solution converges to it.

This convergence is exactly what we are looking for given a serious game. To make this point clear, we use the following example.

Task 3.12 (Beer game)
Consider the beer game[1] *given a four tier supply chain network to produce, distribute and sell beer as depicted in Figure 3.4. Suppose each player represents one company within a single tier and may order quantities of beer packs per week while simultaneously keeping the stock at a reasonable level with no backlog orders as they generate additional costs.*

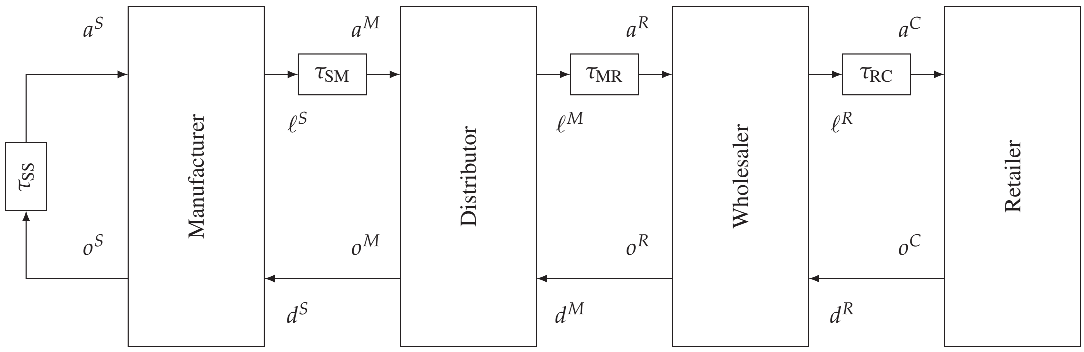


Figure 3.4: Sketch of a three stage supply network

From this basic example, two basic observations may be made:

- If players communicate with one another, then convergence is achieved faster than without communication.
- If players do not understand the complexity of the system, then the solution is ramping up quickly and diverging.

Within this section, we focus on the latter, the so called Bullwhip effect for supply chains:

Definition 3.13 (Bullwhip effect).
Consider a supply chain system $\Sigma : \mathcal{U} \to \mathcal{Y}$ similar to Figure 3.4. We call a behavior of the system a *bullwhip effect* if increase in demand and delay on delivery lead to an excessive increase in orders throughout the supply chain.

Graphically, Figure 3.5 sketches the Bullwhip effect within a supply chain.
The bullwhip effect signifies the increasing distortion of information and the amplification of demand fluctuations as orders float from the customer end to the manufacturer end of the supply chain.

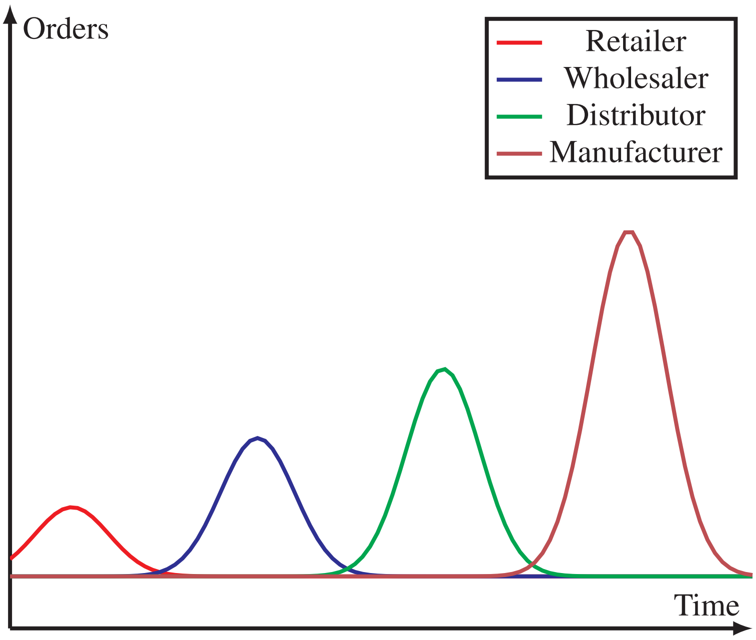


Figure 3.5: Bullwhip effect in supply chain

Task 3.14

Given the example network from Figure 2.9. Split the network into two parts similar to the supply chain given in Figure 3.4.

Solution to Task 3.14: Figure 3.6 highlights a possible split, in this case using the predecessor and successor sets of the sink and source respectively.

Several countermeasures may help to mitigate the bullwhip effect by improving coordination, reducing information distortion, and aligning inventory levels with actual customer demand. Common employed strategies include

- Smoothing demand: Encouraging stable and consistent customer demand, e.g. via price incentives, can help reduce the amplification of fluctuations.
- Just-in-time (JIT) and lean principles: Implementing JIT principles, such as reducing lead times, improving production flexibility, and minimizing batch sizes, can help align production with actual customer demand. This approach reduces inventory levels and enhances responsiveness in the supply chain.
- Supply chain visibility and real-time data: Implementing information systems, such as Enterprise Resource Planning (ERP) systems, Warehouse Management Systems (WMS),

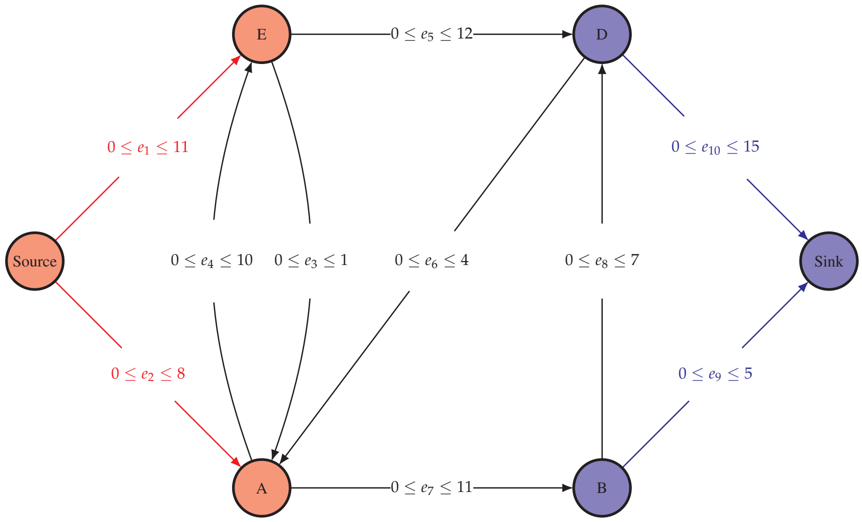


Figure 3.6: Separating the example network from Figure 2.9

or RFID technology, can provide real-time data on inventory levels, demand patterns, and order statuses and allows to respond to fluctuations promptly.

- Reduce order and delivery variability: Minimizing variability in order quantities and delivery lead times may enhance coordination without communication.
- Collaborative planning, forecasting, and replenishment (CPFR): CPFR is a framework that emphasizes collaboration between supply chain partners in planning, forecasting, and replenishing inventory.
- Demand forecasting and information sharing: Accurate demand forecasting, e.g. via sharing demand information among supply chain partners, can help to align production and inventory levels.
- Vendor-managed inventory (VMI): VMI involves the supplier or manufacturer having access to inventory data at the retailer's end and taking responsibility for replenishment decisions.
- Strategic partnerships and long-term contracts: Establishing long-term relationships and strategic partnerships with key suppliers and customers can enhance trust, communication, and collaboration to improve forecasting.

Among the latter, the first four points address local control, i.e. without communication. As such, these ideas are limited by disturbances emanating from supply chain partners, which remain unknown. The last four points require some kind of information exchange.

Table 3.1: Advantages and disadvantages of serious gaming

Advantage	Disadvantage
✓ Requires no preknowledge	✗ Delivers suboptimal solution
✓ Applies to all systems	✗ Neglects bullwhip
✓ May be distributed	✗ Disregards automation

Despite being discussed as cartels, these alternatives offer insights in the structure of the supply chain. In the following sections, we discuss the typical structures of such chains.

3.2 Leader follower

In most logistics networks and also in transport systems, there exist big players who dominate their local networks. Typical examples may be OEMs for trucks, trains, airplanes, cars etc., but also market players like airline companies, 3PL, shipping lines and so forth. Locally, even bus companies are able to enforce a bus schedule according to their needs. Apart from the strategic level, also on the tactical level leader follower structures are possible: For example, AGVs in harbors are typically coordinated by a central entity.
Abstracting from these examples, respective entities are – at least to some extend – able to define their solution and other entities have to accept this lead. Figure 3.7 illustrates the sequence of decisions.
Here, we first need to split the system

$$\mathbf{x}(t_{j+1}) = f(\mathbf{x}(t_j), \mathbf{u}(t_j), t_j), \quad \mathbf{x}(t_0) = \mathbf{x}_0 \tag{3.3}$$

into multiple ones, which represent the entities within a transport and logistics system. Note that these entities may be load units, utilities, infrastructure or even combinations of the latter. Our only requirement is that the subsystem is controlled by one and only one operations logic.

Definition 3.15 (Entity).
Consider a system $\Sigma : \mathcal{U} \to \mathcal{Y}$ given by (3.1). Then we call

$$\mathbf{x}^p(t_{j+1}) = f^p(\mathbf{x}^p(t_j), \mathbf{u}^p(t_j), t_j), \quad \mathbf{x}^p(t_0) = \mathbf{x}_0^p$$

operating on $\mathcal{X}^p \subset \mathcal{X}, \mathcal{U}^p \subset \mathcal{U}$ a *subsystem* and its operations logic an *entity*.

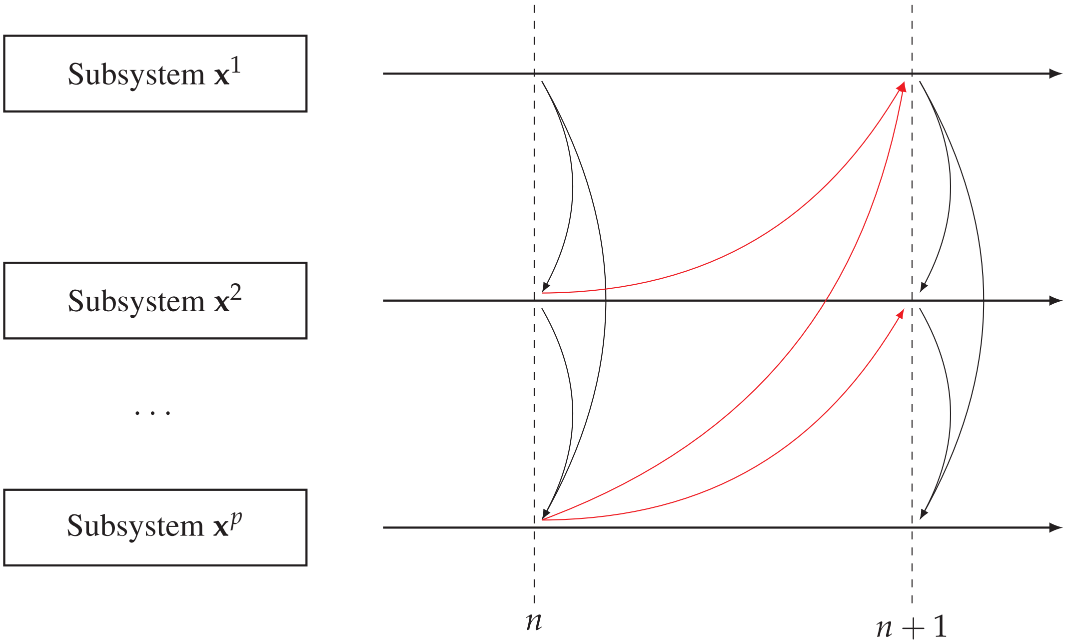


Figure 3.7: Communication structure for leader follower systems

Remark 3.16
Note that the solution to Task 3.14 provides an example for separating a network into two entities.

We like to point out that the split into subsystems is not always clear. The following example illustrates this point:

Task 3.17
Consider the dynamics

$$\begin{pmatrix} \mathbf{x}_1(t_{j+1}) \\ \mathbf{x}_2(t_{j+1}) \end{pmatrix} = \begin{pmatrix} \mathbf{x}_1(t_j) + \mathbf{x}_2(t_j) + \mathbf{u}(t_j)/2 \\ \mathbf{x}_2(t_j) + \mathbf{u}(t_j) \end{pmatrix}$$

and split the system into two subsystems using $\mathbf{x}^1 = \mathbf{x}_1$, $\mathbf{x}^2 = \mathbf{x}_2$ *and* $\mathbf{u}^2 = \mathbf{u}$.

Solution to Task 3.17: Setting $\mathbf{x}^1 = \mathbf{x}_1$, $\mathbf{x}^2 = \mathbf{x}_2$ and $\mathbf{u}^2 = \mathbf{u}$ and leaving $\mathbf{u}^1$ undefined, we obtain

$$\begin{aligned}\mathbf{x}^1(t_{j+1}) &= \mathbf{x}^1(t_j) + \overbrace{\mathbf{x}^2(t_j) + \mathbf{u}^2(t_j)}^{\text{from subsystem 2}}/2 \\ \mathbf{x}^2(t_{j+1}) &= \mathbf{x}^2(t_j) + \mathbf{u}^2(t_j).\end{aligned}$$

For that choice, subsystem 2 is independent from subsystem 1. However, to evaluate subsystem 1 the information $i^1(t_j)$ is required to evaluate $\mathbf{x}^2(t_j)$ and $\mathbf{u}^2(t_j)$ from subsystem 2. Note that the connection depends on how the input from the overall system is assigned to the subsystems. Setting $\mathbf{u}^1 = \mathbf{u}$ and leaving $\mathbf{u}^2$ undefined, both subsystems depend on each other.

The aim of a split is that by recombining the subsystems (3.4) we reobtain the overall transport and logistics system (3.1).
As we have seen in Task 3.17, it may be necessary to split up both the state set $\mathcal{X}$ as well as the input set $\mathcal{U}$. To do that in a coordinated manner, we introduce the following:

Definition 3.18 (Projection).
Given a set S, let $\pi : S \to S$ be a linear map which is idempotent, that is $\pi \circ \pi = \pi$. We call π a projection of S onto $\mathrm{Im}(\pi)$ (along $\mathrm{Ker}(\pi)$) where $\mathrm{Im}(\pi)$ and $\mathrm{Ker}(\pi)$ denote the image and kernel of π.

The projectors can be interpreted as focus lenses to highlight entities only. Apart from highlighting, the projectors directly deliver the links between entities. These will be the basis for any coordination using communication. Formally, we can apply the projections to define a decomposition of a vector space:

Definition 3.19 (Decomposition).
Consider a set S, a set $\mathcal{P} = \{1, \ldots, P\}$ where $P \in \mathbb{N}$, and a set of projections $(\pi^p)_{p \in \mathcal{P}}$ where $S^p := \mathrm{Im}(\pi^p)$ is a subset of S for all $p \in \mathcal{P}$ to be given. If we have that

$$\langle (S^p)_{p \in \mathcal{P}} \rangle = S \text{ and } S^q \cap \langle (S^p)_{p \in \mathcal{P}, p \neq q} \rangle = \{0\} \text{ for all } q \in \mathcal{P}$$

hold, then we call the set $(S^p)_{p \in \mathcal{P}}$ a *decomposition* of S.

Now we can use the decompositon to rewrite our overall system into subsystems defined on subspaces, i.e. entities working on pieces of the transport and logistics network. To derive the entities, we require two projection sets for all $p \in \mathcal{P}$, that is

- $\pi_{\mathcal{X}}^{p} : \mathcal{X} \to \mathcal{X}$ to split the state set such that $\mathrm{Im}(\pi_{\mathcal{X}}^{p}) = \mathcal{X}^{p}$, and
- $\pi_{\mathcal{U}}^{p} : \mathcal{U} \to \mathcal{U}$ to split the input set such that $\mathrm{Im}(\pi_{\mathcal{U}}^{p}) = \mathcal{U}^{p}$.

Unfortunately, these projections will in general not simply separate the state and input set. We already saw the reason for this deficiency in Task 3.17: Subsystems may depend on variables which we project into other subsystems. Hence, the projection in general leave us with three components each, that is:

- For the state projection, we obtain $[\mathcal{X}^{p}, \widetilde{\mathcal{X}}^{p}, \overline{\mathcal{X}}^{p}]$ where $\mathbf{x}^{p} \in \mathcal{X}^{p}$ are our primary variables of interest. In particular, we have that $\widetilde{\mathbf{x}}^{p} \in \widetilde{\mathcal{X}}^{p}$ are the states of neighbors necessary to evaluate the projected dynamic $\pi_{\mathcal{X}}^{p} \circ f$ correctly.
- For the input projection, we have $[\mathcal{U}^{p}, \widetilde{\mathcal{U}}^{p}, \overline{\mathcal{U}}^{p}]$ where again $\mathbf{u}^{p} \in \mathcal{U}^{p}$ is at the core of our interest. Again, $\widetilde{\mathbf{u}}^{p} \in \widetilde{\mathcal{U}}^{p}$ is the necessary input information of neighbors to evaluate the projected dynamic $\pi_{\mathcal{X}}^{p} \circ f$.

Remark 3.20
Note that the inputs $\widetilde{\mathbf{u}}^{p} \in \widetilde{\mathcal{U}}^{p}$ are computed by different entities. Hence, to include them to evaluate another system, we have to transmit the respective data.
Different from $\widetilde{\mathcal{X}}^{p}$ and $\widetilde{\mathcal{U}}^{p}$ we find that $\pi_{\mathcal{X}}^{p} \circ f$ is independent of $\overline{\mathbf{x}}^{p} \in \overline{\mathcal{X}}^{p}$ and $\overline{\mathbf{u}}^{p} \in \overline{\mathcal{U}}^{p}$. For this reason, we call the latter independent states and inputs.

Task 3.21
Reconsider the example network sketched in Figure 2.9. Color code the state/input projections.

Solution to Task 3.21: Figure 3.8 shows the color codes split where the green lines represent the required information by both the red and the blue subsystem.

Utilizing the latter task, we obtain that $\widetilde{\mathcal{X}}$ and $\widetilde{\mathcal{U}}$ are required information to be exchanged, and in particular from which entity this information is required. This reveals

Definition 3.22 (Neighboring index set).
Consider a decomposition of system (3.1). Then we call $\mathcal{I}^{p} = \{p_1, \ldots, p_m\} \subset \mathcal{P} \setminus \{p\}$ neighboring index set if it satisfies

$$(\mathcal{X}^{p_1} \times \ldots \times \mathcal{X}^{p_m}) \times (\mathcal{U}^{p_1} \times \ldots \times \mathcal{U}^{p_m}) \supset (\widetilde{\mathcal{X}}^{p} \times \widetilde{\mathcal{U}}^{p}). \tag{3.4}$$

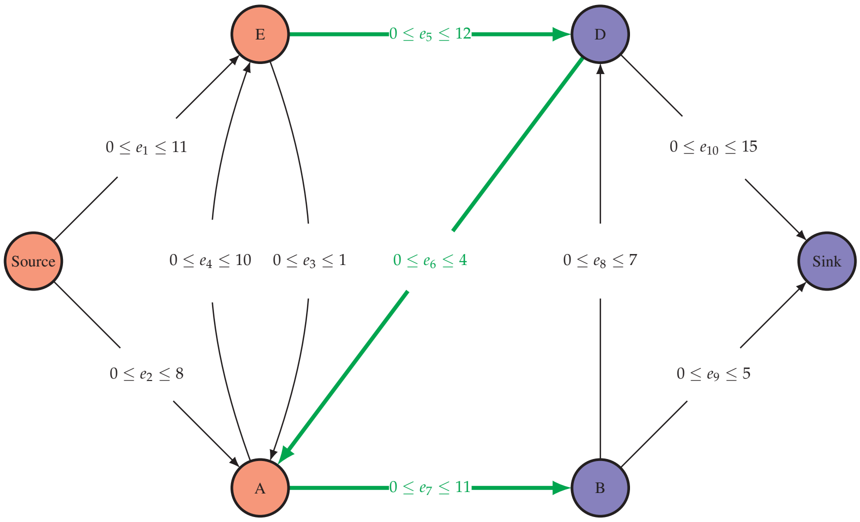


Figure 3.8: Projections for example network from Figure 2.9

Here, we like to stress that the above definition allows us to simply define all systems as part of the index set. However, regarding bandwidth constraints, it is typically a good idea to keep these sets as small as possible. The respective data is called neighboring data:

Definition 3.23 (Neighboring data).
Consider a neighboring index set $\mathcal{I}^p(t_j)$ of subsystem $p \in \mathcal{P}$. We call the set

$$i^p(t_j) = \{(q, t_{j,q}, \mathbf{x}^q, \mathbf{u}^q) \mid q \in \mathcal{I}^p(t_j)\} \in I^p \tag{3.5}$$

neighboring data. The neighboring data set is given by $I^p = 2^Q$ with $Q = (\mathcal{P} \setminus \{p\}) \times \mathbb{N}_0 \times \mathcal{X} \times \mathcal{U}$.

Task 3.24
Reconsider Task 3.17 and compute neighboring index set and neighboring data.

Solution to Task 3.24: For our choice of variables we have $\mathcal{I}^1(t_j) = \{2\}$ and $\mathcal{I}^2(t_j) = \emptyset$. As we have seen in the solution of Task 3.17, we require the information contained in the neighboring data $i^1(t_j) = \{(2, t_j, \mathbf{x}^2(t_j), \mathbf{u}^2(t_j))\}$ to evaluate the system.

Algorithmically, the leader follower approach is very simple, cf. Algorithm 15.
The downside of the latter algorithm is that the sequence of subsystems in $\mathcal{P}$ is not clear and massively influences the outcome of the method.

Algorithm 15 Leader follower

Input: Decomposition of system $\mathbf{x}(t_{j+1}) = f(\mathbf{x}(t_j), \mathbf{u}(t_j), t_j)$
Input: Subsystems $p \in \mathcal{P}$
procedure LEADER FOLLOWER($f, \mathcal{P}$)
  while Not stopped **do**
    $j \leftarrow 0$
    for all $p \in \mathcal{P}$ **do**
      Compute optimal input $\mathbf{u}(t_j)$
      Send neighboring data $i^p(t_j)$ to all neighbors, $j \leftarrow j+1$
    end for
  end while
end procedure

Remark 3.25
The leader follower approach can be modified to work in full parallel, which is outside the scope of this lecture. While such an approach may be more fair for subsystems which are at the end of the sequence in $\mathcal{P}$, in reality such sequences do exist in particular for transport and logistics systems. There are, however, exceptions, e.g. if utilities such as single trains or busses are considered. In such cases, the sequence should be fair or according to traffic rules.

Table 3.2: Advantages and disadvantages of leader follower

Advantage	Disadvantage
✓ Allows coordination without communication	✗ Introduces SPOF
✓ Applies directly to distributed systems	✗ Limited to distributed systems
✓ Simplifies implementation	✗ Disregards global optimality

Further methods regarding coordination and coordination are subject in lectures such as *Modern control systems*.

BIBLIOGRAPHY

[1] DEUTSCHES INSTITUT FÜR NORMUNG E.V.: *DIN 30781-1:1989-05 Transportkette, Grundbegriffe*. Beuth, 1989

[2] DEUTSCHES INSTITUT FÜR NORMUNG E.V.: *DIN IEC 60050-351 Internationales Elektrotechnisches Wörterbuch Teil 351: Leittechnik (IEC 60050-351:2014-09)*. Beuth, 2014

[3] FEICHTINGER, G. ; HARTL, R.F.: *Optimale Kontrolle ökonomischer Prozesse*. deGruyter, 2011

[4] GUDEHUS, T.: *Logistik: Grundlagen, Strategien, Anwendungen*. 4th edt. Springer, 2010

[5] GUDEHUS, T. ; KOTZAB, H.: *Comprehensive Logistics*. Springer, 2012

[6] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION: *ISO 22400-2:2014 Automation systems and integration — Key performance indicators (KPIs) for manufacturing operations management*. ISO, 2014

[7] NEUMANN, K. ; MORLOCK, M.: *Operations Research*. 2nd edt. Hanser, 2004

[8] SARDER, M.D.: *Logistics Transportation Systems*. Elsevier, 2020

[9] SCHÖNBERGER, J.: *Model-Based Control of Logistics Processes in Volatile Environments*. Springer, 2011

[10] VOGT, J.J.: *Business Logistics Management*. 5th edt. Oxford University Press, 2016

Jürgen Pannek
Institute for Intermodal Transport and Logistic Systems
Hermann-Blenck-Str. 42
38519 Braunschweig

During summer term 2024 I give the lecture to the module *Multimodal Transport Systems (Multimodale Transportsysteme)* at the Technical University of Braunschweig. To structure the lecture and support my students in their learning process, I prepared these lecture notes. The aim of the module is to provide an overview on intermodal transport and logistics systems with a particular focus on methods for planning, design and coordination of such systems.
In particular, students shall be able to describe, explain, apply and analyze modes and systems in transport and logistics. Moreover, students can recall, interpret and evaluate key performance indicators for unimodal and intermodal systems. Regarding planning and design, students are able to characterize, apply and differentiate methods with respect to the area of application and assess suitability of these methods. Last, students are able to describe, categorize and evaluate methods of coordination regarding intermodality.