Mathematical Modelling, Simulation and Optimisation of Dynamic Transportation Networks

with Applications in Production and Traffic

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften der RWTH Aachen University zur Erlangung des akademischen Grades einer Doktorin der Naturwissenschaften genehmigte Dissertation

vorgelegt von
Dipl.-Math. techn. Ute Ziegler

aus Bensheim

Berichter: Prof. Dr. Michael Herty und Prof. Dr. Simone Göttlich
Tag der mündlichen Prüfung: 27.11.2012

Diese Dissertation ist auf den Internetseiten der Hochschulbibliothek online verfügbar.

Acknowledgements

I want to express my gratitude to all, who were helping me in many different ways to succeed in writing this dissertation. First of all I would like to thank my supervisors Prof. Michael Herty and Prof. Simone Göttlich, who supported me immensely with their knowledge, advice and the extremely pleasant and cordial working atmosphere. I also want to thank Prof. Arie Koster for his cooperation and sharing his precious knowledge on discrete optimization. I am also thankful to Bernd Bollwerk, our kind and patient Admin, who was always there to help out, whenever technical problems or special software necessities came up. Furthermore, I would like to thank Agnes Dittel for many interesting discussions and for sharing her valuable knowledge about Cplex-programming, Nico Behrent who was always disposed to give useful hints to find my programming bugs and Daniel Junglas who explained me in detail, how to find the right spot to introduce heuristics into the vast universe of Cplex code. Thanks for many useful discussions and hints to Sebastian Kühn and to Oliver Kolb, who explained me the ideas of the staggered Lax-Friedrichs Scheme. I appreciate the hospitality and useful exchanges with Prof. Dieter Armbruster and Prof. Christian Ringhofer, who warmly welcomed me at Arizona State University. Furthermore, I am grateful to Prof. Evelyn Buckwar to give me the opportunity continue my research in Heriot-Watt University in Edinburgh. I am also grateful for the opportunity to meet Prof. Pierre Degond in Université Paul Sabatier in Toulouse. Thanks to the financial support of DFG (project no. HE 5386/6-1) and DAAD (research grants no. 50727872, 50021880, 54365630, D/06/28175 and D/08/11076) and many thanks to RWTH University for supporting me with a research degree completion grant.

Thanks to Richard Barnard, for tuning the style of my introduction and conclusion part and to all colleagues for the friendly and enjoyable atmosphere.

And many thanks to Philipp Monreal, not only for patiently proof-reading the whole thesis and helping me to improve style and readability, but especially for his company and friendship that started many years ago. Finally, I want to express my thankfulness to all my friends and to my family for all the support and for enriching my live. Thank you, Lama Ole.

Contents

Introduction

Dynamic transportation networks have a broad range of applications. No matter if we talk about evacuation systems, traffic flow on roads, or production networks, the underlying structure can in many cases be modelled by dynamic flow equations on networks, which we call dynamic transportation networks (short DTNs). This work contains a description of DTNs in general and presents two specific models: firstly, production networks including dynamic machine capacities and repair workers; and secondly, traffic networks including several junction types as well as traffic lights. Optimisation questions such as optimal worker scheduling or optimal traffic light settings arise. For these complex model structures, classical continuous optimization techniques cannot guarantee to find globally optimal solutions. Hence, it is of interest to develop strategies for transforming DTNs into linear mixed integer optimisation problems (short MIPs), which allow for automated Branch & Bound optimisation techniques. Furthermore, it is rewarding to investigate how knowledge about the problem structure can be used to speed up the optimisation process.

Here, we give a brief but incomplete survey on the existing literature in this field. Plenty of models are dedicated to network flow problems and transportation on networks. The classical description is the maximum flow problem derived in graph theory, where the maximal flow through a network is computed with respect to given upper bounds, also referred to as capacities, see [41]. In recent decades, various models that incorporate dynamic phenomena into the network description have been developed. They are dedicated to various applications, such as queuing theory [9, 16] and supply chain models [2, 3, 24, 32, 35, 42, 46, 47, 54], networks for gas and water pipelines [20], traffic flow models [11, 17, 18, 19, 43, 72], evacuation scenarios [22, 52] , telecommunication networks [34], and many more. In this context, mainly two modelling streams emerged: on the one hand microscopic models – which describe the trajectory of every

single particle in the system, such as discrete event simulators [3, 7]; and, on the other hand, macroscopic models, which use fluid-like descriptions of the transportation process by considering the evolution of density in the system and often entail the use of differential equations [4, 31, 32, 42, 47]. The latter have the advantage that simulation time does not depend on the number of particles in the system and that dynamic phenomena such as forwards and backwards travelling density waves – which occur e.g. in traffic flows – can be reproduced.

Having found a satisfactory description for the dynamics, it is of interest to tackle various optimisation questions, see [22, 37, 42, 46, 48, 52, 57, 98] for an overview. Unfortunately, standard optimisation techniques for continuous PDE/ODE- constrained problems, such as Lagrangian based adjoints and gradient based methods [63, 93] are not reliable for network structured flow models, since they often get stuck in local optima and cannot deliver any information about how close the best found solution is to the global optimum.

Consequently, it is reasonable to follow an alternative optimisation approach. In [8, 37, 42, 48, 49, 53, 57, 75, 76, 77] linearisable dynamic transportation network models are reformulated into linear MIPs using linearisation techniques as described in [62]. Linear mixed integer optimisation is a common problem class in discrete mathematics and entails well-investigated optimisation methods that are able to find global optimal solutions in a reliable way [30, 70, 82, 90, 94]. Elaborated Branch & Bound operations split the original problem into subproblems and compute upper and lower bounds for the globally optimal objective function value. Nowadays numerous software packages exist that can be used as blackbox solvers [23, 60, 85, 91]. Another remarkable advantage is, that discrete decisions – such as binary controls for traffic light settings – and restrictions to integer values – e.g. restrictions to integer numbers of repair workers for production networks – can easily be incorporated. However, this method has the disadvantage that the optimisation time depends exponentially on the number of discretisation steps. Hence, a trade-off between acceptable computation times and accuracy of the description of the underlying dynamics has to be made. Furthermore, the automated solvers encounter problems in finding feasible solutions since every variable originating from the discretised and linearised model description is treated as unknown. Especially, when the grid sizes are small, rounding errors often accumulate, resulting in artificial infeasibilities, such that the optimisation algorithm fails, see [49, 98].

Nevertheless, the knowledge about the problem structure provides many opportunities to stabilise and speed up the automated optimisation algorithms. One approach is a bound-sharpening presolve algorithm, developed by [36]. Another method is the use of starting heuristics, giving the solver a jump start with a promising feasible solution [49, 98]. However, the provision of a feasible start solution does not guarantee a reduction of optimisation time.

As already mentioned, feasible solutions for the linear MIP can be computed easily as soon as the actual control variables are given. Apart from providing starting solutions – to the best of the author's knowledge – this valuable fact has not been applied to its full potential in the course of the optimisation procedure. Providing feasible solutions not only as start up but also during the Branch & Bound Algorithm, paired with prescribed branching priorities is a promising approach to speed up the optimisation procedure.

The content of this work is structured as follows: In Chapter 1 we consider macroscopic dynamic network flow models, i.e. DTNs. We present a general definition for DTNs, review the most common modelling approaches for density evolution and coupling conditions and consider in particular two applications: Firstly, we derive an extension to the common production flow network by including capacity declines and the effect of repair workers. Secondly, we review Lighthill-Witham-Richards traffic flow models [74, 88] and analyse coupling conditions for specific junctions types. In the end we review the most common discretisation techniques and propose a novel algorithm to simulate density evolution on road networks using coupled Hamilton-Jacobi equations [1, 97].

In Chapter 2 we consider optimisation questions on DTNs and outline optimisation techniques used in the solvers for linear MIPs. Then, we point out a general strategy for the transformation of DTNs into linear MIPs and discuss how constraints on the actual control variables can be constructed to avoid undesired oscillation effects. Furthermore, we point out where exactly the knowledge of the problem structure can be used during the Branch & Bound process to speed up the optimisation time. We resume the particular models from Chapter 1 and transform them into linear MIPs. In the context of production networks we are looking for the optimal worker scheduling in order to maximise the production flow. Regarding traffic networks, we derive the modelling of traffic lights, derive requirements on the traffic light settings and propose

a heuristic to find feasible traffic light settings in order to speed up the optimisation algorithm.

In Chapter 3 we show numerical results for the two applications, involving verifications of model behaviour and the effects of several parameters, comparison of discretisation schemes, illustrations of the effect of optional requirements to avoid undesired fluctuations, and the development of the optimisation process comparing the use of several tuning techniques.

Besides general classifications and transformation strategies for DTN models into linear MIPs, the new scientific contribution of this work is the novel production model containing dynamic process capacities and the reformulation into an optimal worker scheduling problem applying linearisation techniques to obtain a linear MIP. In the context of LWR-based traffic flow network models, we show how coupling conditions can be transformed into easily linearisable min-terms for several junction types. Furthermore, we develop a numerical algorithm based on coupled Hamilton-Jacobi equations. Moreover, we derive a traffic light model with dynamic switching periods and constraints for secure traffic light settings. We provide a transformation into a linear MIP and proposed various bounding heuristics to speed up the optimisation algorithm.

Parts of this work will be or have been published in the following journals:

- Göttlich, S. and Herty, M. and Ringhofer, C. and Ziegler, U. *Production systems with limited repair capacity.* Optimization, Vol. 61(8), pp. 915-948, 2012.
- Göttlich, S. and Herty, M. and Ziegler, U. *Numerical discretization of Hamilton-Jacobi equations on networks.* Networks and Heterogeneous Media, in review-process.
- Göttlich, S. and Herty, M. and Ziegler, U. *Modeling and optimizing traffic light settings on road networks.* IEEE Transactions on Automatic Control, in review-process.

1

Network Flow Modelling

This chapter is dedicated to dynamic network flow models, focusing on continuous ODE/PDE-based fluid flow descriptions.

Network flow models have a broad range of applications. On the one hand a lot of research has been done on static models, for which numerous algorithms to solve optimisation issues in polynomial time are available, see [40]. On the other hand, another important field has become a point of interest in the last decades: Time dependent models for transportation systems, which often consider PDE-dynamics to describe the density evolution in the system; starting from internet and telecommunication networks [34], over networks for gas and water pipelines [20], and evacuation scenarios [52], to production flows in economics, see [2, 3, 24, 32, 35, 42, 47, 54] amongst others. A further application are traffic flows on complex road networks [11, 17, 19, 43, 72].

The original network flow problem is dedicated to finding an optimal static throughflow a network where every part is limited by a certain capacity, see [41]. However, in many applications it is of interest to capture dynamical development and to model the transportation process more detailed. To those models we refer to as dynamic transportation networks (short DTNs). Models describing the transportation and trajectories of every single part in the system have been developed, such as models on discrete event simulation [3, 7]. They are called microscopic models. A disadvantage is that, as soon as the number of involved parts grows, the model becomes highly complex and cannot be solved efficiently. For that reason it has become popular to treat the parts as fluid flow and model the density evolution using differential equations whenever the number of parts tends to be large, see [2, 5, 26, 32] for an overview. These

models are often referred to as macroscopic models.

This chapter is structured as follows: In Section 1.1 we provide basic definitions and notation and review some general ideas on the modelling of DTNs, including descriptions of transportation along network edges, buffers and coupling conditions on vertices.

The following two sections focus on DTN models in the context of production and traffic flows on networks:

Section 1.2 describes the main ideas of macroscopic fluid-like production models, such as [32], which consist of capacitated production flows and an ODE-based description for buffers to store waiting parts. The new contribution is an extension of the already known model: We include the deviation of production capacities, that occur due to abrasion effects and breakdown of machines as well as the impact or repair workers. We model these additional dynamics in a smooth way using additional ordinary differential equations. This model extension has recently been published, see [49].

We combine these ideas, including clear priority rules, in a similar way as they are used for cell transmission models [25] and create 4-legged junctions which form part of roundabouts. Furthermore, we reformulate the coupling conditions, which are originally stated as maximisation problems, into min-terms, which can be computed in a straight-forward way. Then, inspired by [83], we reformulate the traffic network flow problem using Hamilton-Jacobi equations, that enable us to compute car trajectories of given scenarios very easily. This advantage has also been used in [18].

In Section 1.4, we summarise the most common discretisation approaches used in the context of DTNs (for a complete overview, see [73]) and derive a novel numerical algorithm to simulate traffic flow models using a reformulation with Hamilton-Jacobi equations. This yields the benefit that we can directly track trajectories of single cars.

1.1 Preliminaries

. The idea of this section is to give a general classification and point out, what we actually mean by the term DTN (dynamic transportation network), namely models on networks where transportation is described by dynamic functions interdepending on each other. Subsequently, we provide the most common modelling techniques to describe transportation and coupling conditions for these model types which can be

adapted according to the specific application. Later-on, in Sections 1.2 and 1.3, we fill these rather abstract definitions and derivations with life by presenting certain models in the context of production and traffic flow networks.

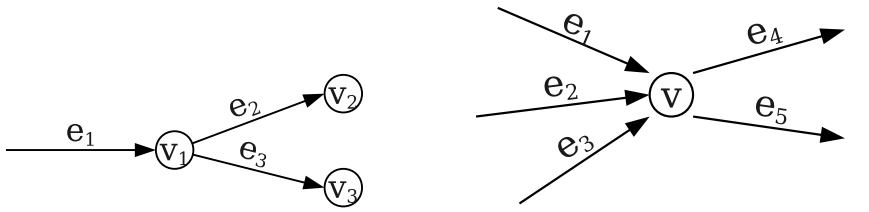
First of all, we introduce some basic notation that is needed to describe the network structure and give a general definition of dynamic transportation networks. We consider models where the flow is only moving along a given direction. For this reason, the structure is based on a directed Graph $G = (V, E)$, where V describes the set of vertices and E the set of directed edges.

In the context of directed graphs, we use the following notations.

Definition 1.1.1. *Given a directed graph $G = (V, E)$.*

- *V denotes the set of vertices, and E the set of edges.*
- *Function $\alpha : E \to V$ maps each edge to its* **starting point** *and function $\omega : E \to V$ maps each edge to its* **endpoint**, *cf. Figure 1.1(a).*
- *The* **set of incoming edges** *of v is denoted by $\delta_v^{in} := \{e \in E : \omega(e) = v\}$ and $\delta_v^{out} := \{e \in E : \alpha(e) = v\}$ is referred to as the* **set of outgoing edges** *of v for all vertices in V, cf. Figure 1.1(b).*
- *A vertex $v' \in V : (\exists e \in E : \alpha(e) = v' \wedge \omega(e) = v)$ is called* **predecessor** *of v.*
- *A vertex $v' \in V : (\exists e \in E : \alpha(e) = v \wedge \omega(e) = v')$ is called* **successor** *of v.*
- *Vertices without predecessors are called* **inflow vertices** *and are collected in the set $V^{in} = \{v \in V : \{e : \omega(e) = v\} = \emptyset\}$. An edge with an inflow vertex as starting point is called* **inflow edge**. *The set of inflow edges is referred to as $E^{in} \subset E$.*
- *Vertices without successors are called* **outflow vertices** *and are collected in the set $V^{out} = \{v \in V : \{e : \alpha(e) = v\} = \emptyset\}$. An edge with an outflow vertex as endpoint is called* **outflow edge**. *The set of outflow edges is referred to as $E^{out} \subset E$.*

Before introducing dynamic networks, we state the classical network flow problem as basis and motivation for all succeeding considerations.



(a) Starting and end vertices. $v_1 = \omega(e_1) = \alpha(e_2) = \alpha(e_3)$, $v_2 = \omega(e_2)$, $v_3 = \omega(e_3)$

(b) Set of incoming and outgoing edges. Here, $\delta_v^{in} = \{e_1, e_2, e_3\}$ and $\delta_v^{out} = \{e_4, e_5\}$

Figure 1.1: Notations on directed graphs.

Static Network Flow Problem. The classical maximum flow problem (short MFP) is found in graph theory. Consider a specific network $G = (V, E)$. Every edge has a maximal capacity, representing the upper bound of through-flow the edge. The question is how to find the maximal amount of flow that can be assigned to the edges respecting cost and benefit parameters.

Given edge capacities c_i for all $i \in E$, a general (MFP) is given by

$$\max \sum_{i \in E} f_i \tag{1.1a}$$

such that

$$\sum_{i \in \delta_v^{in}} f_i = \sum_{j \in \delta_v^{out}} f_j, \qquad \forall v \in V \backslash \{V^{in} \cup V^{out}\} \tag{1.1b}$$

$$0 \leq f_i \leq c_i, \qquad \forall i \in E. \tag{1.1c}$$

see [40, 41, 66], amongst others.

Figure 1.2 shows an example.

A feasible solution of (1.1) describes a static (i.e. time independent) flow scenario of the given network.

This model consists of time independent variables and is useful as long as only static network properties are considered. However, it is of interest in many applications to analyse time dynamic behaviour. As soon as inflow, distribution of flow at branching points or other network properties are changing in time, the network has to be described with other tools including differential equations that describe the development of flows

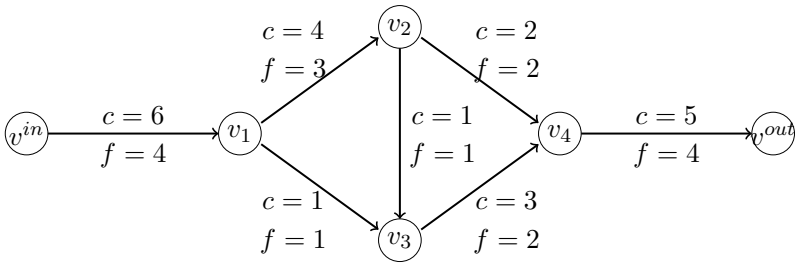


Figure 1.2: Example for a maximum flow problem. The values for f are the optimal solution with respect to the given capacities c.

and queues inside the network. In this way we end up with an extension describing dynamic network flow models including non-linearities.

Subsequently, we state a basic definition of dynamic transportation networks (DTNs). The crucial point of DTNs is that there are several time dependent properties and functions defined on edges of a directed graph. The interdependence of these functions can be described by edge and coupling operators. Furthermore, typically initial and boundary conditions are given. For more details, see Definition 1.1.2.

The network structure is essential to define a dynamic transportation network. (Note, that in this context the number of elements of a vector v is denoted by $|v|$.)

Definition 1.1.2. *A **dynamic transportation network (DTN)** is given by*

- *a directed graph $G = (V, E)$,*
- *a time dimension expressed by variable t and a time horizon T, such that $t \in [0, T] \subset \mathbb{R}_0^+$,*
- *a (possibly time dependent) flow distribution matrix $d \in \mathbb{R}^{|E|\times|E|}$ (details, see Definition 1.1.7) (optional),*
- *a set of network parameters $\mathcal{N} \in \mathbb{R}^{|\mathcal{N}|}$,*
- *a set of edge properties $\mathcal{P}_i \in \mathbb{R}^{|\mathcal{P}_i|}$ $\forall i \in E$,*
- *a spatial dimension expressed by variable x for each edge, details see Remark 1.1.4 (optional),*

- *a set of dynamic functions* $t \mapsto \mathcal{D}_i(t) \in [\mathcal{L}_i, \mathcal{U}_i] \subset \mathbb{R}^{|\mathcal{D}_i|}$*, where* $\mathcal{L}_i$ *and* $\mathcal{U}_i$ *denote the upper and lower bounds of the functions and are usually elements of the network properties* $\mathcal{P}_i$*. (In case of spatial dynamics,* $\mathcal{D}$ *might also depend on* x*) with initial conditions* $\mathcal{D}_{0i} = \mathcal{D}_i(0)\ \forall i \in E$*,*

- *boundary conditions for dynamic functions (if they inherit a spatial dimension), or inflow into the network, described by boundary functions* $\mathcal{B}_i^{in} : t \mapsto \mathcal{B}_i^{in}(t) \in \mathbb{R}^{|\mathcal{B}_i^{in}|}\ \forall i \in E^{in}$*. Optionally right-hand boundary conditions at the outflow edges can be described:* $\mathcal{B}_i^{out} t \mapsto \mathcal{B}_i^{out}(t) \in \mathbb{R}^{|\mathcal{B}_i^{out}|}\ \forall i \in E^{out}$

- *interdependencies (possibly in form of differential equations) of the before mentioned objects described by a set of edge operators* $\mathcal{I}_i(t, \mathcal{N}, \mathcal{P}_i, \mathcal{D}_i, \partial_t \mathcal{D}_i, \mathcal{B}_i) \in \mathbb{R}^{|\mathcal{I}_i|}\ \forall i \in E$ *(In case of spatial dynamics,* $\mathcal{I}$ *might also depend on* x *derivatives of* $\mathcal{D}_i$ *in* x*),*

- *interdependencies of objects from neighbouring edges described by coupling operators* $\mathcal{C}_v(t, d, \mathcal{N}, \mathcal{P}, \mathcal{D}) \in \mathbb{R}^{|\mathcal{C}_v|}\ \forall v \in V$*,*

- *edge conditions of the form* $\mathcal{I}_i(t, \ldots) \equiv 0,\ \forall i \in E$ *and*

- *coupling conditions of the form* $\mathcal{C}_v(t, \ldots) \equiv 0,\ \forall v \in V$*.*

Remark 1.1.3.

- *Depending on the type of coupling, the coupling operators and conditions might also by indexed by the edges* $i \in E$ *instead of the vertices* $v \in V$*.*

- *Usually, the dynamic functions* $\mathcal{D}_i$ *are modelled such that the upper and lower bounds,* $\mathcal{L}_i$ *and* $\mathcal{U}_i$*, are automatically fulfilled. However, we mention them here, since they play an important role for optimisation procedures, as we will see in Chapter 2.*

To fill the previous definition with life, we present some specific DTNs in the next sections. Section 1.2 is dedicated to the derivation of DTNs in the context of production and Section 1.3 considers DTNs for traffic flows on road networks.

The next subsection gives an overview over the most common modelling approaches for DTNs. Subsection 1.1.1 is dedicated to the modelling of movements along network edges and in Subsection 1.1.2 some basic ideas on coupling conditions at the network vertices are pointed out.

1.1.1 Transportation along Network Edges

We consider an amount of particles which are moving along a prescribed direction x. The movement can either be tracked microscopically by describing the trajectories of each single particle, or it can be considered in a macroscopic way, by describing the density evolution. The latter approach is preferable for settings where we assume the particles to have identical properties and where the amount of particles is very large, such that considering each single trajectory would be too costy. There are numerous ways to describe macroscopic particle flow. The most simple approach is to assume that the particles move with constant velocity, without disturbances. In that way, we would only need to compute the time delay, which the flow of a certain point x_1 needs to reach another point x_2.

However, for many applications, especially for traffic flow models, it is important to be able to capture nonlinear dynamic behaviour describing phenomena such as traffic jams and backwards and forward travelling density waves, as observed on highways. In such cases, the density evolution is described using a conservation law, which is a hyperbolic partial differential equation.

Every edge allows for one spatial flow direction. The particles are treated as small mass points. The density on an edge is given by $\rho : (x,t) \mapsto \rho(x,t) \in [0, \rho^{max}] \subset \mathbb{R}$, where x is the spatial variable indicating the location on the edge and t refers to the time. Furthermore ρ^{max} is the maximal possible density of particles which is a known property and depends on the size of every particle. The amount of particles inside an interval $[x_1, x_2]$ at time t is given by

$$\int_{x_1}^{x_2} \rho(x,t)dx.$$

The particle flow $f : (x,t) \mapsto f(x,t) \in \mathbb{R}_0^+$ describes the amount of particles crossing each point of the edge in one time unit. The amount of particles passing through location x during the time interval $[t_1, t_2]$ is given by

$$\int_{t_1}^{t_2} f(x,t)dt.$$

Usually we assume that the particles cannot get lost along an edge and no new particles can appear (except for the start and endpoint of the edge). This means that the amount of particles in an arbitrary interval inside the edge $[x_1, x_2]$ at a certain time t_2 minus

the amount of particles in the same interval at an earlier time t_1 must be equal to the difference of inflowing particles at location x_1 minus outgoing particles at location x_2 during the time interval $[t_1, t_2]$. In other words, the following equations holds:

$$\int_{x_1}^{x_2} \rho(x, t_2)dx - \int_{x_1}^{x_2} \rho(x, t_1)dx = \int_{t_1}^{t_2} f(x_1, t)dt - \int_{t_1}^{t_2} f(x_2, t)dt. \tag{1.2}$$

If ρ and f are sufficiently smooth, (1.2) yields

$$\int_{t_1}^{t_2} \int_{x_1}^{x_2} \partial_x f(x, t) + \partial_t \rho(x, t)dxdt = 0. \tag{1.3}$$

Since (1.3) holds for all t_1, $t_2 > 0$ and all intervals $[x_1, x_2]$ inside the edge $\alpha(e), \omega(e)$, we obtain the continuity equation, a hyperbolic partial differential equation, describing the conservation of mass:

$$\partial_x f(x, t) + \partial_t \rho(x, t) = 0. \tag{1.4}$$

If the flow depends solely on the density, i.e. $f = f(\rho)$, we have

$$\partial_x f(\rho(x, t)) + \partial_t \rho(x, t) = 0. \tag{1.5}$$

For more details, we refer to [10, 73, 95].

We can model this dynamic behaviour on every edge of a given network graph $G = (V, E)$.

Remark 1.1.4. *To include these spatial dynamics into the description of a DTN model, we include the spatial dimension of each network edge $i \in E$ with a variable x living in the interval $[0, L_i] \subset \mathbb{R}_0^+$. According to the notation of Definition 1.1.2, the edge length L_i is considered to be one of the edge properties $\mathcal{P}_i$. Furthermore, f_i and ρ_i are considered as dynamic functions and, hence, are elements of $\mathcal{D}_i$. Moreover, the edge operator*

$$\mathfrak{I}_i(\rho, f, \partial_x f, \partial_t \rho) = \partial_x f(\rho) + \partial_t \rho$$

and the edge condition

$$\mathfrak{I}_i \equiv 0$$

are required in order to satisfy (1.5).

To derive a useful DTN model, convenient coupling conditions are required at the vertices $v \in V$. General ideas on coupling are found in the next subsection.

1.1.2 Coupling

We give a definition of the in- and outflow of an edge.

Definition 1.1.5. *Given a transportation network on a directed graph $G = (V, E)$. We refer to the flow that is leaving an edge i by $\hat{f}_i$, and the flow entering edge i is referred to as $\bar{f}_i$, cf. Figure 1.3. In case that length $L_i \in \mathbb{R}^+$ is a given edge property, as described in Remark 1.1.4, we have $\bar{f}_i(t) := f_i(x = 0, t)$ and $\hat{f}_i(t) := \hat{f}_i(x = L_i, t)$.*

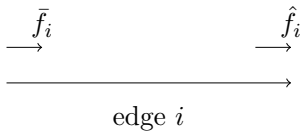


Figure 1.3: In- and outflow of an edge.

Remark 1.1.6. *In some application the model includes buffers in the beginning of each edge, where particles can be stored. In this case, the incoming flow $\bar{f}_i$ can either be diverted to or increased by the parts in the buffer. Let $u_i(t)$ represent the function for the buffer size in front of edge i and f_i the flow entering the edge after having traversed the buffer, see Figure 1.4. Then we have*

$$f_i = \bar{f}_i - \partial_t u_i. \tag{1.6}$$

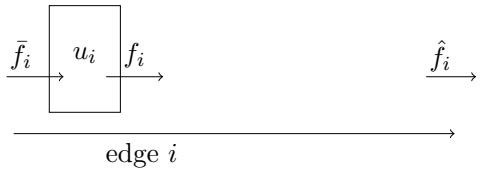


Figure 1.4: Flow functions $\bar{f}_i$, f_i and $\hat{f}_i$ for an edge with buffer u_i.

A typical application for this scenario are production networks as, for example, described in [2, 4, 31, 37, 42]. Another example is derived in more detail in Section 1.2.

In the case that no dynamics along the edges and no buffers are considered, as for example in the MFP (1.1), we have $\bar{f}_i = \hat{f}_i$, $\forall i \in E$.

Consider a vertex $v \in V$. In general we require conservation of mass trough nodes, i.e. we do not want to loose or gain any particles at the vertices, hence Kirchhoff's law has to be fulfilled:

$$\sum_{i \in \delta_v^{in}} \hat{f}_i(t) = \sum_{j \in \delta_v^{out}} \bar{f}_j(t), \quad \forall v \in V \tag{1.7}$$

Sometimes, we also have given requirements on the flow distribution at vertices. Then, we use parameters $0 \leq d_{ij} \leq 1$ which prescribe the percentage of flow going from edge i to edge j. This yields

$$\bar{f}_j(t) = \sum_{i \in \delta_v^{in}} d_{ij} \cdot \hat{f}_i(t), \ \forall j \in \delta_v^{out}. \tag{1.8}$$

The distribution parameters have to be chosen such that

$$\sum_{i \in \delta_v^{out}} d_{ij} = 1 \tag{1.9}$$

holds. Then (1.8) guarantees that (1.7) holds.

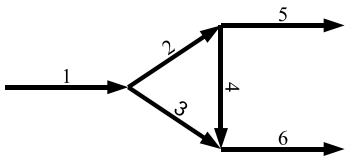
This gives rise to the following definition:

Definition 1.1.7. *Given a directed graph $G = (V, E)$. A matrix $d \in \mathbb{R}^{|E| \times |E|}$ is called a **flow distribution matrix**, when the following properties are fulfilled:*

- $0 \leq d_{ij} \leq 1, \ \forall (i, j) \in E \times E$,
- $\left(\nexists v \in V : \{ i \in \delta_v^{in} \wedge j \in \delta_v^{out} \} \right) \Rightarrow d_{ij} = 0$, *i.e. entries only differ from zero, when edge j is a direct successor of edge j.*
- $\sum_{i \in \delta_v^{out}} d_{ij} = 1 \ \forall j \in E \backslash E^{in}$, *i.e. all incoming flow is distributed, see (1.9).*

Note, that the notation "d" without indices refers to the matrix, whereas "d_{ij}" refers to one element of the matrix.

Definition 1.1.7 implies that the rows sums are 1, i.e. $\sum_{j \in E} d_{ij} = 1$, except of all outgoing edges $j \in E^{out}$. An example is shown in Figure 1.5.



$$\longrightarrow \qquad \mathbf{d} = \begin{pmatrix} 0 & 0.5 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.3 & 0.7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Figure 1.5: Example network with corresponding matrix d. The entry d_{ij} represents the percentrage of flow going from edge i to edge j.

Remark 1.1.8. *According to the notation of Definition 1.1.2, the coupling conditions (1.7) and (1.8) can be included in a DTN model via the coupling operator*

$$\mathcal{C}_v(\bar{f}, \hat{f}) = \sum_{i \in \delta_v^{in}} \hat{f}_i - \sum_{j \in \delta_v^{out}} \bar{f}_j, \ \forall v \in V, \tag{1.10}$$

for settings, in which the flow distribution is variables, and by

$$\mathcal{C}_j((d, \bar{f}, \hat{f}) = \bar{f}_j - \sum_{i \in \delta_{\alpha(j)}^{in}} d_{ij} \hat{f}_i, \ \forall j \in E \backslash E^{in}, \tag{1.11}$$

for cases with prescribed flow distribution matrix d.

The coupling condition is then given by

$$\mathcal{C}_j \equiv 0.$$

Remark 1.1.8 shows that the coupling operator is defined for each edge as soon as the flow distribution is given by d, cf. (1.11). In cases with the flow distribution is variable, it is sufficient to guarantee Kirchoff's law and define the coupling operator for each vertex, cf. (1.10).

In cases where more involved dynamical phenomena such as forward and backwards travelling density waves, e.g. for traffic flow models, we obtain more complex requirements for the coupling densities. This is due to the fact that we have to exclude inadmissible wave directions at the junction. For more details see Section 1.3 and [11, 19, 64], for an overview.

The next sections apply these modelling ideas in the context of production networks and traffic flow networks.

1.2 Application I: Production Network Models

Complex production processes often consist of numerous production steps. In the beginning, raw material is introduced into the system and passed on from one machine to the next, which successively execute various production steps, until the finished product is obtained as output at the end of the chain, cf. Figure 1.6.

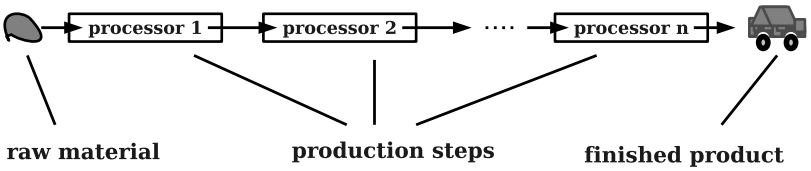


Figure 1.6: Chain of processors.

In general, a production system consists of several branching points and provides various paths where parts are manufactured. Hence, the underlying framework is in many cases a large system of production units, such as suppliers and machines, which are interpreted as a production network.

1.2.1 Transport and Buffers

In the production context, the edges of a network represent different production units, such as assembly lines or machines, where certain production steps are executed. They usually include buffers, where those parts are stored that cannot be processed immediately. The need for buffers is a result of the assumption that there is an upper bound for the particle flow in each unit which represents the production capacity of the corresponding machine. In this context, stationary models on queuing theory have been derived, see [9, 16], where the main focus lies on the mean waiting and arrival time of parts.

An alternative approach is the modelling of dynamics in processors and buffers. The resulting instationary models either consider discrete events [3, 7] or consider continuous flows and compute the density evolution in the system, see [4, 31, 32, 42, 47] for an overview.

In this subsection we will have a closer look at the latter mentioned fluid-like models as a basis and present an extension in the next subsection.

Let $u_i : t \mapsto u_i(t)$ represent the queue size, i.e. the number of waiting parts, in the buffer of edge i, $\bar{f}_i : t \mapsto \bar{f}(t)$ describes the flow from preceding edges entering the buffer of edge $i \in E$ and $\hat{f}_i : t \mapsto \hat{f}(t)$ describes the flow leaving the buffer u_i, cf. Figure 1.7.

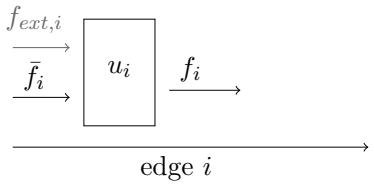


Figure 1.7: Flow and buffer on network edge. For all incoming edges $i \in E^{in}$ $\bar{f}_i \equiv 0$, such that only external inflow $f_{ext,i}$ enters the buffer. For all other edges $i \in E \backslash E^{in}$ the external inflow $f_{ext,i}$ is optional.

The processing capacity is the upper bound for the flow and is given by c_i, $i \in E$. External inflow into processor i is prescribed by $f_{ext,i}(t)$ for $i \in E^{in}$. For all other processors $f_{ext,i} \equiv 0$ holds.

Let d be the flow distribution matrix as stated in Definition 1.1.7. As described in Subsection 1.1.2, the flow coming from preceding edges is given by

$$\bar{f}_i(t) = \sum_{k \in \delta^{in}_{(\alpha(i))}} d_{ki} \hat{f}_k(t), \tag{1.12}$$

where $\hat{f}_i$ describes the flow leaving edge i, cf. equation (1.8).

Remark 1.2.1. *The indexing of (1.12) differs from (1.8) for the following reason: In (1.8), we consider various edges indexed by i and their common **succeeding** edge j, whereas here, we consider a fixed edge i and its **preceding** edges indexed with k.*

We derive equations describing the dynamics of the buffer level u_i at time t. Imagine that the flow which is led from preceding edges to a certain edge i first of all enters the buffer and is then passed on to the actual machine. As already mentioned in Remark 1.1.6, the evolution of the buffer level is given by the difference between the amount of flow entering and leaving the buffer. We get:

$$\frac{du_i(t)}{dt} = \bar{f}_i + f_{ext,i}(t) - f_i(t). \tag{1.13}$$

It is of general interest to efficiently use the available capacity and run a production system at high working load, i.e. when a lot of parts are running through the system. For this reason, we assume that the maximal production capacity of each machine is used whenever possible. That means, the machine runs with its full capacity, as long as there are parts waiting in the buffer, otherwise as much as possible of the incoming parts are used. This yields the following equation:

$$f_i(t) \quad = \quad \begin{cases} \min\{\bar{f}_i(t), c_i(t)\}, & u_i(t) = 0 \\ c_i(t), & u_i(t) > 0 \end{cases} \tag{1.14}$$

In order to avoid the discontinuous dependence of f on the buffer level u, [46] suggests to use a small regularisation parameter $0 < \tau \ll 1$, and replace (1.14) by the relaxed formulation

$$f_i(t) = \min(c_i, \frac{u_i(t)}{\tau}). \tag{1.15}$$

In Section 1.3.1 of [46] is shown that (1.15) is equivalent to (1.14) for the limit $\tau \to 0$.

There are different ways to model the transport of particles from the buffer to the actual machine. If the dynamic of the transportation as such is of interest, it is modelled using the continuity equation (1.5), as described in Subsection 1.1.1. If we imagine the particles to move on conveyor belts, they are transported with constant velocity v. Then, the flow function is given by $f(\rho) = v \cdot \rho$. In this case, the continuity equation (1.5) simplifies to the advection equation

$$\partial_t \rho(x,t) + v \partial_x \rho(x,t) = 0 \tag{1.16}$$

This approach is for example used in [36, 37, 48, 49, 57, 98].

If the main focus does not lie on the density evolution along the edges, it is also convenient to interpret τ_i as a previously fixed throughput time for each edge and integrate the transportation time indirectly in the description of the buffer u_i. In that way τ_i causes a smoothed out delay of parts entering the buffer until they are produced. In that case the assembly line and the buffer are treated as one entity and the space variable x along the edges is neglected, cf. Figure 1.8.

We have $f_i(t) \approx \frac{u_i(t)}{\tau_i}$ and $f_i(t)$ is equivalent to $\hat{f}_i(t)$, the flow leaving the edge, see details in [2, 57].

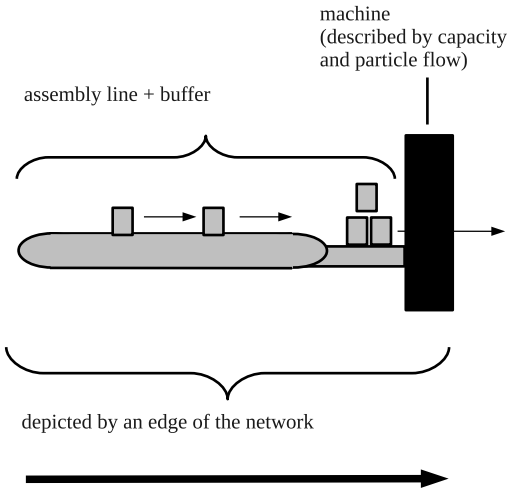


Figure 1.8: Sketch of processor layout.

In summary, we consider the following ordinary differential equation for the buffer levels:

$$\frac{du_i(t)}{dt} = \bar{f}_i + f_{ext,i}(t) - f_i(t) \tag{1.17a}$$

$$f_i(t) = \min\Big\{c_i(t), \frac{u_i(t)}{\tau_i}\Big\}. \tag{1.17b}$$

The interpretation of (1.17) is as follows: Parts are fed into the network, are transported from one machine to another and can be stored in buffers in case of capacity shortage. As soon as parts have traversed the outflow edges, they leave the system. It is clear that the conservation of mass through the whole network should hold, since no parts are lost or generated inside the network. This gives rise for the following lemma.

Lemma 1.2.2. *Let $t \in [0,T] \subset \mathbb{R}$ and $f_i : [0,T] \to \mathbb{R}_0^+$ and $f_{ext,i} : [0,T] \to \mathbb{R}_0^+$ be L^1 functions for all i. Then the total conservation of mass for an initially empty network, i.e. $u_i(0) = 0$, is given by*

$$\int_0^t \sum_{i\in E} f_{ext,i}(\tilde{t})\, d\tilde{t} = \sum_{i\in E} u_i(t) + \int_0^t \sum_{i\in E^{out}} f_i(\tilde{t})\, d\tilde{t} \quad \forall t \in [0,T]. \tag{1.18}$$

In other words, the total number of incoming parts until an arbitrary time t has to be equal to the number of parts, that remain inside the network, i.e. stored in buffers, at time t, plus the parts that have already left the network.

Proof. We argue that the total conservation of mass holds true due to constraint (1.17) which ensures the conservation of mass through nodes only. From the construction of the matrix d, we know that each row that represents an edge which is not an outflow edge, contains positive entries between zero and one, describing how much percent of the flow is sent to consecutive processors. Since flow is distributed between consecutive edges, we know $\sum_{j\in E} d_{ij} = 1$ except for all outgoing edges $i \in E^{out}$. Starting from the total flow in the network and considering the structure of d, we deduce for all t:

$$\sum_{i\in E}(d\cdot f(t))_i = \sum_{i\in E}\sum_{j\in E} d_{ij}f_j(t) = \sum_{i\in E}\Big(\underbrace{\sum_{j\in E} d_{ij}}_{=\begin{cases}1 & \forall i\in E\backslash E^{out}\\ 0 & \text{else}\end{cases}}\Big) f_j(t) = \sum_{j\in E\backslash E^{out}} f_j(t). \tag{1.19}$$

Next, we take equation (1.17a), integrate both sides with respect to time and take the sum over all processors:

$$\sum_{i\in E} u_i(t) - \sum_{i\in E} u_i(0) = \int_0^t \Big(\underbrace{\sum_{i\in E}(d\cdot f(\tilde{t}))_i}_{\text{see (1.19)}} - \sum_{i\in E} f_i(\tilde{t}) + \sum_{i\in E} f_{ext,i}(\tilde{t})\Big)\, d\tilde{t}$$

$$\sum_{i\in E} u_i(t) - \sum_{i\in E} u_i(0) = \int_0^t \Big(\underbrace{\sum_{i\in E\backslash E^{out}} f_i(\tilde{t}) - \sum_{i\in E} f_i(\tilde{t})}_{=-\sum_{i\in E^{out}} f_i(\tilde{t})}\Big)\, d\tilde{t} + \int_0^t \sum_{i\in E} f_{ext,i}(\tilde{t})\, d\tilde{t}$$

Sorting all terms and inserting the initial conditions $u_i(0) = 0\,\forall i$ yields the desired result.

□

1.2.2 Model Extension: Abrasion based Capacity Decline

Machines may break down or abrasion effects can lead to decline of processing capacities or complete interruption. In order to keep production running, repair crews are assigned to currently broken-down or ineffecient machines in order to stabilise or increase the production capacity.

In this subsection we show how deviation in the processing capacity and the effects of repair workers on the production efficiency can be modelled. To keep the model simple, breakdown rates are integrated into the model using experimental values on the reliability of each machine. In this way we avoid the inclusion of stochastic fluctuations.

Hence, as opposed to former production network models such as [37, 47, 57] amongst others, we assume that the capacity is not a fixed parameter, but can fluctuate within the production process.

There is a fixed upper bound for $c_i(t)$, the maximal capacity $\mu_i \in \mathbb{R}^+$. The evolution of capacities depends on the constant breakdown rates l_i and the constant repair rates r_i, which are both deterministic parameters obtained by measurements. Furthermore the capacity evolution is influenced by the percentage of repair workers allocated to each machine, denoted by $\beta_i(t) \in [0,1] \subset \mathbb{R}$. The total number of repair workers is given by $W \in \mathbb{N}$.

We assume that workers are assigned simultaneously and immediately, i.e. time delays at the time of worker shifts are neglected. However, we have to be aware that too frequent worker changes are not convenient in practice. In other words, $\beta_i(t)$ should not be a highly oscillating function, but instead have a lower bounded TV-norm. Hence, we choose $\beta_i(t)$ to be piecewise constant with a reduced amount of discontinuity jumps.

We model breakdowns of machines (rather crudely) by a continuous process reducing the capacity by $l_i \Delta t$ in the interval Δt. The parameter r_i denotes the efficacy of a repair-worker when working on machine i. The meaning of r_i is that assigning W workers to repair machine i will result in increasing the capacity of the machine by an amount $\Delta c_i = W r_i \Delta t$ in the (infinitesimal) time interval Δt. Therefore the rate of change in capacity is given by the equation

$$\frac{dc_i(t)}{dt} = r_i \beta_i(t) W - l_i. \tag{1.20}$$

Depending on the application, the decrease of capacity can be the consequence of abrasion during the production process or because a machine runs out of material. In this case it makes sense to link the magnitude of the capacity descent to the amount of processed parts. Then, equation (1.20) changes to

$$\frac{dc_i(t)}{dt} = r_i \beta_i(t) W - l_i \cdot f_i(t) \tag{1.21}$$

Equation (1.20) has to be modified as to guarantee that the capacity $c_i(t)$ is bounded from below by zero and from above by some maximal capacity μ_i. Here, μ_i denotes the capacity value when all the parts of the machine are running, and therefore assigning any repair workers to the machine would be wasteful. We approach this in the same

way we model the flow function $f_i(t)$ in (1.17a), by introducing a small relaxation parameter ϵ, and model the dynamic evolution of the capacity c_i by

$$\frac{dc_i(t)}{dt} = \min\left\{\frac{\mu_i - c_i(t)}{\epsilon}, Wr_i\beta_i(t)\right\} - \min\left\{\frac{c_i(t)}{\epsilon}, l_i\right\}. \tag{1.22}$$

Equation (1.22) will asymptotically produce the correct bounded dynamics for $0 < \epsilon \ll 1$:

The interpretation is as follows:

- Assuming a total loss of capacity, i.e. $c_i(t) = 0$. This implies

  $$\partial_t c_i(t) = \min\left\{\frac{\mu_i}{\epsilon}, Wr_i\beta_i(t)\right\}.$$

  Hence, the broken machine will be repaired and the capacity starts increasing again.

- The machine works with maximal capacity $\mu_i = c_i(t) > 0$. Equation (1.22) reduces to

  $$\partial_t c_i(t) = -\min\left\{\frac{\mu_i}{\epsilon}, l_i\right\}.$$

  Then, the capacity rate can only decrease.

- The capacity is $0 \ll c_i(t) \ll \mu_i$. This yields $\partial_t c_i(t) = Wr_i\beta_i(t) - l_i$. Hence, the capacity increases, if $Wr_i\beta_i(t) > l_i$ and decreases otherwise.

For cases in which the capacity decline depends proportionally on the through-going material flow, see (1.21), we use

$$\frac{dc_i(t)}{dt} = \min\left\{\frac{\mu_i - c_i(t)}{\epsilon}, Wr_i\beta_i(t)\right\} - l_i \cdot f_i(t) \tag{1.23}$$

instead of (1.22). Since the flow is bounded by the capacity, see (1.15), it is already guaranteed that c will not become negative. For that reason, we do not need to construct another min-term expression as in (1.22).

Remark 1.2.3. *A complete production network model including dynamic capacities and repair workers is given by the following DTN model:*

- *directed graph* $G = (V, E)$,
- *time horizon* $t \in [0, T]$,

- *flow distribution matrix* d,
- *network parameters* $\mathcal{N} = \big(W,\, \epsilon\big)^T$,
- *edge properties* $\mathcal{P}_i = \big(\mu_i,\, r_i,\, l_i,\, \tau_i\big)^T \; \forall i \in E$,
- *dynamic functions* $\mathcal{D}_i = \big(f_i,\, \bar{f}_i,\, c_i,\, u_i,\, \beta_i\big)^T \; \forall i \in E$,
- *external inflow* $\mathcal{B}_i = \big(f_{ext,i}\big) \; \forall i \in E$,
- *edge operators:*

$$\mathcal{I}_i^{(1)}(f_i,\, \bar{f}_i,\, f_{ext,i},\, \frac{d}{dt}u_i) = \frac{d}{dt}u_i - \bar{f}_i - f_{ext,i} + f_i,$$
$$\mathcal{I}_i^{(2)}(\tau_i,\, f_i,\, c_i,\, u_i) = f_i - \min\big\{c_i, \frac{u_i}{\tau_i}\big\} \;\; \textit{and}$$
$$\mathcal{I}_i^{(3a)}(W,\, \epsilon,\, \mu_i,\, r_i,\, l_i,\, c_i,\, \frac{d}{dt}c_i, \beta_i) = \frac{d}{dt}c_i - \min\big\{\frac{\mu_i - c_i}{\epsilon}, Wr_i\beta_i\big\} + \min\big\{\frac{c_i}{\epsilon}, l_i\big\} \;\; \textit{or}$$
$$\mathcal{I}_i^{(3b)}(W,\, \epsilon,\, \mu_i,\, r_i,\, l_i,\, f_i,\, c_i,\, \frac{d}{dt}c_i, \beta_i) = \frac{d}{dt}c_i - \min\big\{\frac{\mu_i - c_i}{\epsilon}, Wr_i\beta_i\big\} + l_i f_i, \; \forall i \in E,$$

- *coupling operator*

$$\mathcal{C}_j(H,\, f,\, \bar{f}) = \bar{f}_j - \sum_{i \in \delta^{in}_{\alpha(j)}} d_{ij} f_i, \; \forall j \in E \backslash E^{in},$$

- *edge conditions* $\mathcal{I}_i^{(1)/(2)/(3)} \equiv 0, \; \forall i \in E$ *and*
- *coupling conditions* $\mathcal{C}_j \equiv 0, \; \forall j \in E \backslash E^{in}$.

Chapter 2, Section 2.3, is dedicated to find the optimal worker scheduling for production network based on the DTN model of Remark 1.2.3.

The next section present the derivation of a DTN model in the context of traffic flow on road networks.

1.3 Application II: Traffic Flow Models

In this section we describe a traffic flow model on networks based on the ideas of Lighthill, Whitham and Richards [74, 88]. We discuss coupling conditions for certain kinds of junctions. Subsequently, we consider a reformulation of the network model using Hamilton-Jacobi equations.

Traffic flow models have been intensively studied during the last years; see [11, 12, 17, 19, 28, 33, 43, 55, 56, 58, 74, 78, 79, 83, 88] and the references therein.

In the context of traffic flow models on networks [11, 17, 19, 43, 72], we focus on macroscopic models which are based on partial differential equations for the traffic density (parts per unit length). The network under consideration consists of edges and vertices where edges correspond to unidirectional roads and vertices to road intersections. From a mathematical point of view, we assume the macroscopic equations, more precisely the Lighthill-Whitham-Richards equations (1.24), to hold on each road. For the modelling of roads with different speed limits, we allow for different flow functions. The crucial point in network models is the coupling at junctions. We refer to [11, 17, 21, 25, 39, 56] for an overview. Our approach will pick up these ideas and additionally establish a new coupling rule in the case of merging junctions. We avoid the use of *right of the way parameters* which determine the proportion of flow and instead introduce priority roads, similar to [25], who followed this approach for cell transmission models. As we will see the coupling conditions will lead to a uniquely solvable network problem.

1.3.1 Modelling Traffic on Roads

We consider Lighthill-Whitham-Richards (LWR) traffic flow model for roads [74, 88]. Here, the macroscopic traffic flow is described assuming that it depends solely on the traffic density;

$$\begin{cases} \partial_t \rho + \partial_x f(\rho) = 0, \\ \rho(x,0) = \rho_0(x), \end{cases} \tag{1.24}$$

where $\rho : (x,t) \mapsto \rho(x,t) \in [0, \rho^{max}] \subset \mathbb{R}^+$ denotes the density of cars, $x \in [0, L] \subset \mathbb{R}^+$ describes the location on the road, L is the length of the road (from one intersection to the next) and $t \in \mathbb{R}^+$ denotes the time.

In general, the flow function is concave with a unique maximum at a designated point $\rho^* \in [0, \rho^{max}]$.

Remark 1.3.1. *Typical flow functions are for example*

$$f(\rho) = v(\rho) \cdot \rho \tag{1.25}$$

where

$$v(\rho) = \frac{v^{max}}{\rho^{max}}(\rho^{max} - \rho)$$

is the velocity which cars are assumed to have depending on the actual traffic density, the maximal allowed velocity for the road v^{max} *and the maximal traffic density (bumper-to-bumper density)* ρ^{max}*, see for example [11].*

Another broadly used function with constant velocity λ *for light traffic, i.e.* $\rho \leq \rho^*$*, is the triangular flow function:*

Definition 1.3.2. *Given* $\alpha \in \mathbb{R}^+$ *and maximal density* $\rho^{max} \in \mathbb{R}^+$*, a* ***triangular flow function*** *is given by*

$$f : [0, \rho^{max}] \to \mathbb{R}^+, \quad \textit{with}$$

$$f(\rho) = \begin{cases} \lambda \cdot \rho & \textit{if } 0 \leq \rho \leq \rho^* \\ \lambda \cdot (2\rho^* - \rho) & \textit{if } \rho^* < \rho \leq \rho^{max} \end{cases} . \tag{1.26}$$

with $\rho^* = \frac{1}{2}\rho^{max}$*.*

This function is used by [18, 28, 79] and in the context of data traffic in telecommunication networks by [34], amongst others.

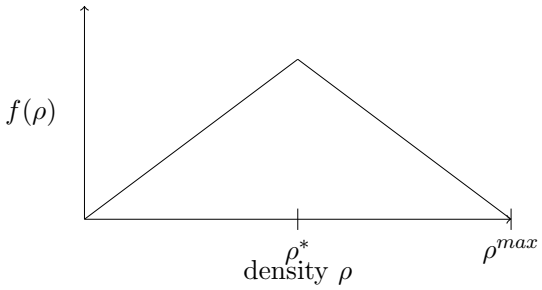


Figure 1.9: Triangular flow function.

We define the function $\tau : [0, \rho^{max}] \to [0, \rho^{max}]$ which maps the density to a distinct density value with equal flow, if existent. In other words we want τ to fulfill the following property:

$$f(\rho) = f(\tau(\rho)),$$

with

$$\tau(\rho) \neq \rho, \text{ if } \rho \neq \rho^*.$$

Since f is a strictly concave function, τ is uniquely defined.

Moreover, $f_l^{-1} : f \mapsto f_l^{-1}(f) \in [0, \rho^*]$ and $f_r^{-1} : f \mapsto f_r^{-1}(f) \in [\rho^*, \rho^{max}]$ denote the inverse flow function for the left and the right side of the maximum, respectively.

For flow function (1.26) τ is given by

$$\tau(\rho) = 2 \cdot \rho^* - \rho. \tag{1.27}$$

1.3.2 Coupling Conditions for Road Networks

We consider a road network given by a directed graph $G(V, E)$, where E denotes the set of edges that represent the roads and V the set of vertices that represent the traffic intersections and which will be referred to as junctions. The length of each road i, leading from one junction to the next, is given by $L_i \in \mathbb{R}^+$. The roads are unidirectional. Different speed limits and amount of lanes of different roads can be modelled by choosing different flow functions. Lanes for different directions can be described by separate edges as depicted for example in Figure 1.15.

The density at the junction for the road of any incoming edge i will be denoted by $\hat{\rho}_i(t)$ and the density at the coupling point for any outgoing road j will be referred to as $\bar{\rho}_j(t)$. At every junction the conservation of cars holds:

$$\sum_{i \in \delta_v^{in}} f(\hat{\rho}_i(t)) = \sum_{j \in \delta_v^{out}} f(\bar{\rho}_j(t)), \quad \forall t > 0, \tag{1.28}$$

as seen in (1.7).

Remark 1.3.3. *The validity of equation (1.28) is also confirmed by [58]: They define a weak solution of the traffic network problem based on the Cauchy problem (1.24) for each road, using smooth test functions $\phi_i : (x, t) \mapsto \phi(x, t), i \in E$ with compact support on $[0, L_i] \times \mathbb{R}_0^+$. and $\phi_i(L_i, t) = \phi_j(0, t)$ and $\partial_x \phi_i(L_i, t) = \partial_x \phi_j(0, t)$ at junctions (i.e. if $i \in \delta_v^{in}$ and $j \in \delta_v^{out}$), $\forall v \in V$, $t \geq 0$. The density functions ρ_i, $i \in E$, are a weak solution if*

$$\sum_{i \in E} \Big(\int_0^\infty \int_0^{L_i} \big(\rho_i \partial_t \phi_i + f(\rho_i) \partial_x \phi_i\big) dx dt + \int_0^{L_i} \rho_i(x, 0) \phi_i(x, 0) dx \Big) = 0$$

is satisfied. For the weak solution, equation (1.28) holds and is also known as Rankine-Hugoniot relation.

For simplicity we use the following notation for the flow at junctions: $\hat{f}_i := f(\hat{\rho}_i))$ for incoming edges and $\bar{f}_j := f(\bar{\rho}_j))$ for outgoing edges.

As in [11, 12, 43, 58], we assume to deal with piecewise constant initial data. In that way, it is possible to use the theory about Riemann problems [59, 92] for the computation of the coupling conditions. A brief review on Riemann problems is given in following paragraph.

The Riemann Problem. A Riemann problem is a Cauchy problem where the initial value is of the form:

$$\rho_0(x) = \begin{cases} \rho_l, & x < 0, \\ \rho_r, & x \geq 0, \end{cases} \tag{1.29}$$

We consider the conservation law (1.5) with initial data of the form (1.29).

It can be shown that the solution is constant on straight lines in the (x,t)-plane, i.e. $\rho(x(t),t) = u(x(0),0)$ with

$$x(t) = \begin{cases} \frac{d}{d\rho} f(\rho_l) \cdot t + x(0), & \text{if } x(0) < 0 \\ \frac{d}{d\rho} f(\rho_r) \cdot t + x(0), & \text{if } x(0) \geq 0. \end{cases}$$

Hence, we obtain the following cases for strictly concave flow function f:

- If $\rho_l < \rho_r$, the solution is given by a shock (in the sense of Lax [71]):

$$\rho(x,t) = \begin{cases} \rho_l, & \text{if } x \leq \frac{f(\rho_r) - f(\rho_l)}{\rho_r - \rho_l} \cdot t, \\ \rho_r, & \text{else.} \end{cases} \tag{1.30}$$

- If $\rho_l > \rho_r$, we get a rarefaction wave:

$$\rho(x,t) = \begin{cases} \rho_l, & \text{if } x \leq \frac{d}{d\rho} f(\rho_l) \cdot t, \\ (\frac{d}{d\rho} f)^{-1}(\frac{x}{t}), & \frac{d}{d\rho} f(\rho_l) \cdot t \leq x \leq \frac{d}{d\rho} f(\rho_r) \cdot t, \\ \rho_r & \text{if } x > \frac{d}{d\rho} f(\rho_r) \cdot t. \end{cases} \tag{1.31}$$

- If $\rho_l = \rho_r$, the solution is constant, namely $\rho(x,t) = \rho_l = \rho_r$.

For more details, we refer to [10, 59, 71].

Deriving admissible coupling densities. We assume that the density terms on each road are initially constant. In this way we obtain initial conditions of Riemann type for the network. To get admissible solutions at junctions, we need waves of non-positive speed for incoming roads and waves of non-negative speed for outgoing roads. This is done by analysing (1.30) and (1.31) for density values $\hat{\rho}_i$ and $\bar{\rho}_i$ of neighbouring roads i and j. The characteristics of the density only leave the junction, when the coupling densities $\hat{\rho}_i$ and $\bar{\rho}_j$ lie in certain regions depending on the initial value on the road close to the junction, i.e. $\rho_i(L_i)$ and $\rho_j(0)$, respectively.

For incoming roads we get:

$$\hat{\rho}_i \in \begin{cases} \{\rho_i(L_i)\} \cup]\tau(\rho_i(L_i)), \rho_i^{max}], & \text{if } 0 \leq \rho_i(L_i) \leq \rho_i^* \\ [\rho_i^*, \rho_i^{max}], & \text{else.} \end{cases} \tag{1.32}$$

And for outgoing roads:

$$\bar{\rho}_j \in \begin{cases} [0, \rho_j^*], & \text{if } 0 \leq \rho_j(0) \leq \rho_j^*, \\ \{\rho_j(0)\} \cup [0, \tau(\rho_j(0))[, & \text{else.} \end{cases} \tag{1.33}$$

Figure 1.10 shows the admissible coupling densities for some exemplary values of $\rho_i(L_i)$ and $\rho_j(0)$.

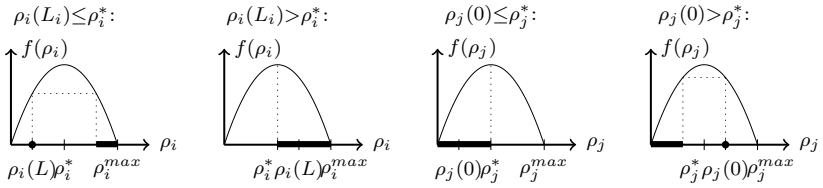


Figure 1.10: Feasible coupling density for incoming road i and outgoing road j (depicted by thick black line).

The existence of solution to Riemann problems on road networks is proven in [19]. To obtain uniqueness, we assume prescribed distribution parameters d_{ij} at junctions in the sense of Definition 1.1.7. As explained in e.g. [11, 19, 43, 55], the previous assumptions still leave us with an additional degree of freedom. Hence, we additionally assume that the drivers' behaviour is to obtain maximal possible flow at junctions.

If the density values at the boundaries of a road are known, we can compute the maximal possible flow at the boundary of the roads using (1.32) and (1.33). This yields:

$$\hat{F}_i := \begin{cases} f_i(\rho_i(L_i)), & \text{if } 0 \le \rho_i(L_i) \le \rho_i^* \\ f_i(\rho_i^*), & \text{else} \end{cases} \tag{1.34}$$

and

$$\bar{F}_j := \begin{cases} f_j(\rho_j^*), & \text{if } 0 \le \rho_j(0) \le \rho_j^* \\ f_j(\rho(0)), & \text{else} \end{cases} . \tag{1.35}$$

As explained before, we assume that drivers behave such that the traffic flow at a junction is maximal respecting the necessary conditions we derived before. Hence, for a general junction v the coupling flow $\left\{\hat{f}_i, i \in \delta_v^{in}; \bar{f}_j, j \in \delta_v^{out}\right\}$ is given by an optimal solution of the following problem:

$$\max \sum_{i \in \delta_v^{in}} \gamma_i \tag{1.36a}$$

such that

$$\gamma_j = \sum_{i \in \delta_v^{in}} d_{ij}\gamma_i, \qquad \forall j \in \delta_v^{out} \tag{1.36b}$$

$$0 \le \gamma_i \le \hat{F}_i, \qquad \forall i \in \delta_v^{in} \tag{1.36c}$$

$$0 \le \gamma_j \le \bar{F}_j, \qquad \forall j \in \delta_v^{out}. \tag{1.36d}$$

We will refer to the optimal solution of (1.36) by $\hat{f}_i \; \forall i \in \delta_v^{in}$ and $\bar{f}_j \; \forall j \in \delta_v^{out}$.

Lemma 1.3.4. *If*

$$\sum_{i \in \delta_v^{in}} d_{ij}\hat{F}_i \le \bar{F}_j, \quad \forall j \in \delta_v^{out} \tag{1.37}$$

holds, (1.36) is uniquely solvable, and the optimal solution is given by

$$\hat{f}_i = \hat{F}_i, \qquad \forall i \in \delta_v^{in} \tag{1.38}$$

$$\bar{f}_j = \sum_{i \in \delta_v^{in}} d_{ij}\hat{f}_i, \qquad \forall j \in \delta_v^{out}. \tag{1.39}$$

Proof. From (1.38) and (1.39) follows immediately that conditions (1.36b) and (1.36c) are fulfilled. Furthermore, (1.36d) holds due to assumption (1.37). Hence, $\{\hat{f}_i, i \in \delta_v^{in}\}$ and $\{\bar{f}_j, j \in \delta_v^{out}\}$ form a feasible solution of (1.36). Let $\{\hat{f}_i^o, i \in \delta_v^{in}\}$ and $\{\bar{f}_j^o, j \in \delta_v^{out}\}$ be a second feasible solution of (1.36). From (1.36c) we get $\hat{f}_i^o \le \hat{f}_i \; \forall i \in \delta_v^{in}$. And since

$\bar{f}_j^o$ is uniquely defined by (1.36b) for all $j \in \delta_v^{out}$, the second solution is only different from the first, when $\hat{f}_i^o < \hat{f}_i$ for at least one $i \in \delta_v^{in}$. But this means for the objective function value that $\sum_{i\in\delta_v^{in}} \hat{f}_i^o < \sum_{i\in\delta_v^{in}} \hat{f}_i$. Hence, $\{\hat{f}_i,\, i \in \delta_v^{in}\}$ and $\{\bar{f}_j\, j \in \delta_v^{out}\}$ form the unique optimal solution of (1.36). □

Lemma 1.3.4 shows that in cases where all arriving flow can be absorbed by the outgoing roads, the optimal solution of (1.36) is unique and can be found easily. Otherwise, there is a bottleneck at the capacity of the outgoing roads. If there is more than one incoming road, we need additional rules – so-called priority rules – to uniquely prescribe the proportion of traffic of the incoming roads that is entering the outgoing edges. One approach is derived on page 32, see (1.46) and (1.47).

As soon as the flow for the coupling is known, a unique coupling density value can be found. It is denoted by $\hat{\rho}_i$ and $\bar{\rho}_j$, respectively. Since we deal with strictly concave flow functions that are piecewise invertible, there are at most two density values that can lead to a specific flow value. Knowing the boundary densities $\rho_i(L_i)$ and $\rho_j(0)$, equations (1.32) and (1.33) lead to further restrictions on the coupling densities $\hat{\rho}_i$ and $\bar{\rho}_j$. In this way we end up with uniquely defined coupling densities given by

$$\hat{\rho}_i = \begin{cases} \rho_i(L_i), & \text{if } f_{il}^{-1}(\hat{f}_i) = \rho_i(L_i) \\ f_{ir}^{-1}(\hat{f}_i), & \text{else} \end{cases} \tag{1.40}$$

and

$$\bar{\rho}_j = \begin{cases} \rho_j(0), & \text{if } f_{jr}^{-1}(\bar{f}_j) = \rho_j(0) \\ f_{jl}^{-1}(\bar{f}_j), & \text{else.} \end{cases} \tag{1.41}$$

In the following paragraphs, we explicitly state the coupling conditions for several specific types of junctions. A similar analysis has been done in [11]. In particular, we focus on priority rules at junctions with two incoming roads and allow different flow functions on each road, e.g. in order to consider roads with different speed limits. Furthermore, we consider the modelling of a realistic roundabout.

Subsequently, we compute the coupling flow at each type of junction. Due to (1.40) and (1.41) we know that this information is enough to uniquely compute the coupling density.

1.3.3 Consideration of specific Junction Types

Two simply connected roads

In a bottleneck situation, the capacity of traffic load decreases at a certain point, as schematically depicted in Figure 1.11.

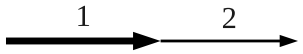


Figure 1.11: Bottleneck road.

For instance, think of road narrows (e.g. when one lane of a multiple lane road ends) or of lower speed limit.

This can be modelled using different flow functions for each part of the road, see also [11]. At the intersection point the coupling condition is given by the maximal feasible flow

$$\max \gamma_1 \tag{1.42a}$$

such that

$$\gamma_2 = \gamma_1 \tag{1.42b}$$

$$0 \leq \gamma_1 \leq \hat{F}_1 \tag{1.42c}$$

$$0 \leq \gamma_2 \leq \bar{F}_2 \tag{1.42d}$$

where $\hat{F}_1$ and $\bar{F}_2$ are given by (1.34) and (1.35), respectively. Obviously, the system (1.42) has the unique optimal solution:

$$\hat{f}_1 = \bar{f}_2 = \min\{\hat{F}_1, \bar{F}_2\}. \tag{1.43}$$

Dispersing junction

We consider the traffic at a dispersing junction as depicted in Figure 1.12. Following the idea of Subsection 1.1.2, we assume the distribution rate at the junction to be previously known due to statistical data. In this way we can use prescribed distribution parameters d_{12} and d_{13}, indicating the percentage of the traffic from road 1 to road 2 and road 3, respectively. This approach is also used by [11, 55]. The parameters may change over time and have to fulfill $d_{12} + d_{13} = 1$, $d_{12} \geq 0$ and $d_{13} \geq 0$.

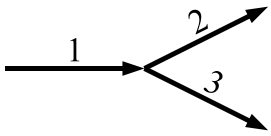


Figure 1.12: Dispersing junction.

As in [25], we assume that the total through-flow at the junction is restricted as soon as one of the outgoing roads is not able to absorb all the designated incoming flow. This corresponds to a first-in-first-out-rule (FIFO) of cars and is a realistic assumption, since a car waiting at the junction blocks all the traffic behind it until it can continue.

Here, we obtain the following optimisation problem:

$$\max \gamma_1 \tag{1.44a}$$

such that

$$\gamma_2 = d_{12}\gamma_1 \tag{1.44b}$$

$$\gamma_3 = d_{13}\gamma_1 \tag{1.44c}$$

$$0 \leq \gamma_1 \leq \hat{F}_1 \tag{1.44d}$$

$$0 \leq \gamma_{2/3} \leq \bar{F}_{2/3}. \tag{1.44e}$$

This linear programming problem can be computed manually by using the Simplex algorithm [30]. Depending on whether $\hat{F}_1$, $\bar{F}_2$ or $\bar{F}_3$ turns out to be the sharpest bound, the solution of (1.44) is given by

$$\bar{f}_2 = \min\{d_{1,2}\hat{F}_1, \bar{F}_2, \frac{d_{1,2}}{d_{13}}\bar{F}_3\}, \tag{1.45a}$$

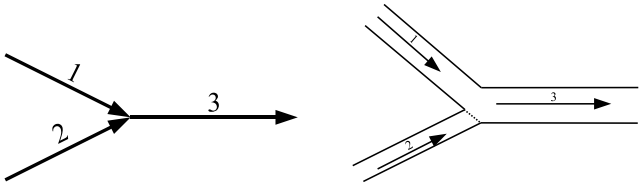
$$\bar{f}_3 = \min\{d_{13}\hat{F}_1, \frac{d_{13}}{d_{1,2}}\bar{F}_2, \bar{F}_3\}, \tag{1.45b}$$

$$\hat{f}_1 = \bar{f}_2 + \bar{f}_3, \tag{1.45c}$$

where $\hat{F}_1$ is given by (1.34) and $\bar{F}_2$ and $\bar{F}_3$ are given by (1.35). The resulting boundary densities at the junction are again given by (1.40) and (1.41).

Merging junction

At a merging junction as depicted in Figure 1.13(a), we again want to find the coupling with the maximal through-flow $\bar{f}_3(t) = \hat{f}_1(t) + \hat{f}_2(t)$. As described in [11, 17], this



(a) Merging junction. (b) Priority road.

Figure 1.13: Merging junction.

coupling condition is not uniquely solvable, if (1.37) is not fulfilled. Hence, we need to assign a further rule. The authors in [11, 17, 56] propose a *right of way* parameter $q \in]0,1[$ that prescribes the proportion of flow coming from 1 and 2 in the case of a bottleneck situation. In a similar way, a priority rule for merging junctions has been developed in [25]. Based on these approaches we formulate a priority traffic rule, where the traffic of the main road always is prioritised over the traffic of a side road as depicted in Figure 1.13(b).

As soon as road 3 has such a dense traffic, that it cannot immediately allow all incoming cars to continue, cars from road 1 are preferred. Again, we want to maximise the flow at the junction. The priorisation of the flow coming from road 1 is obtained by using a weighting parameter $w > 1$ in the objective function:

$$\max w \cdot \gamma_1 + \gamma_2 \tag{1.46a}$$

such that

$$\gamma_3 = \gamma_1 + \gamma_2 \tag{1.46b}$$

$$0 \leq \gamma_{1/2} \leq \hat{F}_{1/2} \tag{1.46c}$$

$$0 \leq \gamma_3 \leq \bar{F}_3 \tag{1.46d}$$

Lemma 1.3.5. *There exists a unique solution of (1.46) which is given by*

$$\bar{f}_3 = \min\{\hat{F}_1 + \hat{F}_2, \bar{F}_3\}, \tag{1.47a}$$

$$\hat{f}_1 = \min\{\hat{F}_1, \bar{F}_3\}, \tag{1.47b}$$

$$\hat{f}_2 = \bar{f}_3 - \hat{f}_1, \tag{1.47c}$$

where $\hat{F}_1$ and $\hat{F}_2$ are given by (1.34) and $\bar{F}_3$ is given by (1.35).

Proof. Problem (1.46) is an easy linear optimisation problem. Techniqually, it can be solved by Simplex algorithm [30]. Due to the simple structure of the problem, it is also possible to directly get the optimal solution by the consideration of different cases. We distinguish between three cases depending on the size of $\bar{F}_3$. The feasible region of each case is depicted in Figure 1.14 and the optimal solution is indicated by the black dot.

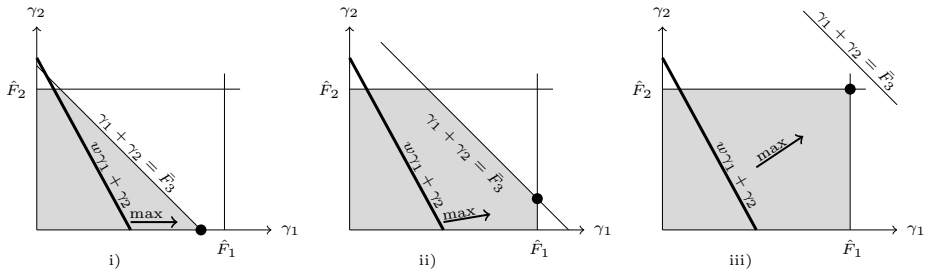


Figure 1.14: Feasible region for flows at junction.

i) $\bar{F}_3 \leq \hat{F}_1$: In this case the optimal solution of (1.46) is given by

$$\hat{f}_1 = \bar{F}_3,\ \hat{f}_2 = 0,\ \bar{f}_3 = \bar{F}_3.$$

ii) $\hat{F}_1 < \bar{F}_3 \leq \hat{F}_1 + \hat{F}_2$: In this case the optimal solution of (1.46) is given by

$$\hat{f}_1 = \hat{F}_1,\ \hat{f}_2 = \bar{F}_3 - \hat{F}_1,\ \bar{f}_3 = \bar{F}_3.$$

iii) $\bar{F}_3 > \hat{F}_1 + \hat{F}_2$: In this case the optimal solution of (1.46) is given by

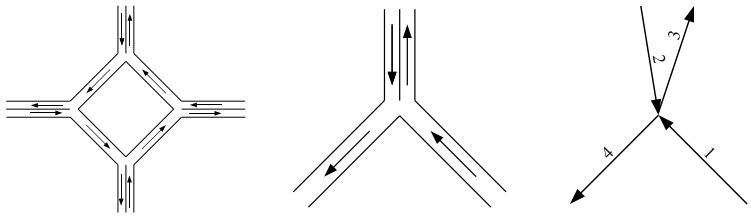
$$\hat{f}_1 = \hat{F}_1,\ \hat{f}_2 = \hat{F}_2,\ \bar{f}_3 = \hat{F}_1 + \hat{F}_2.$$

This yields directly (1.47a) - (1.47c). Hence, (1.46) is uniquely solvable. □

Roundabout

We consider a roundabout as depicted in Figure 1.15(a). It is composed of four junctions with two incoming and two outgoing roads, see 1.15(b).

This junction type is a combination of a merging junction with priority rules and dispersion junction. Since the inner ring of the roundabout has priority, road 1 is



(a) (b) (c)

Figure 1.15: modelling of a roundabout as a combination of dispersing and merging junctions.

prioritised over road 2. The dispersing is managed in the following way: We assume no car is going from road 2 to road 3, but all go to road 4. Hence, the distribution parameters are $d_{23} = 0$ and $d_{24} = 1$. Furthermore the traffic distribution from road 1 to road 3 and 4 is also prescribed by d_{13} and d_{14}, respectively.

We model the coupling condition as the optimal solution of

$$\max\ w \cdot \gamma_1 + \gamma_2 \tag{1.48a}$$

such that

$$\gamma_3 = d_{13}\gamma_1 + d_{23}\gamma_2 \tag{1.48b}$$

$$\gamma_4 = d_{14}\gamma_1 + d_{24}\gamma_2 \tag{1.48c}$$

$$0 \leq \gamma_{1/2} \leq \hat{F}_{1/2} \tag{1.48d}$$

$$0 \leq \gamma_{3/4} \leq \bar{F}_{3/4}, \tag{1.48e}$$

where $w > 1$ is a previously fixed weight.

Lemma 1.3.6. *The problem (1.48) with $d_{23} = 0$ and $d_{24} = 1$ is uniquely solvable if $d_{13} \neq 0$ and $d_{14} \neq 0$. The solution is given by:*

$$\hat{f}_1 = \min\{\hat{F}_1, \frac{1}{d_{13}}\bar{F}_3, \frac{1}{d_{14}}\bar{F}_4\}, \tag{1.49a}$$

$$\hat{f}_2 = \min\{\hat{F}_2, \bar{F}_4 - d_{14}\hat{f}_1\}, \tag{1.49b}$$

$$\bar{f}_3 = d_{13}\hat{f}_1, \tag{1.49c}$$

$$\bar{f}_4 = d_{14}\hat{f}_1 + \hat{f}_2. \tag{1.49d}$$

Proof. With the given distribution parameters $d_{23} = 0$ and $d_{24} = 1$, the optimisation problem reduces to

$$\begin{aligned} &\max\ w \cdot \gamma_1 + \gamma_2 \\ &\text{such that} \\ &\gamma_3 = d_{13}\gamma_1 \\ &\gamma_4 = d_{14}\gamma_1 + \gamma_2, \\ &0 \leq \hat{f}_{1/2} \leq \hat{F}_{1/2},\ 0 \leq \hat{f}_{3/4} \leq \bar{F}_{3/4}, \end{aligned} \tag{1.50}$$

Since (1.50) is a linear programming problem, it can be solved by the Simplex algorithm [30]. It is possible to reduce the unknowns, by reformulating the constraints. In this way we get rid of the variables γ_3 and γ_4:

$$\begin{array}{rll} \max w\gamma_1 & +\gamma_2 & \\ \text{such that} & & \\ \gamma_1 & & \leq b \\ d_{14}\gamma_1 & +\gamma_2 & \leq \bar{F}_4 \\ & \gamma_2 & \leq \hat{F}_2 \\ & \gamma_1, \gamma_2 & \geq 0 \end{array} \tag{1.51}$$

where b is a known parameter given by

$$b := \min\{\frac{\bar{F}_3}{d_{13}}, \hat{F}_1\}, \quad d_{13} \neq 0. \tag{1.52}$$

Be aware that γ_i are variables which are up to optimisation while F_i are fix parameters representing the upper bounds. We introduce slack variables $s_i \geq 0$, $i = 1, \ldots, 3$ and rewrite (1.51). This yields

$$\begin{array}{rrrrrrr} z & -w\gamma_1 & -\gamma_2 & & & & = 0 \\ & \gamma_1 & & +s_1 & & & = b \\ & d_{14}\gamma_1 & +\gamma_2 & & +s_2 & & = \bar{F}_4 \\ & & \gamma_2 & & & +s_3 & = \hat{F}_2, \end{array} \tag{1.53}$$

where z represents the objective function value. First, we want γ_1 to enter the basis. According to rules used by the Simplex algorithm, we have to pivot the row where the ratio between the right hand side and the entering variable coefficient is minimal. In our case, we have to find the minimum of b and $\frac{\bar{F}_4}{d_{14}}$. Hence, we distinguish two cases.

Case i) $\frac{\bar{F}_4}{d_{14}} \leq b$

In this case the next transformation yields

$$\begin{array}{rrrrrrl} z & & +(\frac{w}{d_{14}}-1)\gamma_2 & & +\frac{w}{d_{14}}s_2 & & = \frac{w}{d_{14}}\bar{F}_4 \\ & & -\frac{1}{d_{14}}\gamma_2 & +s_1 & -\frac{1}{d_{14}}s_2 & & = b - \frac{\bar{F}_4}{d_{14}} \\ & \gamma_1 & +\frac{1}{d_{14}}\gamma_2 & & +\frac{1}{d_{14}}s_2 & & = \frac{\bar{F}_4}{d_{14}} \\ & & \gamma_2 & & & +s_3 & = \hat{F}_2. \end{array} \tag{1.54}$$

Since $w > 1$ and $d_{14} \leq 1$, we know that $\frac{w}{d_{14}} - 1 > 0$. Hence, all coefficients in the first row are positive. Thus, the basic solution of (1.54) is optimal, with $s_2 = \gamma_2 = 0$ and $\gamma_1 = \frac{1}{d_{14}}\bar{F}_4$, which corresponds to (1.49).

Case ii) $b \leq \frac{\bar{F}_4}{d_{14}}$

In this case the first Simplex transformation leads to

$$\begin{array}{rrrrrl} z & -\gamma_2 & +ws_1 & & & = wb \\ \gamma_1 & & +s_1 & & & = b \\ & \gamma_2 & -d_{14}s_1 & +s_2 & & = \bar{F}_4 - d_{14}b \\ & \gamma_2 & & & +s_3 & = \hat{F}_2 \end{array} \tag{1.55}$$

γ_2 has a negative coefficient in the first row. Hence, the basic solution is not optimal. We have to transform the system a second time such that γ_2 enters the basis as well. In order to pivot the row with minimal ratio between right hand side and coefficient of the entering variable, again two different cases have to be considered.

Case iia) $\bar{F}_4 - d_{14}b \leq \hat{F}_2$

The second Simplex transformation yields

$$\begin{array}{rrrrrl} z & & +(w-d_{14})s_1 & +s_2 & & = (w-d_{14})b + \bar{F}_4 \\ \gamma_1 & & +s_1 & & & = b \\ & \gamma_2 & -d_{14}s_1 & +s_2 & & = \bar{F}_4 - d_{14}b \\ & & d_{14}s_1 & -s_2 & +s_3 & = \hat{F}_2 - \bar{F}_4 + d_{14}b. \end{array} \tag{1.56}$$

Because $w > 1$ and $d_{14} \leq 1$, all coefficients in the first row are positive. Hence, the basic solution with $s_1 = s_2 = 0$, $\gamma_1 = b$ and $\gamma_2 = \bar{F}_4 - d_{14}b$ is optimal and fulfills (1.49).

Case iib) $\hat{F}_2 \leq \bar{F}_4 - d_{14}b$

In this case the next transformation of the system (1.55) looks like

$$\begin{array}{cccccc} z & & +ws_1 & & +s_3 & = wb + \hat{F}_2 \\ & \gamma_1 & +s_1 & & & = b \\ & & -d_{14}s_1 & +s_2 & -s_3 & = \bar{F}_4 - \hat{F}_2 - d_{14}b \\ & & & \gamma_2 & +s_3 & = \hat{F}_2 \end{array} \tag{1.57}$$

All coefficients of the first row of (1.57) are positive. Hence, the basic solution is given by $s_1 = s_3 = 0$, $\gamma_1 = b$ and $\gamma_2 = \hat{F}_2$.

These cases cover all possibilities and proof the claim. □

Remark 1.3.7. *Note, that in [11, 12, 19] the considered distribution parameters d_{ij} are strictly larger than zero and strictly smaller than 1. The proof of Lemma 1.3.6 especially considers the case, where $d_{23} = 0$ and $d_{24} = 1$. However, (1.48) can be solved analogously for different traffic distribution settings.*

In Chapter 2, Section 2.4, optimisation problems are considered. In particular, we derive a model for traffic light settings. Since the traffic lights lead to further restrictions on the outgoing traffic flow, they allow for the modelling of even more complex junctions, see Figure 2.10.

Remark 1.3.8. *In summary, following the notation for DTN models of Definition 1.1.2, a traffic flow network model is for example given by*

- *directed graph $G = (V, E)$, with at most two incoming and two outgoing edges per vertex (containing only the four junction types derived before),*
- *time horizon $t \in [0, T]$,*
- *valid flow distribution matrix d,*
- *edge properties $\mathcal{P}_i = \left(L_i,\, \rho_i^*,\, \lambda_i\right)^T \; \forall i \in E$,*
- *dynamic functions $\mathcal{D}_i = \left(\rho_i,\, \hat{\rho}_i,\, \bar{\rho}_i,\, f_i,\, \hat{f}_i,\, \bar{f}_i,\, \hat{F}_i,\, \bar{F}_i\right)^T \; \forall i \in E$,*
- *boundary conditions $\mathcal{B}_i = \left(\rho_i(x = 0, t)\right) \; \forall i \in E^{in}$,*
- *edge operators, including equations for $f(\rho)$, e.g. (1.26), $\hat{F}_i$ and $\bar{F}_i$, cf. (1.34) and (1.35), $\hat{\rho}_i$ and $\bar{\rho}_i$, cf. (1.40) and (1.41). Furthermore, we need:*

$$\mathcal{I}_i(\rho_i,\, \partial_t\rho_i,\, f,\, \partial_t f) = \partial_t\rho_i + \partial_x f(\rho),\; \forall i \in E.$$

- *coupling operators for every junction type:*

  1) *simply connected roads:* $\forall v \in \{V : |\delta_v^{in}| = 1 \wedge |\delta_v^{out}| = 1\}$.
  Let $i \in \delta_v^{in}$ *be the ingoing edge and* $j \in \delta_v^{out}$ *the outgoing edge of* v.

$$\mathfrak{C}_j^{(1j)}(\bar{f}_j, \bar{F}_j, \hat{F}_i) = \bar{f}_j - \min\{\bar{F}_j, \hat{F}_i\}$$
$$\mathfrak{C}_i^{(1i)}(\bar{f}_j, \hat{f}_i) = \hat{f}_i - \bar{f}_j.$$

  2) *dispersing junctions:* $\forall v \in \{V : |\delta_v^{in}| = 1 \wedge |\delta_v^{out}| = 2\}$.
  Let $i \in \delta_v^{in}$ *be the ingoing edge and* $j, k \in \delta_v^{out}$ *the outgoing edges of* v.

$$\mathfrak{C}_j^{(2j)}(d, \bar{f}_j, \bar{F}_j, \bar{F}_k, \hat{F}_i) = \bar{f}_j - \min\{d_{ij}\hat{F}_i, \bar{F}_j, \frac{d_{ij}}{d_{ik}}\bar{F}_k\}$$
$$\mathfrak{C}_k^{(2k)}(d, \bar{f}_k, \bar{F}_j, \bar{F}_k, \hat{F}_i) = \bar{f}_k - \min\{d_{ik}\hat{F}_i, \bar{F}_k, \frac{d_{ik}}{d_{ij}}\bar{F}_j\}$$
$$\mathfrak{C}_i^{(2i)}(\bar{f}_j, \bar{f}_k, \hat{f}_i) = \hat{f}_i - \bar{f}_j - \bar{f}_k.$$

  3) *merging junctions:* $\forall v \in \{V : |\delta_v^{in}| = 2 \wedge |\delta_v^{out}| = 1\}$.
  Let $i, k \in \delta_v^{in}$ *be the ingoing edges of* v*, where* i *is the priority road, and* $j \in \delta_v^{out}$ *the outgoing edge of* v.

$$\mathfrak{C}_j^{(3j)}(\bar{f}_j, \bar{F}_j, \hat{F}_k, \hat{F}_i) = \bar{f}_j - \min\{\hat{F}_i + \hat{F}_k, \bar{F}_j\}$$
$$\mathfrak{C}_i^{(3i)}(\hat{f}_i, \bar{F}_j, \hat{F}_i) = \hat{f}_i - \min\{\hat{F}_i, \bar{F}_j\}$$
$$\mathfrak{C}_k^{(3k)}(\bar{f}_j, \hat{f}_i, \hat{f}_k) = \hat{f}_k + \hat{f}_i - \bar{f}_j.$$

  4) *combined junctions:* $\forall v \in \{V : |\delta_v^{in}| = 2 \wedge |\delta_v^{out}| = 2\}$.
  Let $i, k \in \delta_v^{in}$ *be the ingoing edges of* v*, where* i *is the priority road, and* $j, l \in \delta_v^{out}$ *the outgoing edges of* v*, with* $d_{k,j} = 0$ *and* $d_{k,l} = 1$.

$$\mathfrak{C}_i^{(4i)}(d, \hat{f}_i, \hat{F}_i, \bar{F}_j, \bar{F}_l) = \hat{f}_i - \min\{\hat{F}_i, \frac{1}{d_{ij}}\bar{F}_j, \frac{1}{d_{il}}\bar{F}_l\}$$
$$\mathfrak{C}_k^{(4k)}(d, \hat{f}_i, \hat{f}_k, \hat{F}_k, \bar{F}_l) = \hat{f}_k - \min\{\hat{F}_k, \bar{F}_l - d_{il}\hat{f}_i\}$$
$$\mathfrak{C}_j^{(4j)}(d, \bar{f}_j, \hat{f}_i) = \bar{f}_j - d_{ij}\hat{f}_i$$
$$\mathfrak{C}_l^{(4l)}(d, \bar{f}_l, \hat{f}_i, \hat{f}_k) = \bar{f}_l - d_{il}\hat{f}_i - \hat{f}_k.$$

- *edge conditions* $\mathfrak{I}_i \equiv 0$, $\forall i \in E$ *and*
- *coupling conditions* $\mathfrak{C}_i^{(1/2/3/4i)} \equiv 0$, $\forall i \in E$ *(choosing the matching condition for start and end point of each road).*

1.3.4 Transformation into Hamilton-Jacobi Formulation

As in [79, 80], the traffic network model in Section 1.3.1 can be interpreted as Hamilton-Jacobi equations. This formulation has also been studied in the engineering context in [27, 28]. This approach has the advantage that trajectories of cars can be easily derived from it. For that reason, Hamilton-Jacobi equations have been used for example for data-assimilation models [18]. Recent analysis has been done to extend the Hamilton-Jacobi formulation to the network case, see [1, 97].

In this section, we resume the connection between the LWR-equations (1.24) and the Hamilton-Jacobi formulation (1.58). Later on, in Subsection 1.4.2, we apply a numerical Scheme and derive an algorithm to simulate traffic flow via Hamilton-Jacobi equations on road networks.

A Hamilton-Jacobi equation with Hamiltonian f is given by

$$M_t(x,t) + f(M_x(x,t)) = 0. \tag{1.58}$$

Remark 1.3.9. *If we consider roads on which vehicles cannot overtake, it is possible to number them according to the order they pass a certain point of the road. In [12, 78, 80, 83] a continuous function is considered, where the space-time trajectory of each car is given by its the integer contour curves.*

In detail, if we start counting with the foremost car at time $t = t_0$ we get

$$N(x,t_0) = \int_x^L \rho(x',t_0)dx'$$

and for a general point in time t, the car number at (x,t) is given by

$$N(x,t) = \int_x^L \rho(x',t)dx' + N(L,t) = N(0,t) - \int_0^x \rho(x',t)dx'. \tag{1.59}$$

where the value of the left boundary is given by

$$N(0,t) = \int_{t_0}^t f(\rho(x,t')dt'.$$

Consequently, the curve

$$\{(x,t) : N(x,t) = n\}$$

describes the trajectory of the n^{th} car.

Depending on the scaling of ρ, n needs to be multiplied by a constant to yield an integer number.

According to the notation used here, the function $M : (x,t) \mapsto -N(x,t)$ is considered. Assuming sufficient regularity, we obtain from (1.59) that $M_x(x,t) = \rho(x,t)$, $\forall (x,t) \in [0,L] \times [t_0, +\infty)$. From (1.58), we can also derive that the continuity equation used in the LWR-model (1.24) holds. Differentiation of (1.58) with respect to x yields:

$$0 = M_{tx} + f(M_x)_x = M_{xt} + f(M_x)_x.$$

Consequently, if we find an M that satisfies (1.58), $\rho := M_x$ also satisfies (1.24). On traffic problems we have $\rho \geq 0$, hence M is monotonically increasing in x.

Extension to the Network Case

For the network model, we provide an additional index indicating the road $i \in \{1, ..., |E|\}$. The coupling conditions in terms of M_i are of Neumann type:

$$\forall i \in E \begin{cases} \partial_t M_i + f(\partial_x M_i) = 0 & \\ \partial_x M_i(x,0) = \rho_0(x) & \text{initial condition} \\ \partial_t M_i(0,t) = \partial_x \bar{M}_i(t) = \bar{\rho}_i(t) & \text{left boundary condition} \\ \partial_t M_i(L_i,t) = \partial_x \hat{M}_i(t) = \hat{\rho}_i(t) & \text{right boundary condition,} \end{cases} \tag{1.60}$$

where the boundaries $\bar{\rho}_i$ and $\hat{\rho}_i$ are given by the coupling of junctions, see Section 1.3.2, (1.40) and (1.41), computing the flow depending on the type of junction given. An algorithm to simulate traffic flows on networks using Hamilton-Jacobi equations is derived in Section 1.4.2.

Remark 1.3.10. *A trajectory of a car can be tracked, when its location at a certain point in time and its path through the network is known. On every road the contour lines of M describe the car trajectories, which then possibly changes to another value after having crossed a junction. An example is given in Chapter 3, Figure 3.30.*

1.4 Discretisation

In order to be able to simulate scenarios modelled by DTNs, we first have to discretise the differential equations and apply numerical schemes. There is a wide range of schemes with different properties and of different rates of convergence. For a detailed overview of schemes on hyperbolic differential equations, such as conservation laws of the form (1.5), we refer to [73]. For our purposes it is sufficient to work with first order

schemes. They have advantageous properties such as being TVD and are often easy to linearise. The latter is of importance for optimisation purposes, since it allows the schemes to be integrated into DTN-MIPs, as explained in Chapter 2.

This section gives a brief overview over several discretisation schemes which are used within this work. The first part discusses several techniques to discretise the continuity equation (1.5). The second part contains the derivation of a complete algorithm particularly created to solve the Hamilton-Jacobi traffic network model (1.60).

In the following we only consider a single edge i of a DTN.

Let $L_i \in \mathbb{R}^+$ be the length of the edge i and $T \in \mathbb{R}^+$ the considered time horizon. We introduce a discrete time grid $\mathcal{T} = \{t : t = 0, \dots, n_t\}$ with time step size Δt and number of time steps $n_t := \lceil \frac{t}{\Delta t} \rceil$. Furthermore, we work with a discrete spatial grid given by $\mathcal{K} = \{k : k = 0, \dots, n_i\}$, where the spatial size is referred to as Δx and the number of space steps is given $n_i := \lceil \frac{L_i}{\Delta t} \rceil$. We will work with equal step sizes Δ_t on the whole network. The space steps n_i per edge can differ from each other depending on the edge length L_i. Since we first consider the discretisation for one edge, we omit the index i in the sequel for the sake of readability.

We use a discrete set of variables containing a subscript indicating the space step and a superscript referring to the time step. For example, the discrete density variable ρ_k^t represents the density value at location $k \cdot \Delta x$ at time $t \cdot \Delta t$.

1.4.1 Schemes for Conservation Laws

In this subsection, we give a rough outline of some first order numerical schemes based on the idea of finite differences to approximate the derivative. This means that we approximate $\partial_t \rho(x,t)$ by its difference quotient, i.e.

$$\partial_t \rho(x,t) \approx \frac{\rho(x, t+\Delta t) - \rho(x,t)}{\Delta t},$$

and so on.

In the sequel we mention those schemes that we use in the course of this work to discretise the continuity equation (1.5) of DTNs, which is only a small part of all existing schemes. For a thorough overview and more details on numerical schemes for conservation laws, see [73].

Upwind. In most models of dynamic transportation networks, the flow moves only in one direction at each edge. In many production network models the particles are assumed to move with constant velocity v, i.e $\rho(x,t) = \rho(x - vt, 0)$. In these cases we model the density evolution along the edges with the advection equation (1.16), as described in Subsection 1.2.1. Here, it is reasonable to use the first order Upwind Scheme for discretisation. It is a scheme that only uses information of one side of the spatial grid. In our case, the velocity $v > 0$ is given. Hence, we know the direction of information. For that reason we only take the values on the left of the considered grid point into account, as shown by the discretisation stencil depicted in Figure 1.16.

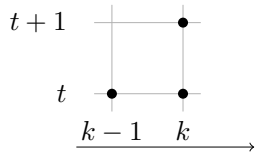


Figure 1.16: Stencil of Upwind Scheme.

The next time iteration for the density value for all inner grid points is given by

$$\rho_k^{t+1} = \rho_k^t - v \cdot \frac{\Delta t}{\Delta x} \cdot (\rho_k^t - \rho_{k-1}^t). \tag{1.61}$$

In [73] it is proven that (1.61) converges to the exact solution in first order, when Δt and Δx tend to zero and when the grid sizes are chosen such that the CFL condition

$$\max_{\rho \in [0, \rho^{max}]} |f'(\rho)| \cdot \frac{\Delta t}{\Delta x} \leq 1, \tag{1.62}$$

holds.

The CFL-conditions is an abbreviation for Courant-Friedrichs-Lewy condition. It ensures that the time step size is fine enough to capture all information that is transported. The so called grid-velocity is given by $\frac{\Delta x}{\Delta t}$ and has to be greater or equal than the velocity of information of the analytical solution $f'(\rho)$. In case of the advection equation (1.16) we have $f(\rho) = v \cdot \rho$; hence the speed of information is constant and given by v.

Supply chain models using advection equation and applying the Upwind discretisation are examined in [36, 37, 48, 49, 57, 98], amongst others.

Lax-Friedrichs Scheme. Another easy, straight-forward first order scheme is the Lax-Friedrichs Scheme. It is a central scheme and hence, takes information coming from the left and from the right side of the considered grid point into account. This is depicted in Figure 1.17.

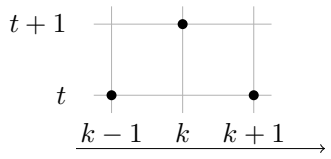


Figure 1.17: Stencil of the Lax-Friedrichs Scheme.

This is important for models, which allow for forwards and backwards travelling density waves, such as the LWR-traffic model (1.24). To guarantee stable numerical simulations, the discretisation grids have to respect the CFL condition (1.62).

The evolution of the density for all inner grid points is computed in the following way:

$$\rho_k^{t+1} = \frac{1}{2}(\rho_{k+1}^t + \rho_{k-1}^t) - \frac{\Delta t}{2\Delta x}(f(\rho_{k+1}^t) - f(\rho_{k-1}^t)). \tag{1.63}$$

The commonly known disadvantage of the scheme is its diffusivity. This leads to dispersion effects appearing during the simulation of shock waves, see for example [10, 47]. However, for certain relations of grid sizes and parameter settings, these effects are minimal, see Lemma 1.4.2. For these settings the scheme becomes attractive due to its simplicity and linear appearance.

Staggered Lax-Friedrichs Scheme. As we will see later in Subsection 2.4.4, for the coupling of two roads it is advantageous to use a numerical scheme that does not incorporate the boundary values in terms of density, but only in terms of the flow.

The staggered Lax-Friedrichs Scheme, introduced in [61] and further developed and applied by [65], fulfills exactly these requirements. In addition to that, it is less diffusive than the standard Lax-Friedrichs Scheme.

The main idea is to use a staggered grid, see Figure 1.19, as intermediate step. The staggered density values are obtained by averaging over the neighbouring densities. Then centred differences are used with respect to the original grid points, that are

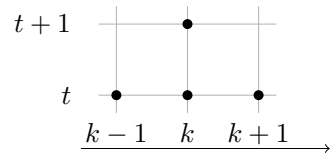


Figure 1.18: Stencil of staggered Lax-Friedrichs Scheme.

located in half a step size, i.e. $\frac{1}{2}\Delta x$, distance to the considered point. Finally, the values are projected back to the original grid. Due to the fact that half step sizes are used, the grid sizes have to fulfill CFL/2, i.e.

$$\Delta t \leq \frac{\Delta x}{2 \cdot \max_{\rho \in [0,\rho^{max}]} |f'(\rho)|}., \tag{1.64}$$

to guarantee convergence.

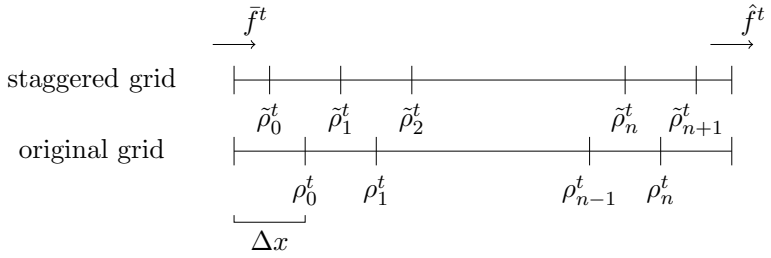


Figure 1.19: Staggered grid.

The detailed derivation of the scheme is as follows:

Step 1. Compute the values of the staggered grid as averaged values of the neighbouring original density values:

$$\text{left side:} \qquad \tilde{\rho}_0^t = \rho_0^t \tag{1.65a}$$

$$\text{central points:} \qquad \tilde{\rho}_k^t = \frac{1}{2}(\rho_{k-1}^t + \rho_k^t),\ \forall k = 1, \ldots, n \tag{1.65b}$$

$$\text{right side:} \qquad \tilde{\rho}_{n+1}^t = \rho_n^t \tag{1.65c}$$

Step 2. $t \to t+1$ (Time evolution of the staggered values using centered differences with

respect to the original grid points):

$$\text{left side:} \quad \tilde{\rho}_0^{t+1} = \tilde{\rho}_0^t - \frac{\Delta t}{\frac{1}{2}\Delta x}(f(\rho_0^t) - \bar{f}^t) \tag{1.66a}$$

$$\text{central points:} \quad \tilde{\rho}_k^{t+1} = \tilde{\rho}_k^t - \frac{\Delta t}{\Delta x}(f(\rho_k)^t - f(\rho_{k-1}^t)),\ \forall k = 1, \dots, n \tag{1.66b}$$

$$\text{right side:} \quad \tilde{\rho}_{n+1}^{t+1} = \rho_{n+1}^t - \frac{\Delta t}{\frac{1}{2}\Delta x}(\hat{f}^t - f(\rho_n^t)) \tag{1.66c}$$

Step 3. Project the solution back to the original grid:

$$\begin{aligned}\text{left side:} \quad \rho_0^{t+1} &= \frac{1}{2}(\tilde{\rho}_0^{t+1} + \tilde{\rho}_1^{t+1}) \\ &\overset{(1.66a)}{=} \frac{1}{2}\tilde{\rho}_0^t - \frac{\Delta t}{\Delta x}(f(\rho_0^t) - \bar{f}) \\ &\quad + \frac{1}{2}\tilde{\rho}_1^t - \frac{\Delta t}{2\Delta x}(f(\rho_1^t) - f(\rho_0^t))\end{aligned} \tag{1.67a}$$

$$\begin{aligned}\text{central points:} \quad \rho_k^{t+1} &= \frac{1}{2}(\tilde{\rho}_k^{t+1} + \tilde{\rho}_{k+1}^{t+1}) \\ &\overset{(1.66b)}{=} \frac{1}{2}\tilde{\rho}_k^t - \frac{\Delta t}{2\Delta x}(f(\rho_k^t) - f(\rho_{k-1}^t)) \\ &\quad + \frac{1}{2}\tilde{\rho}_{k+1}^t - \frac{\Delta t}{2\Delta x}(f(\rho_{k+1}^t) - f(\rho_k^t))\end{aligned} \tag{1.67b}$$

$$\begin{aligned}\text{right side:} \quad \rho_n^{t+1} &= \frac{1}{2}(\tilde{\rho}_n^{t+1} + \tilde{\rho}_{n+1}^{t+1}) \\ &\overset{(1.66c)}{=} \frac{1}{2}\tilde{\rho}_n^t - \frac{\Delta t}{2\Delta x}(f(\rho_n^t) - f(\rho_{n-1}^t)) \\ &\quad + \frac{1}{2}\tilde{\rho}_{n+1}^t - \frac{\Delta t}{\Delta x}(\hat{f}^t - f(\rho_n^t))\end{aligned} \tag{1.67c}$$

Finally, applying again (1.65a) to (1.65c) we end up with the following scheme:

$$\text{left side:} \quad \rho_0^{t+1} = \frac{1}{4}(3\rho_0^t + \rho_1^t) - \frac{\Delta t}{2\Delta x}(f(\rho_1^t) + f(\rho_0^t) - 2\bar{f}^t) \tag{1.68a}$$

$$\text{central points:} \quad \rho_k^{t+1} = \frac{1}{4}(\rho_{k-1}^t + 2\rho_k^t + \rho_{k+1}^t) - \frac{\Delta t}{2\Delta x}(f(\rho_{k+1}^t) - f(\rho_{k-1}^t)),\ \forall k = 1, \dots, n \tag{1.68b}$$

$$\text{right side:} \quad \rho_n^{t+1} = \frac{1}{4}(\rho_{n-1}^t + 3\rho_n^t) - \frac{\Delta t}{2\Delta x}(2\hat{f}^t - f(\rho_n)^t - f(\rho_{n-1}^t)) \tag{1.68c}$$

The computation of the outer cells (1.68a) and (1.68c) only involve flow values at the boundaries $\bar{f}_i^t$ and $\hat{f}_i^t$ and not the boundary density. As we will see later in Subsection 2.4.4, this is advantageous for the linearisation process to transform the model into a linear mixed integer optimisation problem, because it saves us a complicated

linearisation process of the coupling density values $\bar{\rho}_i^t$ and $\hat{\rho}_i^t$ at the junctions. Due to other available values, such as the maximal possible coupling flow $\bar{F}_i^t$ and $\hat{F}_i^t$, introduced in the next chapter, we are equipped with sufficient information to capture the entire situation at the junction.

However, on the outer edges of the network, E^{in} and E^{out}, it is necessary to integrate the boundary density values, since the computation from the flow down to the corresponding density is not unique. In order to avoid numerical instabilities at the outer boundaries, we reformulate the discretisation scheme of the outermost cells including the outer boundary densities $\rho_{i,lb}$, $\forall i \in E^{in}$ and $\rho_{i,rb}$, $\forall j \in E^{out}$.

Discretisation for edges without predecessors – $i \in E^{in}$ (inflow edges into the network) – is given by

- leftmost cell:

$$\rho_{0,i}^{t+1} = \frac{1}{4}(\rho_{i,lb}^t + 2\rho_{0,i}^t + \rho_{1,i}^t) - \frac{\Delta t}{2\Delta x}(f(\rho_{1,i}^t) - f(\rho_{i,lb}^t)),\ \forall i \in E^{in} \tag{1.69}$$

- inner cells and rightmost cell as above.

Discretisation for edges without successors – $i \in E^{out}$ (outflow edges of the network) – is given by

- rightmost cell:

$$\rho_{n,i}^{t+1} = \frac{1}{4}(\rho_{n-1,i}^t + 2\rho_{n,i}^t + \rho_{i,rb}^t) - \frac{\Delta t}{2\Delta x}(f(\rho_{i,rb}^t) - f(\rho_{n-1,i}^t)),\ \forall i \in E^{out} \tag{1.70}$$

- inner cells and leftmost cell as above

Godunov Scheme. The Godunov Scheme first appeared in [45] and became one of the most popular schemes for solving hyperbolic partial differential equations. It is a first order scheme that bases on the idea to solve Riemann problems (1.29) of neighbouring cells for each time iteration. The used grid-points for one iteration step is the same as for the staggered Lax-Friedrichs Scheme, cf. Figure 1.20.

To obtain reliable results, the choice of the grids must again fulfill the CFL-condition (1.62).

- The initial values ρ_k^0 are given by the mean value of the grid cell.

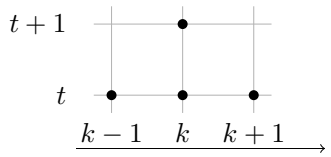


Figure 1.20: Stencil of Godunov Scheme.

- In each time iterate $t \in \mathcal{T}$, we imagine the values ρ_k^t as a piecewise constant functions on the space-grid and solve the corresponding Riemann problem for one time step. Using a concave flow function f yield the following cases for the midpoints of the cells:

  - If $(f'(\rho_k^t) \geq 0 \wedge f'(\rho_{k+1}^t) \geq 0)$

    $$\rightarrow \rho_{k+\frac{1}{2}}^t = \rho_k^t$$

  - If $(f'(\rho_k^t) \geq 0 \wedge f'(\rho_{k+1}^t) < 0)$

    $$\rightarrow s = \frac{f(\rho_{k+1}^t - f(\rho_k^t)}{\rho_{k+1}^t - \rho_k^t},\ \rho_{k+\frac{1}{2}}^t = \begin{cases} \rho_k^i, & \text{if } s \geq 0 \\ \rho_{k+1}^i, & \text{else} \end{cases}$$

  - If $(f'(\rho_k^t) < 0 \wedge f'(\rho_{k+1}^t) < 0)$

    $$\rightarrow \rho_{k+\frac{1}{2}}^t = \rho_{k+1}^t$$

  - If $(f'(\rho_k^t)) < 0 \wedge f'(\rho_{k+1}^t) \geq 0)$

    $$\rightarrow \rho_{k+\frac{1}{2}}^t = \rho^*$$

- The density for the next time step is then given by

  $$\rho_k^{t+1} = \rho_k^t - \frac{\Delta t}{\Delta x}(f(\rho_{k+\frac{1}{2}}^t) - f(\rho_{k-\frac{1}{2}}^t))$$

  for all k.

- Repeat these steps for all time steps $t \in \mathcal{T}$.

1.4.2 Hamilton-Jacobi Scheme

We discuss how to couple the Hamilton-Jacobi formulation for road networks as described in Subsection 1.3.4 at road intersections and derive a reliable algorithm combining the coupling conditions with a numerical scheme for the Hamilton-Jacobi equations [67].

We assume f to be a concave flow function with unique maximum. We introduce a space and time grid, as described in the beginning of this section. The time grid size Δt is set according to the CFL-condition (1.62).

Before we consider the network case, we stick to a single road. Whenever the context is clear, we drop the first sub-index indicating the road on the network for the sake of readability. Hence, the remaining subindex denotes the space step on the road.

Note, that the grid points of M are shifted by $\frac{\Delta x}{2}$ compared to the grid of ρ. Here, $M_j^t = M(x_j, t \cdot \Delta t)$, where $x_j = (j - \frac{1}{2})\Delta x$ and $j = \{0, n_x + 1\}$. For the discretisation of the Hamilton-Jacobi equations we use the central one-dimensional first order scheme derived in [67]. The time evolution of M at the inner grid points is computed as follow:

$$\begin{aligned} M_j^{t+1} =& M_j^t - \frac{\Delta t}{2}\left[f\Big(\frac{M_{j+1}^t - M_j^t}{\Delta x}\Big) + f\Big(\frac{M_j^t - M_{j-1}^t}{\Delta x}\Big)\right] \\ &+ \frac{\Delta t}{2\Delta x} a_j^t (M_{j+1}^t - 2M_j^t + M_{j-1}^t) \end{aligned} \tag{1.71}$$

with

$$a_j^n \geq \max_{x \in [(j-1)\Delta x, (j+1)\Delta x]} |f'(M_x)|.$$

The coupling is done in terms of densities. Therefore we need to approximate the derivative of M close to the junction. This is done via finite differences:

$$\partial_x M_{j+\frac{1}{2}}^t =: \rho_i^t = \frac{M_{j+1}^t - M_j^t}{\Delta x}. \tag{1.72}$$

This scheme is strongly related to the Lax-Friedrichs Scheme (1.63), see Lemma 1.4.1.

Discretisation of the boundary condition

Due to dispersion effects of the discretisation scheme, cf. [73], the coupling is not always captured in the correct way. Therefore we need to introduce suitable ghost-cells added on both ends of each road. The wave fronts travel along the roads until they reach the

next junction, providing the coupling routine with information about the new density values on the road. The waves run through these artificial cells, but use the value at the road boundary to compute the coupling condition, as depicted in Figure 1.22 on page 56. This leads to the correct density information at the boundary. A more detailed explanation of the algorithm can be found later on starting from page 53.

This method only works, when the number of ghost-cells is large enough to absorb the whole amplitude of the front dissipation. Consequently, it is necessary to know the number of required ghost-cells related to the possible dispersion amplitude of the wave front.

In the sequel we will show that two ghost-cells on each side of the roads are sufficient for settings with a specific flow function, a certain correlation between space and time grid size and a certain choice of parameter a_j^n for the Hamilton-Jacobi scheme, see (1.71). Note that for higher-order schemes more ghost cells might be required.

Lemma 1.4.1. *If the parameter a_j^n of the Hamilton-Jacobi Scheme (1.71) is set to*

$$a_j^n := \max_{\rho} |f'(\rho)|, \ \forall j, n \tag{1.73}$$

and the time grid Δt is set to the maximal possible value satisfying the CFL-condition (1.62), then the Hamilton-Jacobi Scheme (1.71) is equivalent to the Lax-Friedrichs Scheme (1.63).

Proof. Scheme (1.71) and equation (1.72) allow for the following calculation:

$$\begin{aligned}
\frac{\rho_i^{t+1} - \rho_i^t}{\Delta t} &\overset{(1.72)}{=} \Big(\frac{M_{j+1}^{t+1} - M_{j+1}^t}{\Delta t \cdot \Delta x}\Big) - \Big(\frac{M_j^{t+1} - M_j^t}{\Delta t \cdot \Delta x}\Big) \\
&\overset{(1.71)}{=} -\frac{1}{2\Delta x}\left[f\Big(\frac{M_{j+2}^t - M_{j+1}^t}{\Delta x}\Big) - f\Big(\frac{M_j^t - M_{j-1}^t}{\Delta x}\Big)\right] \\
&\quad + \frac{a}{2\Delta x}\Big(\frac{M_{j+2}^t - M_{j+1}^t}{\Delta x} - 2 \cdot \frac{M_{j+1}^t - M_j^t}{\Delta x} + \frac{M_j^t - M_{j-1}^t}{\Delta x}\Big) \\
&\overset{(1.72)}{=} -\frac{1}{2\Delta x}\Big[f(\rho_{i+1}^t) - f(\rho_{i-1}^t)\Big] + \frac{a}{2\Delta x}\Big[\rho_{i+1}^t - 2\rho_i^t + \rho_{i-1}^t\Big] \\
\overset{(1.73)\&(1.74)}{\Longleftrightarrow} \rho_i^{t+1} &= \frac{1}{2}\Big(\rho_{i+1}^t + \rho_{i-1}^t\Big) - \frac{\Delta t}{2\Delta x} \cdot \Big[f(\rho_{i+1}^t) - f(\rho_{i-1}^t)\Big],
\end{aligned}$$

which is exactly the Lax-Friedrichs Scheme (1.63).

□

The next lemma shows that two ghost-cells are sufficient to capture the dispersion amplitude of wave fronts in the scheme for a certain parameter setting.

Lemma 1.4.2. *Let a road describing the traffic flow with a triangular flow function as in (1.26) be given. Assume that the traffic density evolution is described by:* $\partial_t \rho + \partial_x f(\rho) = 0, \forall t \in [0,T]$, $x \in [0,L]$. *The density at time* t *is piecewise constant. i.e.* $\exists \hat{x} \in [0,L]$ *with*

$$\rho(x,t) = \begin{cases} l, & x \leq \hat{x} \\ r, & x > \hat{x}. \end{cases}$$

Then, using Lax-Friedrich-discretisation, with

$$\Delta t := \frac{\Delta x}{\max_{\rho \in [0,\rho^{max}]} |f'(\rho)|} = \frac{\Delta x}{\lambda}, \tag{1.74}$$

the dispersion over time of the wave front will not exceed two grid points.

Proof. The space-grid is given such that the discontinuity of the initial condition is located between grid point i and grid point $i+1$. Hence, the density values around the discontinuity at time-step t are given by:

$$\rho^t = (l, \ldots, \underset{i-1}{l}, \underset{i}{l}, \underset{i+1}{r}, \underset{i+2}{r}, \ldots, r).$$

The Lax-Friedrichs Scheme preserves the density values inside the constant regions, because (1.63) yields

$$\text{if } \rho^t_{j-1} = \rho^t_{j+1} \;\Rightarrow\; \rho^{t+1}_j = \rho^t_{j-1}$$

for an arbitrary space-grid point j. Hence, it is sufficient to consider the density evolution next to the discontinuity. For this purpose we distinguish several cases:

Case 1: $l \in [0,\rho^*] \;\wedge\; r \in [0,\rho^*]$:

Applying (1.63) to ρ^t, we get

$$\rho^{t+1} = (l, \ldots, \underset{i-1}{l}, \underset{i}{l}, \underset{i+1}{l}, \underset{i+2}{r}, \ldots, r),$$

Hence, we get a sharp forward travelling front without any dispersion.

Case 2: $l \in [\rho^*,\rho^{max}] \;\wedge\; r \in [\rho^*,\rho^{max}]$:

Applying (1.63) to ρ^t, we get

$$\rho^{t+1} = (l, \ldots, \underset{i-1}{l}, \underset{i}{r}, \underset{i+1}{r}, \underset{i+2}{r}, \ldots, r),$$

Hence, we get a sharp backwards travelling front without any dispersion.

Case 3: $l \in [0, \rho^*] \ \wedge \ r \in [\rho^*, \rho^{max}]$: This case is slightly more involved. We show the claim in two steps:

i) Computing the next time step via Lax-Friedrich leads to

$$\rho^{t+1} = (l, \ldots, \underset{i-1}{l}, \underset{i}{m}, \underset{i+1}{m}, \underset{i+2}{r}, \ldots, r),$$

with $m = l + r - \rho^* \in [l, r]$.

ii) Given the densities

$$\rho^{\bar{t}} = (l, \ldots, \underset{i-1}{l}, \underset{i}{m}, \underset{i+1}{m}, \underset{i+2}{r}, \ldots, r),$$

with an arbitrary $m \in [l, r]$.

a) If $m \in [0, \rho^*]$, the density for the next time step evolves to

$$\rho^{\bar{t}+1} = (l, \ldots, \underset{i}{l}, \underset{i+1}{\hat{m}}, \underset{i+2}{\hat{m}}, \underset{i+3}{r}, \ldots, r),$$

with $\hat{m} = m + r - \rho^*$. Due to the assumption made for Case 3, we have

$$\hat{m} = m + r - \rho^* \geq m \geq l$$

Furthermore, we have

$$\hat{m} = \underbrace{m}_{\text{a) } \leq \rho^*} - \rho^* + r \leq r.$$

Consequently, we get $\hat{m} \in [l, r]$.

b) If $m \in [\rho^*, \rho^{max}]$, the density values for the following time step are

$$\rho^{\bar{t}+1} = (l, \ldots, \underset{i-2}{l}, \underset{i-1}{\mathring{m}}, \underset{i}{\mathring{m}}, \underset{i+1}{r}, \ldots, r),$$

with $\mathring{m} = l + m - \rho^*$. We have

$$\mathring{m} = l + \underbrace{m}_{b) \geq \rho^*} - \rho^* \geq l$$

and

$$\mathring{m} = \underbrace{l}_{\text{(Case 3)} \leq \rho^*} + m - \rho^* \leq m \leq r$$

$$\Rightarrow \ \mathring{m} \in [l, r].$$

Hence, $\rho^{\bar{t}+1}$ again fulfills the assumptions imposed to $\rho^{\bar{t}}$, with the shape shifted by one space step either to the left or to the right. Therefore, by applying the Lax-Friedrichs Scheme iteratively over time, the dispersion will never become greater than two space steps.

Case 4: $l \in [\rho^*, \rho^{max}] \ \wedge \ r \in [0, \rho^*]$:

i) Computing time-step $t+1$ via Lax-Friedrich yields:

$$\rho^{t+1} = (l, \ldots, \underset{i-1}{l}, \underset{i}{\rho^*}, \underset{i+1}{\rho^*}, \underset{i+2}{r}, \ldots, r).$$

ii) Applying again (1.63) the densities for time-step $t+2$ are given by:

$$\rho^{t+2} = (l, \ldots, \underset{i-1}{\rho^*}, \underset{i}{\rho^*}, \underset{i+1}{r}, \underset{i+2}{r}, \ldots, r).$$

Hence, the resulting wave front is moving backwards carrying along two middle density values ρ^*.

□

Algorithm for Hamilon-Jacobi Scheme on Networks

The complete numerical scheme for solving Hamilton-Jacobi equations on road networks is described in Algorithm 1. Some steps are illustrated in Figure 1.22.

A crucial point is the computation of the coupling condition, depicted in Figure 1.22(c). As denoted in line 16 of Algorithm 1, equations (1.34) to (1.49) are used. The detailed procedure is the following: Consider a junction v with at most two incoming roads ($\in \delta_v^{in}$) and at most two outgoing roads ($\in \delta_v^{out}$). The leftmost grid-points of the incoming roads and the rightmost grid-points of the outgoing roads in terms of the density ρ have already been computed for time-step $t+1$, see Figure 1.22(b). Hence, the values for $\rho_{e,n_x}^{t+1} \, \forall e \in \delta_v^{in}$ and $\rho_{e,0}^{t+1} \, \forall e \in \delta_v^{out}$ are given corresponding to $\rho_e(L)$ and $\rho_e(0)$ in the continuous notation. Now, we use equations (1.34) and (1.35) to obtain the maximal possible flow γ_e^{max} for all roads e at the junction. Depending on the junction type we compute the coupling flows $\hat{f}_e \, \forall e \in \delta_v^{in}$ and $\bar{f}_e \, \forall e \in \delta_v^{out}$ using equations (1.43), (1.45), (1.47) or (1.49). The density boundary values $\hat{\rho}_e \, \forall e \in \delta_v^{in}$ and $\bar{\rho}_e \, \forall e \in \delta_v^{out}$ are uniquely given by (1.40) and (1.41). An illustration of this procedure is given in Figure 1.21.

Algorithm 1: Hamilton-Jacobi Scheme for networks.

/* Input: Road network with length and flow function for each road, initial and boundary conditions in terms of ρ, time horizon T, grid size Δx, number of ghost-cells n_g */

/* Output: Simulation of the traffic in terms of density */

begin

/* Compute number of grid-points */

number of space-steps: $n_{xe} = \lceil \frac{L_e}{\Delta x} \rceil + 1 - n_g, \forall e \in E$;

time grid size: $\Delta t = \frac{\Delta x}{\max_{e \in E}\{\max_\rho |f_e'(\rho)|\}}$;

number of time steps: $n_t = \lceil \frac{T}{\Delta t} \rceil + 1$;

/* Transfer initial values from ρ to M. */

forall the $e \in E$ **do**

$\hat{M}_e^0 = 0$; /* right boundary value */

$M_{e,r_{n_g}}^0 = \hat{M}_e^0 - \Delta x \hat{\rho}_e^0$; /* rightmost ghost-cell */

$M_{e,j}^0 = M_{e,j+1} - \Delta x \rho_{e,i}^0 \forall$ grid-points j (including ghost-cells);

$\bar{M}_e^0 = M_{e,0}^0 - \Delta x \bar{\rho}_e^0$; /* left boundary value */

for $t = 0, \ldots, n_t - 1$ **do**

/* Compute next time iteration for each road e */

forall the $e \in E$ **do**

Compute $M_{e,j}^{t+1}$ by (1.71) $\forall$ grid-points j (including ghost-cells) /* see Figure 1.22(a) */

/* Transfer M to ρ */

forall the $e \in E$ **do**

$\rho_{e,i}^{t+1} = \frac{M_{e,j+1}^{t+1} - M_{e,j}^{t+1}}{\Delta x}$, $\forall$ grid-points j /* see Figure 1.22(b) */

/* Compute coupling at junctions */

forall the $v \in V$ **do**

Compute coupling for time-step t according to junction type using density values next to ghost-cells. /* see Figure 1.22(c) and 1.21 */

/* Get boundary value in terms of M */

forall the $e \in E$ **do**

left: $\bar{M}_e^{t+1} = \tilde{M} l_{n_g} - \bar{\rho}_e^{t+1} \Delta x$;

right: $\hat{M}_e^{t+1} = \tilde{M} r_{n_g} + \hat{\rho}_e^{t+1} \Delta x$ /* see Figure 1.22(d) */

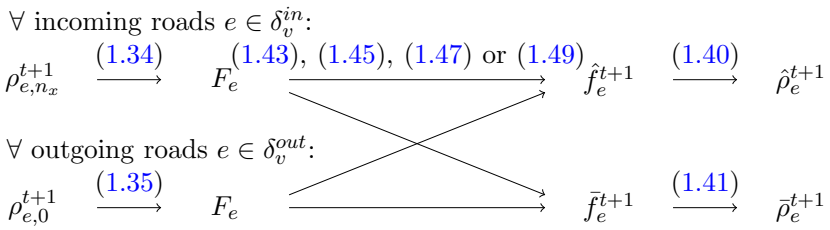


Figure 1.21: Computation of the coupling.

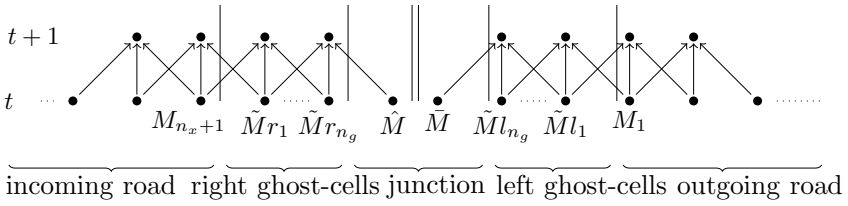
Remark 1.4.3. *We give some further explanations on Algorithm 1:*

line 2: Note that the Godunov Scheme [45] does not need any ghost-cells to compute the coupling condition. The presented scheme introduces numerical diffusion such that the ghost cells need to be sufficiently large. Its size has been discussed in the previous lemma. The length of the ghost cells is chosen equal to the size of the interior cells. In order to have the same speed of propagation those cells do not enter the computation of the length of the road.

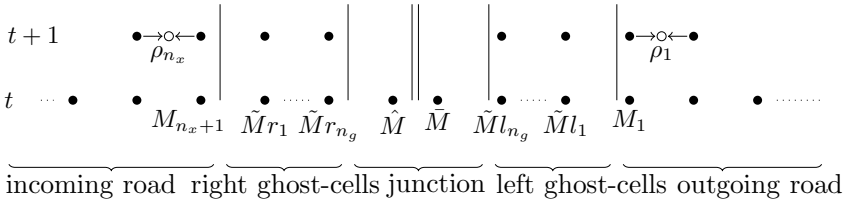
line 3: Choose the size of the time grid such that the CFL-condition holds.

line 8: M is initialised from right to left on each road.

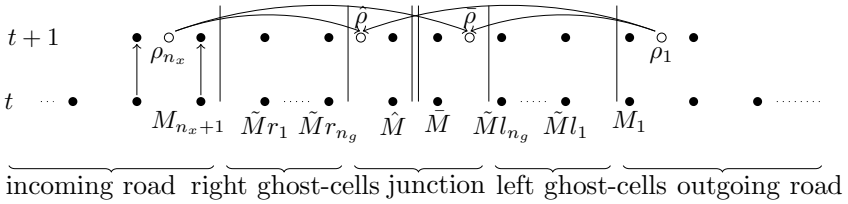
Algorithm 1 is not only useful to simulate the traffic density evolution on road networks with prescribed initial and boundary data, it also permits to compute the trajectories of single roads, by plotting the contour lines of M. Numerical results are shown in Chapter 4, see Figure 3.30.



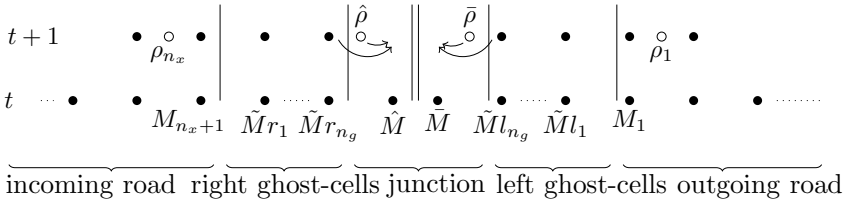
(a) Computation of the next time step for the inner cells in terms of M, see Algorithm 1, line 12.



(b) Computation of density value at last grid point before the ghost-cells, see Algorithm 1, line 14.



(c) Computation of the coupling density values $\hat{\rho}_i$ and $\bar{\rho}_j$, see Algorithm 1, line 16.



(d) Computation of the coupling values in terms of M, namely $\hat{M}_i$ and $\bar{M}_j$, see Algorithm 1, line 19.

Figure 1.22: Schematic procedure of the algorithm, exemplarily for one incoming and one outgoing road. For the sake of readability, we skip the road index in this illustration.

2

Optimisation containing discrete Decisions

In the previous chapter we derived dynamic transportation networks (DTNs) for several applications. They enable us to simulate various scenarios with respect to dynamic flows on networks such as production flows or traffic density evolution with prescribed parameters, as well as initial and boundary conditions. The next step is to use these models in order to answer questions concerning the optimal parameter setting in terms of the best possible performance of the given scenario. Comparable to the classical static Maximum Flow Problem (1.1), there also exists a broad range of optimisation tasks for DTNs depending on the considered model and application. In the context of production networks various questions of interest have been considered. For example, [42, 46] focus on finding the best distribution inside a production network in order to achieve minimal queuing sizes and maximal production flow. This leads to storage cost reduction and increasing output of products. In addition to that, [37, 98] present an extended model that allows for choosing properties of machines such that maximal product output is obtained. [48, 57] describe a model with different types of goods having different priorities and derive optimal control policies for each processor. In the context of evacuation models [22, 52] are dedicated to find optimal routing of cars on road networks and people in buildings in emergency situations. Furthermore, there exists a broad variety of literature devoted to optimal signal timing of traffic lights on road networks in order to minimise travel times and maximise traffic flow, see [8, 13, 15, 44, 68, 76, 86] for an overview. These and many other questions outline the necessity to derive efficient

and reliable methods capable to solve DTN-based optimisation problems.

Following the notation of Definition 1.1.2, these optimisation problems typically have the following structure:

$$\begin{aligned}
&\text{objective function:} && \max \mathcal{F}(\mathcal{D}_i) && \text{(2.1a)}\\
& && \text{such that} &&\\
&\text{control constraints:} && \mathcal{K} \cdot k \leq \mathcal{U}_k && \text{(2.1b)}\\
&\text{edge constraints:} && \mathcal{I}_i \equiv 0,\ \forall i \in E && \text{(2.1c)}\\
&\text{coupling constraints:} && \mathcal{C}_{i/v} \equiv 0,\ \forall i \in E / v \in V && \text{(2.1d)}\\
&\text{initial conditions:} && \mathcal{D}_i(t=0) \equiv \mathcal{D}_{0i},\ \forall i \in E && \text{(2.1e)}\\
&\text{inflow boundary conditions:} && \mathcal{D}_i(x=0) \equiv \mathcal{B}_i^{in},\ \forall i \in E^{in} && \text{(2.1f)}\\
&\text{outflow boundary conditions:} && \mathcal{D}(x=L_i) \equiv \mathcal{B}_i^{out},\ \forall i \in E^{out} && \text{(2.1g)}\\
&\text{box constraints:} && \mathcal{D}_i(t) \in [\mathcal{L}_i, \mathcal{U}_i],\ \forall i \in E && \text{(2.1h)}
\end{aligned}$$

where $\mathcal{F}$ represent the objective function depending on the optimisation question of interest, k are the actual control parameters depending on the application and are either members of the dynamic functions $\mathcal{D}_i$ or the edge properties $\mathcal{P}_i$. $\mathcal{K}$ is a matrix, representing the linear constraints the control parameters have to fulfill. Typically, the edge constraints (2.1c) consist of coupled ordinary or partial differential equations.

There are mainly two different approaches to solve optimisation problems based on DTNs. Since (2.1) is typically ODE/PDE-constrained, a common solution method is the use of adjoint equations deduced from the Lagrange principle, see [93]. They involve the use of iterative gradient based optimisation methods, cf. [63]. However, the feasible domain of the variables is often highly complex due to the network structure of DTNs. For that reason it is rather difficult to obtain reliable solutions using common iterative descent methods, since the optimisation procedure will easily get stuck in local extrema.

Hence, a different optimisation approach is considered in this chapter: The application of Branch & Bound techniques for linear mixed integer optimisation problems (short linear MIPs), see [30, 70, 82, 90, 94]. This approach has the big advantage that the iterative computation of primal and dual bounds during the optimisation process ensures the global optimality of the returned solution. If the process is interrupted

before an optimal solution has been found, the interval where the optimal objective function value is situated – the so-called optimality gap – is returned. Furthermore, there exist many commercial optimisation software packages, which can be used as blackbox solvers. Another advantage of this technique is that it is extremely easy to consider only integer values of certain variables (such as number of workers on each machine) by using integer constraints and integrate discrete decisions.

These methods are applicable to many DTNs for the following reason: Often, DTNs are transformable into linear MIPs, which we will refer to as DTN-MIPs, see Definition 2.2.1. This can be obtained employing common numerical discretisations, cf. Section 1.4, combined with rewriting techniques borrowed from discrete optimisation [62]. It is possible to convert particular nonlinear structures (e.g. the *min*-function) into a dynamic mixed-integer framework. These MIPs can be optimised using branching techniques as well as primal and dual bounds, providing reliable information about the interval, in which the optimal solution can be found. In the context of production network models, this approach has been introduced in [42] and has been successfully applied to a wide variety of production problems, see [37, 48, 49, 57] for an overview. Furthermore, similar ideas have been developed and applied for a specific type of traffic models, the cell transmission models, see [8, 53, 75, 76, 77].

Since the resulting MIPs are highly complex, it is important to develop methods leading to runtime improvements of the optimisation procedure. At that point, we can exploit the fact that we posses a lot of information due to the problem structure which can easily be provided to the optimisation algorithm. One efficient approach – presented in [36] – are adapted presolve techniques to strengthen bounds of constraints, such that the actual optimisation can be completed much faster. Additionally, it is possible to tune the optimisation process itself by applying suitable heuristics in order to find good primal bounds throughout the optimisation procedure. These ideas are considered in the course of this chapter.

Section 2.1 contains some basic definitions as well as a short review on classical optimisation techniques for linear MIPs, such as the before mentioned Branch & Bound Algorithm. The following section, Section 2.2, is the heart of this work. It derives a general strategy, how DTNs can be transformed into linear MIPs and how the knowledge of the dynamics can be exploited to speed up the optimisation procedure. In this context we point out common properties of DTNs and propose linearisation techniques,

that are required to transform the setting into a linear MIP. Furthermore, we show how common drawbacks, such as high oscillating control variables in optimal solutions, can be avoided. The aim of this section is to formulate these strategies in a general way. In that way it provides a framework that helps to solve optimisation issues for a broad range of DTNs.

The following two sections provide examples how these strategies can be applied to the particular DTNs derived in Section 1.2 and 1.3. Section 2.3 is dedicated to find optimal worker scheduling for production networks in order to maximise the production flow. Basing on the traffic flow network model derived in Section 1.3, Section 2.4 presents the modelling of traffic lights on complex urban junctions, transforms the setting into a linear MIP and provides promising tuning techniques for the optimisation procedure.

2.1 Linear Mixed Integer Optimisation Methods

In this section a review of classical techniques for solving linear mixed integer optimisation problems is provided. Widely used techniques are LP-based Branch & Bound Algorithms optionally combined with cutting plane methods.

We will summarise some of the main ideas of these approaches. For more detailed information we refer to [29], [69] and [90], amongst others.

2.1.1 Basic Definitions

We start with some basic definitions on linear and mixed integer programming.

Definition 2.1.1. *A **linear program** (short: **LP**) has the following form: Find a vector $x \in \mathbb{R}^n$ that solves*

$$\max c^T x \tag{2.2a}$$

such that

$$Ax \leq b \tag{2.2b}$$

$$x \geq 0 \tag{2.2c}$$

$$x \in \mathbb{R}^n, \tag{2.2d}$$

with given vectors $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ and a given matrix $A \in \mathbb{R}^{m \times n}$.

A linear mixed integer optimisation problem has a similar form. The only difference is that some of the variables are integers or binaries.

Definition 2.1.2. *A **linear mixed integer optimisation problem** (short: **linear MIP** or **LMIP**) has the following form: Find a vector $x \in \mathbb{R}^n$ that solves*

$$\max c^T x \tag{2.3a}$$

such that

$$Ax \leq b \tag{2.3b}$$

$$x \geq 0 \tag{2.3c}$$

$$x \in \{0,1\}^p \times \mathbb{Z}^l \times \mathbb{R}^{n-p-l}, \tag{2.3d}$$

with given vectors $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ and a given matrix $A \in \mathbb{R}^{m \times n}$. Furthermore, we have a prescribed number of binary variables p and integer variables l with $p + l \leq n \in \mathbb{N}$.

LP (2.2) is also called **relaxation** of (2.3), since the binary and integrality constraints are neglected.

Definition 2.1.3. *The **dual problem** of (2.2) is given by*

$$\max b^T y \tag{2.4a}$$

such that

$$A^t y \geq c \tag{2.4b}$$

$$y \geq 0 \tag{2.4c}$$

$$y \in \mathbb{R}^m, \tag{2.4d}$$

with given vectors $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ and a given matrix $A \in \mathbb{R}^{m \times n}$.

A crucial theorem of optimality theory is the **Duality Theorem**. It states that, if the LP (2.2) has an optimal solution, its corresponding dual problem (2.4) has an optimal solution and the optimal objective function values coincide.

2.1.2 Branch & Bound Algorithm

One basic algorithm that is used in mixed integer optimisation theory, is the Branch & Bound Algorithm. Later on, we show how it is possible to use the knowledge of the structure of DTN-MIPs, see Definitions 2.2.1, to speed up the optimisation procedure based on the Branch & Bound Algorithm. More details are given in Subsection 2.2.4.

First of all, we shortly summarise the main ideas which can be adapted and extended according to the specific problem under consideration, followed by a small example. Then we present the structure of the algorithm, cf. Algorithm 2, which consists of an iterative application of the following operations.

Branching. In the course of the optimisation process the original problem is split into several disjoint subproblems. This technique is called branching and is used iteratively, leading to a tree whose nodes present the disjoint subproblems. One example for branching is the following: Choose a binary variable $x_i \in \{0, 1\}$ and add the additional constraint $x_i = 0$ to the first new subproblem and $x_i = 1$ to the second new subproblem, see Figure 2.1.

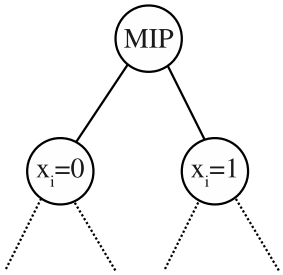


Figure 2.1: Branching on binary variables.

Pruning. There are various techniques to find upper and lower bounds of the optimal objective function value of a subproblem. In this way, we can tell beforehand which subproblems might contain the optimal solution of the original MIP and which ones can be neglected. Hence, nodes of the tree can be cut off. This technique is called pruning and can be divided into three different types:

- **Pruning by optimality:** When the optimal solution of a subproblem has been found, no further branching on that node is necessary.
- **Pruning by bound:** When the lower bound of the optimal objective function value of a subproblem is greater than a global upper bound being defined as the minimum of all upper bounds that have been found so far, the optimal solution is not included in this subproblem. Hence it can be pruned.

- **Pruning by infeasibility:** If a subproblem does not contain any feasible solution, it can also be neglected. One common method to find infeasible subproblems is to compare the dual bound of the subproblem with the currently best found feasible solution of the whole tree. If the dual bound is already worse than a feasible solution of another subproblem, the considered subtree can be pruned.

Bounding. As mentioned earlier, it is of interest to find good bounds for the subproblems. In fact, the Branch & Bound Algorithm terminates the faster, the sharper the bounds are. The reason for this is that more nodes can be pruned and hence the size of the tree is kept small. Various methods to determine bounds have been developed within the scope of integer optimisation research. A common procedure is to find dual bounds by relaxing the problem to a simple linear programming problem that can be solved by the Simplex Algorithm, which is described in [51, 81, 84] and others. The relaxation is done by neglecting the integrality constraints of x. Another method to find dual bounds is finding a feasible solution of the dual problem. Primal bounds are provided by any feasible solutions of the subproblems using appropriate heuristic algorithms. For more detailed information, read for example [82].

Figure 2.2 illustrates the Branch & Bound procedure.

Example 2.1.4. *The procedure is illustrated by a small example which is taken from [38].*

$$\begin{aligned} &\min -5x_1 - 6x_2 - 9x_3 \\ &\textit{such that } 5x_1 + 9x_2 + 4x_3 \leq 15 \\ &x = (x_1\, x_2\, x_3)^T \in \{0,1\}^3 \end{aligned}$$

The corresponding Branch & Bound tree is shown in Figure 2.3.

The structure of a typical Branch & Bound Algorithm is depicted in Algorithm 2. For details of each step, we refer to [70].

2.1.3 Cutting Planes

Another popular method to find optimal solutions of linear MIPs is the cutting plane algorithm. The first step is to solve the LP-relaxed problem with the Simplex Algorithm, see e.g. [81]. In the case that the optimal solution does not fulfill all required

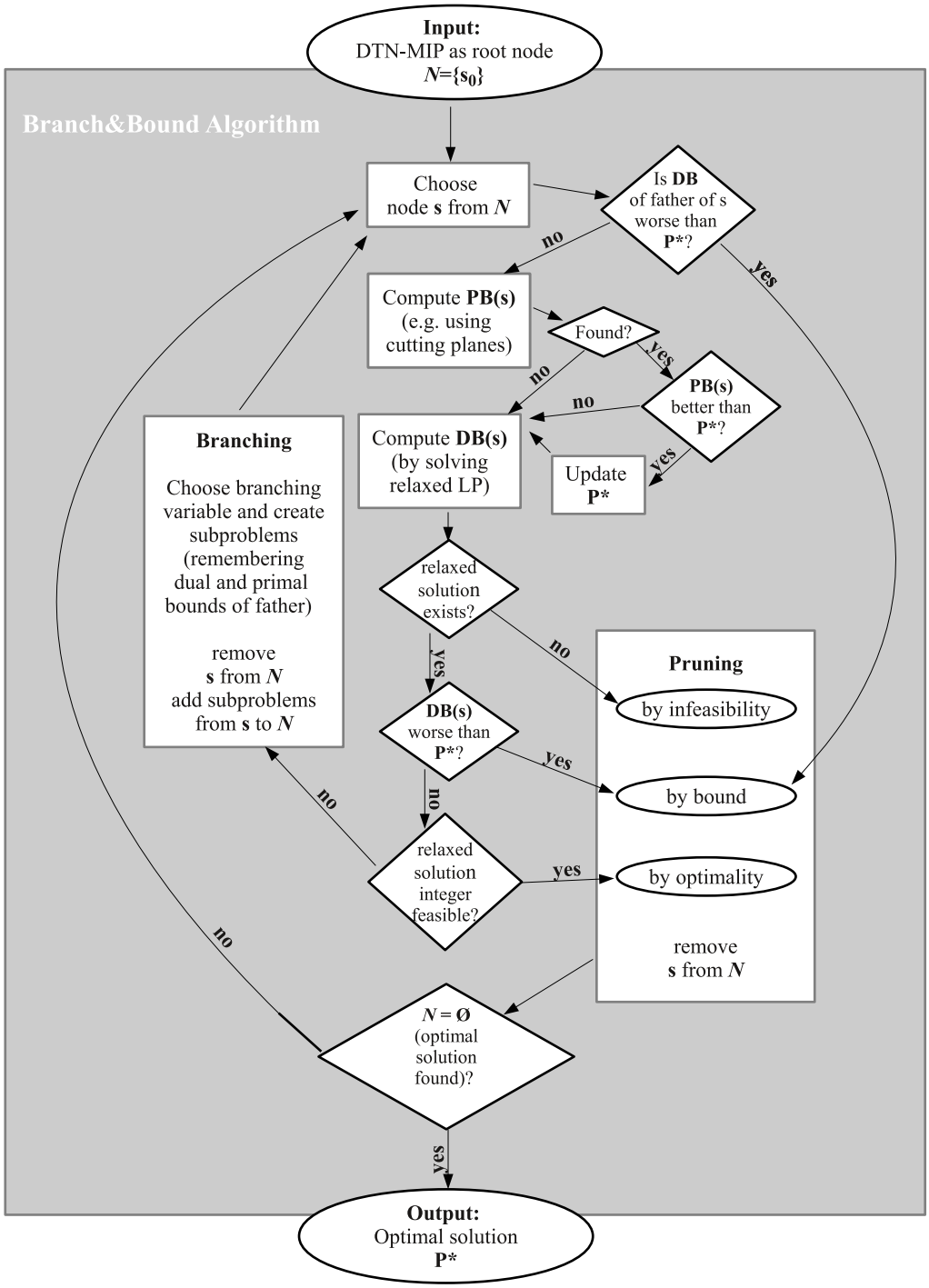


Figure 2.2: Procedure of the Branch & Bound Algorithm, where $PB(s)$ and $DB(s)$ denote the primal and dual bound of node s and the best known feasible solution is referred to as P^*.

Algorithm 2: Branch & Bound Algorithm.

```
   /* Input:  A linear mixed integer optimisation problem with
      variables x_i, i ∈ I                                          */
   /* Output:  An optimal solution                                  */
 1 begin
      /* N is the set of active nodes of the Branch & Bound tree,
         with root problem s_0 containing all constraints of the
         original linear MIP.                                       */
      /* X̃ is the currently best found feasible solution.          */
 2    N := {s_0}
 3    while no optimal solution is found and N ≠ ∅ do
 4       Choose node s ∈ N.
 5       Compute dual bound of s.
 6       Compute primal bound of s.
 7       Apply pruning techniques and remove unnecessary nodes from N.
 8       Choose index i for branching.
 9       Create J subnodes
         /* e.g.  s_j = {s ∪ x_i ≤ d_j}, j = 1,...,J if x_i is continuous or */
         /*  0    s_j = {s ∪ x_i = d_j}, d_j ∈ Z, j = 1,...,J if x_i is integer.
            */
10       Add s_j, j ∈ J to N.
11       Set X̃ to the currently best found feasible solution.
12    return X̃.
```

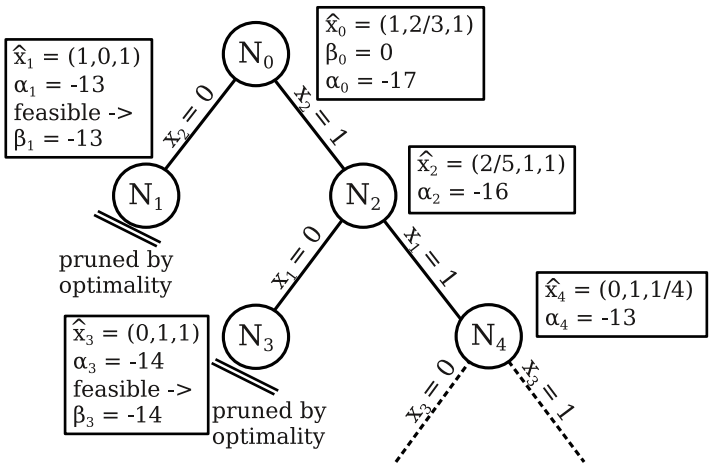


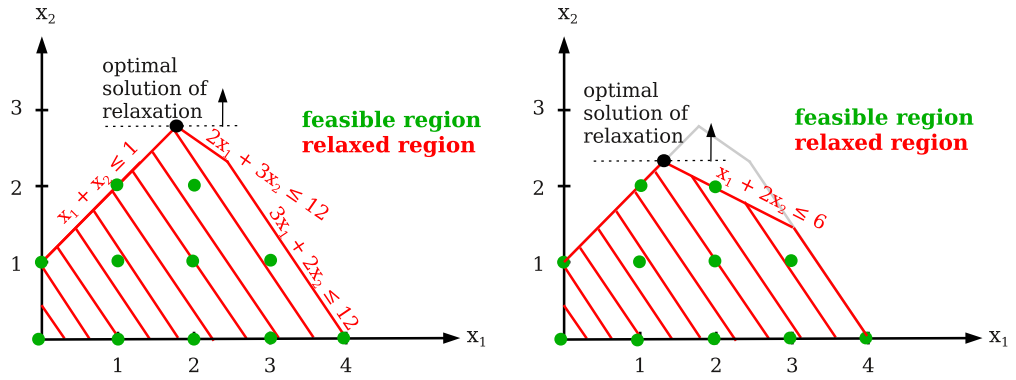
Figure 2.3: Branch & Bound tree. N_i denote the subproblems in the order they are generated. β_i denotes the upper bound of the node and α_i the lower bound. $\hat{x}_i$ is the optimal solution of the LP-relaxed problem.

integer constraints, we iteratively add additional constraints that reduce the relaxed feasible region in a way such that the non-integer feasible variable of the relaxed solution is cut off. In this way we strengthen the feasible region step by step, getting closer to the convex hull of the original integer problem. We continue until a integer feasible solution of the MIP is found. This method is depicted in Figure 2.4 for a small example case.

$$\begin{aligned} & \min -2x_2 \\ \text{such that } & -x_1 - x_2 \leq 1 \\ & 3x_1 + 2x_2 \leq 12 \\ & 2x_1 + 3x_2 \leq 12 \\ & x = (x_1, x_2)^T \in \mathbb{Z}^2 \end{aligned}$$

2.1.4 Branch & Cut

The Branch & Cut Algorithm combines techniques from the Branch & Bound and cutting plane algorithm. At every node of the Branch & Bound tree, the primal bound



(a) Depiction of the feasible regions. The black dot denotes the optimal solution of the relaxed problem.

(b) The cutting plane $x_1 - 2x_2 \leq 6$ is added.

Figure 2.4: Cutting plane algorithm.

is computed using the LP-relaxation. Then, cutting planes are added to achieve a sharper bound or even an integer feasible solution. Hence, this approach leads to smaller Branch & Bound trees. This is one of the most popular procedures to compute optimal solutions for linear MIPs and is applied in many software packages, see the following subsection.

2.1.5 Optimisation Software

Currently, there are several optimisation software packages on the market, which are able to solve linear MIPs automatically, for instance [23, 60, 85, 91]. Some of them provide interfaces for the user to adapt the solution procedure to the specific problem type in order to obtain increased efficiency. This can either mean to set certain parameters for the optimisation algorithm, such as error tolerances or priorities for the order of branching or considered subnodes, or it can even allow the user to integrate own tuning elements, such as heuristics for finding primal bounds or subroutines for creation of new subproblems and branching rules. We use Cplex [23] to solve DTN-MIPs. For more details we refer to Chapter 3, where numerical results are presented.

2.2 Mixed-Integer-Techniques meet DTN-Models

In this Section we derive a general strategy how MIP optimisation techniques can be applied to dynamic transportation network models. We need to transform the model into a linear MIP. A linear MIP can be solved by a blackbox solver, as mentioned in Subsection 2.1.5. Optionally, it is also possible to reduce optimisation time by providing information on the network model to the optimisation process.

Definition 2.2.1. *A **DTN-MIP** is a linear MIP (2.3) based on a DTN model. The variables x comprise all control, linearisation and state variables for all time and space discretisation points. The coefficience matrix A and coefficient vector c depend on network parameters $\mathcal{N}$ and edge properties $\mathcal{P}$.*

For the sake of clarity, we distinguish between three types of variables:

Control variables. Control variables depend on the specific optimisation issue; they represent the quantities that are used as controls to find the optimal solution and usually originate from dynamic functions $\mathcal{D}$ or edge parameters $\mathcal{P}$ of the underlying DTN. In the applications described below they represent the number of workers that are at each point in time at each machine (cf. Section 2.3) and the state of the traffic light at each road (cf. Section 2.4), respectively.

State variables. State variables are discretised quantities originating from all dynamic functions $\mathcal{D}$ that do not play the role of control variables, such as density ρ and flow evolution f or buffer levels u.

Linearisation variables. Linearisation variables are additional continuous or binary variables that are needed in order to linearise the model constraints. For more details see Subsection 2.2.2.

2.2.1 Transformation and Solution Strategy

We suggest the following strategy to apply mixed integer optimisation techniques on optimisation problems originating from DTN models:

I. Introducing control variables and their constraints. We introduce control variables k with respect to the considered optimisation issue and derive conditions $\mathcal{K}$ that have to be satisfied by the control variables. Constraints which only depend on control variables will be referred to as **control constraints**, see (2.1b). According to the notation of (2.3), the variables k are elements of x and $\mathcal{K}$ is incorporated into A.

Remark 2.2.2. *The optimisation methods used later apply the Simplex Algorithm for the relaxed formulations in order to find dual bounds for the problem. When the optimal solution of the relaxed formulation happen to fulfill the integer (and binary) constraints as well, we immediately get an optimal solution for a subproblem of the MIP. This leads to the reduction of branches in the Branch & Bound Algorithm. In that way the optimisation time is reduced. This effect is enhanced, when we formulate the control constraints in a way that the feasible region of the relaxed problem is close to the convex hull of the original problem. Then, integer feasible solutions are found at the corners of the region. For more details on polyhedra theory and integer optimisation, see [89, 90].*

II. Deriving a suitable objective function. Next, we introduce an objective function to obtain a continuous, PDE- or ODE-constraint optimisation problem, see (2.1). Depending on the considered DTN model, the constraints may contain the evolution of density on the arcs of the network or the evolution of the buffer sizes, coupling conditions at the vertices and the control constraints of step I.

III. Discretising constraints of the DTN model. The next step is to introduce a discrete time and spatial grid and apply discretisation schemes, cf. Section 1.4. We preferably choose methods that are easy to linearise. In this way we obtain the discrete state variables possibly for each time and space-step.

IV. Linearizing constraints. All constraints are linearised with respect to the control as well as to the state variables. This requires the application of linearisation techniques described in Subsection 2.2.2. For this procedure additional linearisation variables are introduced. In that way, we obtain a linear DTN-MIP.

V. Preventing high oscillations of control variables in the optimal solution. It is possible that the control variables of the optimal solution are strongly fluctuating in time. In most of the applications, this is not of practical use. In our examples this

would mean that a traffic light switches every second or workers have to chance their position too frequently. In order to avoid these fluctuation effects, further techniques can be applied as described in Subsection 2.2.3.

VI. Tuning the optimisation procedure. The obtained MIP is highly complex, since the number of constraints and variables is not only proportional to the number of arcs of the network, but also to the number of time- and space-steps. For that reason the optimisation time is often unacceptably large. However, we possess a lot of useful information due to the structure of the model. Subsection 2.2.4 is dedicated to strategies, how this information can be used to significantly speed up the optimisation procedure.

2.2.2 Linearisation Techniques

In this subsection we consider step IV of the above mentioned strategy. We show how to linearise several expressions that might be encountered in optimisation problems originating from DTN models, such as those derived in Section 1.2 (cf. Remark 1.2.3) and Section 1.3 (cf. Remark 1.3.8). We describe techniques how to reformulate these terms using linear constraints without loosing information or accuracy. For more details we refer to [62].

Product of binary and continuous variable. As explained in Section 5.6.5 of [62], it is possible to linearise the product of a positive continuous variable x and a binary variable $\beta \in \mathbb{B}$.

Assume that

$$\Big\{0 \leq x \leq M; \quad x \in \mathbb{R}; \quad \beta \in \mathbb{B}\Big\} \tag{2.5}$$

and consider equation

$$y = \beta \cdot x. \tag{2.6}$$

Now, (2.6) can be described by

$$\beta = 0 \quad \Longrightarrow \quad y = 0 \tag{2.7a}$$

$$\beta = 1 \quad \Longleftrightarrow \quad y = x. \tag{2.7b}$$

Hence, (2.7) is equivalent to

$$y \leq M \cdot \beta \quad \wedge \quad y \leq x \quad \wedge \quad y \geq x - M(1-\beta). \tag{2.8}$$

Minimum-expressions. DTNs often contain min-terms such as (1.17b), (1.22) and (1.23) for the production flow model, cf. Section 1.2, and equations (1.43), (1.45), (1.47) and (1.49) for the traffic model, see Section 1.3. They can be linearised applying the following Lemmata.

Lemma 2.2.3. *An expression of the form* $c = \min\{a, b\}$ *is linearised by introducing a binary variable* $\gamma \in \{0, 1\}$ *and using the additional inequality constraints*

$$\begin{aligned} \gamma \cdot a \leq c \leq a \\ b - M \cdot \gamma \leq c \leq b \end{aligned}$$

where M *is sufficiently large, such that* $M > b$ *holds.*

One can easily check that $\gamma = 1$ is equivalent to the case $c = a$, and $\gamma = 0$ is valid, if and only if $c = b$, cf. references [42, 62].

This approach can iteratively be used for minimum expressions consisting of an arbitrary number of terms. For example, we get the following transformation for a minimum expression containing three terms:

Lemma 2.2.4. $a = \min(b, c, d)$ *with* $b, c, d \geq 0$ *is equivalent to the set of constraints*

$$\begin{aligned} \beta b \leq e \leq b \\ c - c^{max}\beta \leq e \leq c \\ \eta e \leq a \leq e \\ d - d^{max}\eta \leq a \leq d \\ \beta, \eta \in \mathbb{B} \end{aligned} \tag{2.9}$$

with c^{max} *and* d^{max} *upper bounds for* c *and* d *and* $e \in \mathbb{R}$.

Proof. Setting $e := \min(b, c)$ it becomes clear that $a = \min(e, d)$. Then applying Lemma 2.2.3 completes the proof. □

If-else-construction. The triangular flow function 1.26 used in the traffic flow context, is defined piecewies. In this paragraph we show, how these constructions are linearised. Let A and B be two sets, let $s \in [0, s^{max}]$ be a variable. The following expression needs to be linearised:

$$x \in A, \quad \text{if } 0 \leq s \leq s^* \tag{2.10a}$$
$$x \in B, \quad \text{if } s^* \leq s \leq s^{max}. \tag{2.10b}$$

Remark 2.2.5. *We assume that for the case $s = s^*$ both $x \in A$ and $x \in B$ are allowed.*

(2.10) is equivalent to

$$x = y, \quad \text{if } 0 \leq s < s^* \tag{2.11a}$$
$$x = z, \quad \text{if } s^* < s \leq s^{max} \tag{2.11b}$$
$$x \in \{y, z\}, \quad \text{if } s = s^* \tag{2.11c}$$
$$y \in A \tag{2.11d}$$
$$z \in B. \tag{2.11e}$$

Now, we introduce a binary variable $\zeta \in \mathbb{B}$. We want to find linear constraints which are equivalent to the following relations

$$\zeta = 0 \quad \Leftrightarrow \quad x = y \quad \Leftarrow \quad 0 \leq s < s^* \tag{2.12a}$$
$$\zeta = 1 \quad \Leftrightarrow \quad x = z \quad \Leftarrow \quad s^* < s \leq s^{max}. \tag{2.12b}$$

We use:

$$x = \zeta z + (1 - \zeta) y \tag{2.13a}$$
$$0 \leq (\frac{1}{2} - \zeta)(s^* - s) \tag{2.13b}$$

Now, we have to linearise the following terms

$$\tilde{y} := \zeta \cdot y \text{ and } \tilde{z} := \zeta \cdot z.$$

Applying steps (2.5) - (2.8), we end up with the following constraints for $\tilde{y}$:

$$0 \leq \tilde{y} \leq y^{max} \cdot \zeta$$
$$y - y^{max}(1-\zeta) \leq \tilde{y} \leq y$$

The linearisation for $\tilde{z}$ is done analogously.

Altogether, this yields the following set of constraints, which replace (2.10):

$$x = \tilde{z} + y - \tilde{y} \tag{2.14a}$$
$$0 \leq (\frac{1}{2} - \zeta)(s^* - s) \tag{2.14b}$$
$$0 \leq \tilde{y} \leq y^{max}\zeta \tag{2.14c}$$
$$y - y^{max}(1-\zeta) \leq \tilde{y} \leq y \tag{2.14d}$$
$$0 \leq \tilde{z} \leq z^{max}\zeta \tag{2.14e}$$
$$z - z^{max}(1-\zeta) \leq \tilde{z} \leq z \tag{2.14f}$$
$$\zeta \in \mathbb{B} \tag{2.14g}$$
$$y \in A \tag{2.14h}$$
$$z \in B \tag{2.14i}$$

with linearisation variables y, z, $\tilde{y}$, $\tilde{z} \in \mathbb{R}^+$ and $\zeta \in \mathbb{B}$.

Piecewise linear functions. Piecewise linear functions such as (1.26) often appear in the context of DTNs. As described in detail in Section 5.6.3 of [62], with the help of additional binary variables it is possible to find linear constraints which express piecewise linear functions. Here, we want to apply these techniques as well as the techniques of the previous paragraph to the special case of triangular flow functions, which are commonly used in traffic flow models, see Subsection 1.3.

Note, that for the case $\rho = \rho^*$ holds $\lambda \cdot \rho = \lambda(2\rho^* - \rho)$.

Analogously to the previous paragraph, we reformulate (1.26) by

$$f = \kappa \cdot \lambda\rho + (1-\kappa)(\lambda(2\rho^* - \rho)) \tag{2.15a}$$
$$0 \leq \rho \leq \rho^{max} \tag{2.15b}$$
$$\kappa \in \mathbb{B}. \tag{2.15c}$$

Following steps (2.11) to (2.14), we add a new variable $\tilde{\rho}$ representing the product of κ with ρ and obtain the following linear constraints:

$$f = 2\lambda\tilde{\rho} - \rho^* 2\lambda\kappa - \lambda\rho + \rho^* 2\lambda \tag{2.16a}$$

$$0 \leq \rho^*\kappa - \tilde{\rho} + \frac{1}{2}\rho - \frac{1}{2}\rho^* \tag{2.16b}$$

$$0 \leq \tilde{\rho} \leq \rho^{max} \cdot \kappa \tag{2.16c}$$

$$\rho - \rho^{max}(1-\kappa) \leq \tilde{\rho} \leq \rho \tag{2.16d}$$

$$0 \leq \rho \leq \rho^{max} \tag{2.16e}$$

$$\kappa \in \mathbb{B}, \tag{2.16f}$$

which are equivalent to (1.26).

2.2.3 Avoiding Oscillations

We can apply various strategies in order to avoid high oscillations of the control variables in the optimal solution. Two of them are described in the sequel.

A: Reducing number of control variables. Instead of using different control variables at each time-step, we decide a-priori at which points in time switching of a control variable is allowed. Then, we can simply use the same variable for the whole period which we do not want to allow for switching. This can also be interpreted in the way that the control variable lives on a coarser time grid than the other variables. If we transform this idea back to the original continuous model, we assume the control parameter to be a piecewise constant function in time, where the time of the jumps is previously fixed, but not the value of the function after each jump. The main disadvantage of this procedure is the fact that the points in time of the switching has to be fixed previously and are not up to optimisation. The main advantage is that no additional constraints for the MIP are required and even the number of variables is reduced, which leads to a slight reduction of complexity of the MIP.

B: Deriving additional constraints on switching behaviour. Another technique is to add constraints to the MIP guaranteeing that the time period between the switching of a control ranges between a certain prescribed upper and lower bound.

The following lemmata show, how these constraints can be designed in the case that the control variables are binary.

Lemma 2.2.6. ***Lower bound on switching period***
Let $C^t \in \mathbb{B}$ be a set of control variables $\forall\, t = 1, \dots, n_t$ and let L_0 and $L_1 \in \mathbb{N}$ denote the lower bounds for the number of consecutive time-steps that C^t can be set to 0 or 1, respectively.

Variables C^t respect the minimal switching period if and only if the following constraints hold:

$$\sum_{l=t+1}^{t+L_1} C^l \geq L_1(-C^t + C^{t+1}), \qquad \forall t \leq n_t - L_1 \tag{2.17}$$

$$\sum_{l=t+1}^{t+L_0} C^l \leq (L_0 + 1)(1 - C^t + C^{t+1}), \qquad \forall t \leq n_t - L_0. \tag{2.18}$$

Proof. We consider both directions separately.

$\Longrightarrow$:
When there is a switch from 0 to 1 after time-step t, i.e. if $C^t = 0 \wedge C^{t+1} = 1$, then the next L_1 control variables have to be 1 as well, i.e.

$$\sum_{l=t+1}^{t+L_1} C^l \stackrel{!}{=} L_1 \tag{2.19}$$

has to be fulfilled. Hence, (2.17) holds. Considering constraint (2.18), we have

$$\sum_{l=t+1}^{t+L_0} C^l \leq (L_0 + 1)(1 - C^t + C^{t+1}) = (L_0 + 1) \cdot 2.$$

This holds as well, since C_l cannot be greater than one.

When there is a switch from 1 to 0 after time-step t, i.e. if $C^t = 1 \wedge C^{t+1} = 0$, then the next L_0 control variables have to be zero as well. Hence,

$$\sum_{l=t+1}^{t+L_1} C^l \stackrel{!}{=} 0. \tag{2.20}$$

In that case $\sum_{l=t+1}^{t+L_1} C^l \geq L_1(-C^t + C^{t+1}) = -L_1$ is fulfilled, since C_l is always larger than zero, and constraint (2.18) is also fulfilled, since $(L_0 + 1)(1 - C^t + C^{t+1}) = 0$.

For the remaining cases, i.e. if $C^t = C^{t+1} = 0$ or $C^t = C^{t+1} = 1$ holds, constraint (2.17) holds with $\sum_{l=t+1}^{t+L_1} C^l \geq L_1(-C^t + C^{t+1}) = 0$, as well as constraint (2.18) with $\sum_{l=t+1}^{t+L_0} C^l \leq (L_0 + 1)(1 - C^t + C^{t+1}) = (L_0 + 1)$, since C^t cannot be larger than one.

$\Longleftarrow$:
Assume, there is a switch from 0 to 1 at time t, i.e. $C^t = 0$ and $C^{t+1} = 1$, and the following values for C would not respect the minimal switching period L_1. Then

$$\sum_{l=t+1}^{t+L_1} C^l \leq L_1,$$

which contradicts constraint (2.18).

Now, we consider the case of a switch from 1 to 0 at time t, i.e. $C^t = 1$ and $C^{t+1} = 0$. If the consecutive control variables would not respect the minimal switching period L_0, this would yield

$$\sum_{l=t+1}^{t+L_0} C^l \geq 0,$$

which clearly contradicts (2.17).

This completes the proof. □

In some applications, it might also be desired to guarantee an upper bound for the switching period. As an example, think of a traffic light which should not be red for longer than 3 minutes.

Lemma 2.2.7. ***Upper bound on switching period:***
Let $C^t \in \mathbb{B}$ be a set of control variables $\forall\, t = 1, \ldots, n_t$, U_0 and $U_1 \in \mathbb{N}$ denote the upper bound for the number of consecutive time-steps that C can be set to 0 or 1, respectively.

Variables C^t respect the maximal switching period if and only if the following constraints hold:

$$\sum_{l=t+1}^{t+U_1+1} C^l \leq U_1, \qquad \forall t \leq n_t - U_1 - 1 \tag{2.21}$$

$$\sum_{l=t+1}^{t+U_0+1} C^l \geq 1, \qquad \forall t \leq n_t - U_0 - 1. \tag{2.22}$$

Proof. If C^l is never set to one more than U_1 times in a row, constraint (2.21) holds and vice versa. In the same way constraint (2.22) holds if and only if C^l is never set to 0 more than U_0 times in a row. □

The main disadvantage of this approach is that the complexity of the MIP increases, since another set of constraints is added for every time-step. Furthermore, it becomes more involved to find feasible control settings which can be used for bounding heuristics

in order to speed up the optimisation algorithm. This is due to the fact that in many DTN-MIPs the control constraints usually describe relations between different control variables for the same time-step, which enables us to find feasible control settings for each time-step separately; whereas constraints of type (2.18), (2.17), (2.22) or (2.21) lead to further dependencies of control variables of different time-steps. For more details see Subsection 2.2.4. The main advantage of this method is that the switching time itself is also up to optimisation (in contrast to method A), which is especially useful in applications, where the choice of the switching times has a big influence on the optimal solution.

2.2.4 Tuning the Branch & Bound Optimisation

As described above, it is possible to reformulate optimisation problems on DTN models as linear DTN-MIPs. Thus it is possible to give it into one of various available optimisation solvers, e.g. [23]. Unfortunately, we need constraints and variables for each time-step leading to a large problem size. Often, this does not allow to obtain an optimal solution during an acceptable time frame. Especially because of the large number of binary variables, it is extremely difficult for the blackbox solver to find feasible solutions at all.

By using the optimisation software as a black box tool, we deprive the solver of a lot of valuable information; in fact, for us it is easy to construct a feasible solution manually: We only have to find a feasible setting for the control variables (in our example models the worker distribution and traffic light setting respectively). Given those variables, we can simply simulate the solution using a forward solver, see Algorithm 3, to obtain the state variables and then apply linearisation techniques to compute the linearisation variables.

After these simple computations, we can provide the solver with a feasible start solution. This procedure is illustrated in Figure 2.5.

However, this is not always enough to reduce the optimisation time, since it takes still a long time to find more feasible solution during the Branch & Bound Algorithm. For a more detailed study on applying starting heuristics for large DTN-MIPs within the production context we refer to [37] and [98]. Additionally, it is promising to have the solver branch only at the original control variables and provide another primal

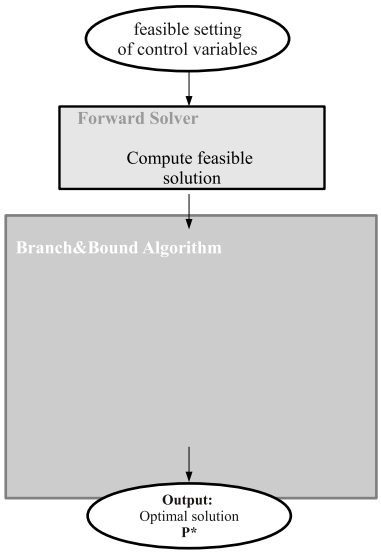


Figure 2.5: Using a starting heuristic to provide feasible start solution for Branch & Bound Algorithm.

bound for each new subproblem by finding a new feasible setting for these variables and applying the forward solver.

To clarify this concept, we consider the different steps of the Branch & Bound Algorithm, that has been presented in the beginning of this chapter, see Algorithm 2.

First of all we prescribe solely the control variables for branching, see line 8 in Algorithm 2. After that, we create a heuristic for finding a feasible solution of the considered subproblem, see line 6. Let $\tilde{I}$ be the index set indicating all variables, that have been fixed due to former branching. Respecting the fixed values x_i, $i \in \tilde{I}$, we first compute the dual bound by applying the Simplex Algorithm to the relaxed MIP (see line 5). If we construct the constraints for the control variables in a certain way, see Remark 2.2.2, it is possible that the optimal solution of the relaxed problem already contains control variables which are integer feasible. A bounding heuristic, see Algorithm 4, will keep the values of the integer feasible variables as well as the fixed branching variables and sets the others in a feasible way; and if possible, in a way that a good objective function value is obtained. This is illustrated in Figure 2.6. The resulting feasible solution is used as a primal bound for the Branch & Bound Algorithm in line 6. Figure 2.7 illustrates on which point of the Branch & Bound Algorithm the

Bounding Heuristic takes effect.

A bounding heuristic can only work properly, when the branching only takes place on the control variables. Otherwise some binary linearisation variables might be fixed in a branch. These fixed values can not be respected by the bounding heuristic, since it applies the forward solver to obtain the values of the linearisation variables.

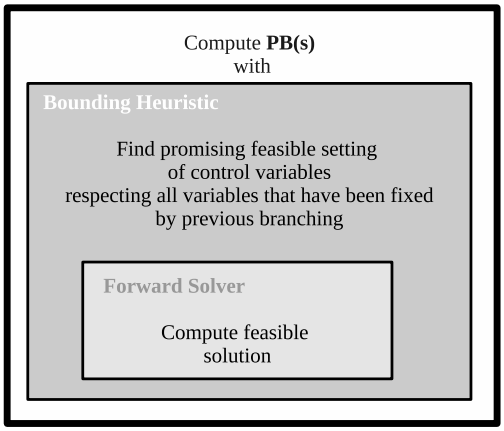


Figure 2.6: Structure of the bounding heuristic. It serves as a building block of the Branch & Bound Algorithm, see Figure 2.7.

Remark 2.2.8. *Typically, the control variables have indices for time-steps t and network arcs i. For the sake of simplicity, we skip the superindices t in the description of Algorithm 4.*

Later in this chapter, a more detailed bounding heuristic is described for the model of traffic light optimisation, see Algorithm 6 and 7.

In Section 2.4, we apply these ideas to find an optimal traffic light setting for traffic networks. In Chapter 3, Figure 3.44(a), 3.45(a) and 3.46(a) show the evolution of primal and dual bounds during the optimisation process applying only a starting heuristic on the one hand (cf. Figure 2.5) and the incorporated bounding heuristic (cf. Figure 2.7) on the other hand. The strong improvements of the second technique are convincing and enable us to find close to optimal solutions in a reasonable time for DTN-MIPs with around 10^6 variables and constraints.

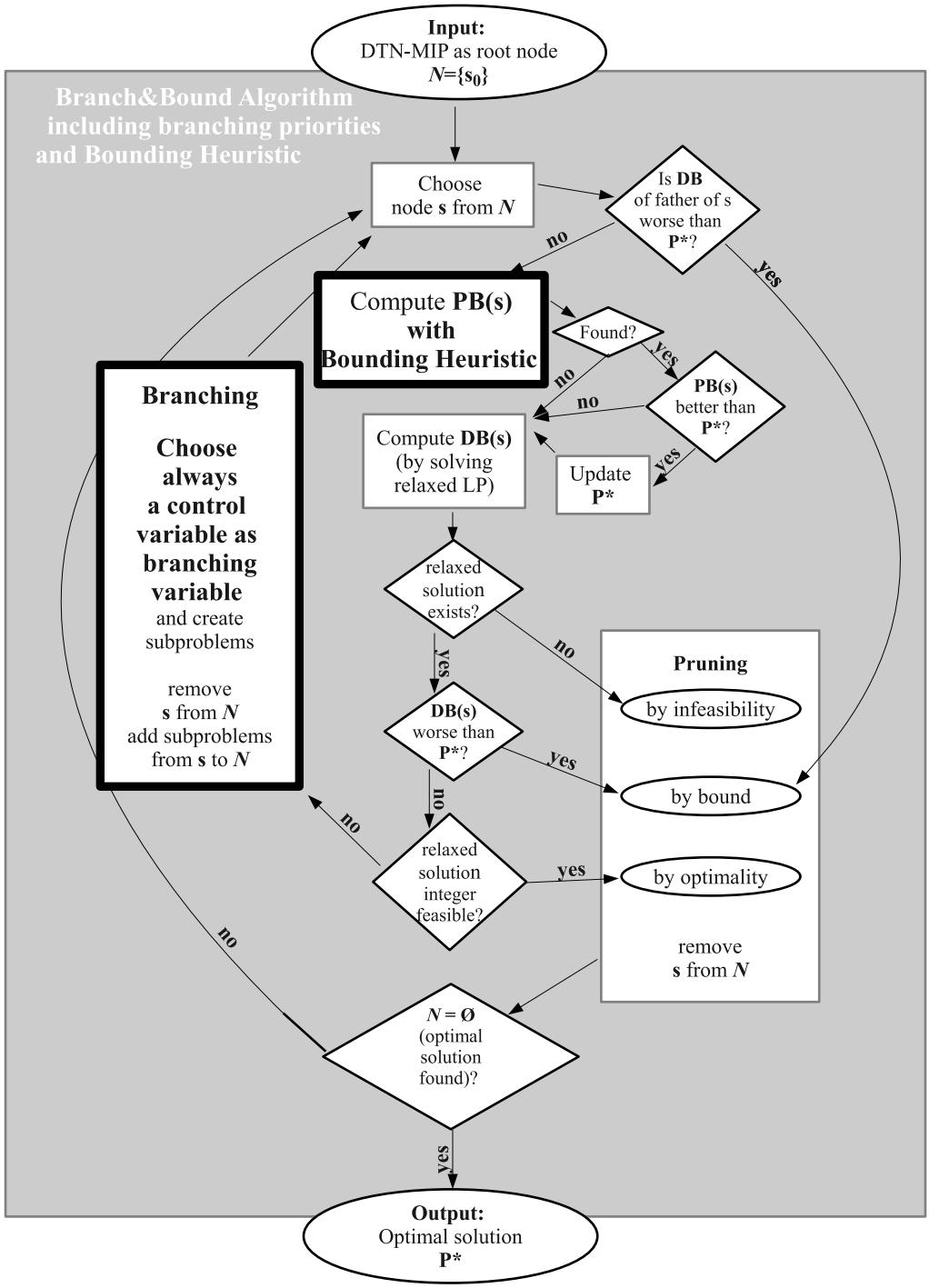


Figure 2.7: Using branching priorities and bounding heuristic iteratively during the Branch & Bound Algorithm.

Algorithm 3: Forward Solver.

```
/* Input: DTN-MIP and a feasible setting of control variables.
   */
/* Output: An integer feasible solution                        */
1 begin
2     for t = 0, ..., nt − 1 do
3         Compute coupling for all junctions of the network.
4         Apply PDE-solver (i.e. one time iterative of a discretisation scheme
          presented in Section 1.4) for each edge to obtain state variables (i.e.
          density, flow, buffer level etc.) for time-step t + 1.
5         Compute the corresponding values for linearisation variables.
6     Return feasible solution.
```

Algorithm 4: Bounding Heuristic.

```
/* Input: A relaxed solution (i.e. all variables are considered
   to be continuous) of the linear MIP with control variables
   C_i, i ∈ I and already fixed branching variables with index
   i ∈ Ĩ ⊂ I.                                                  */
/* Output: An integer feasible solution                        */
1 begin
      /* Optional: Include integer feasible variables of relaxed
         solution in fixed index set:                          */
2     for i ∈ I\Ĩ do
3         if C_i ∈ ℤ then
4             Ĩ → Ĩ ∪ {i}
      /* Find feasible setting of control variables, i.e. fulfilling
         all control conditions. Do it in a way that a good objective
         function value of the original DTN-MIP is supported.  */
5     Compute controls C_i, i ∈ I\Ĩ. /* see e.g. Algorithm 6 or 7  */
6     Apply Forward Solver(C_i, i ∈ I) /* see Algorithm 3          */
7     Return feasible solution.
```

2.3 Application I: Optimal Worker Scheduling for Production Networks

Considering DTN-models in the production context, we encounter a wide range of optimisation questions containing discrete decisions. Typically, the aim is to reduce storage costs or increase output of products. [42, 46] look for the best flow distribution inside a production network in order to achieve minimal queuing sizes and maximal production flow. In addition to that, in [37, 98] choices of certain supplier configurations are up to optimisation. [48, 57] control policies for processors on network models where parts have different priorities. In this section, we want to consider the model introduced in Section 1.2, where the point of interest is the optimal scheduling of workers throughout the network such that the production is maximised, see also [49], using the techniques described in the previous section.

2.3.1 Deriving a linear DTN-MIP

Following the steps of Subsection 2.2.1, we derive a linear DTN-MIP of the model derived in Section 1.2.

I. Control variables and constraints. Our goal is to find out which worker schedule is best in order to guarantee a stable production process and maximise the possible outflow of goods. Hence, our control variables represent the worker distribution $\beta_i(t)$ as percentage of a total number of workers W.

To get a well-defined allocation of available workers, we state the following control constraints:

$$\sum_{i \in E} \beta_i(t) = 1, \quad 0 \leq \beta_i(t) \leq 1, \qquad \forall i, t. \tag{2.23}$$

This means, we distribute all workers among the machines and require them to be a positive number. Furthermore, it is reasonable to only allow integer workers, hence we request

$$W \cdot \beta_i(t) \in \mathbb{Z}, \qquad \forall i, t. \tag{2.24}$$

II. Objective function value and continuous optimisation problem. Since we are interested in increasing the ouflow, we use the total outflow of the network as objective function, see (2.25a). Respecting the conditions of the DTN model derived in Section 1.2, we end up with the following ODE-restricted optimisation problem:

$$\max \int_0^T \sum_{i \in E^{out}} f_i(t) dt \tag{2.25a}$$

$$\text{s. t. } \forall i \in E:$$

$$\frac{du_i(t)}{dt} = Bf(t) + f_{ext,i}(t) - f_i(t) \tag{2.25b}$$

$$f_i(t) = \min\left\{c_i(t), \frac{u_i(t)}{\tau_i}\right\}. \tag{2.25c}$$

$$\frac{dc_i(t)}{dt} = \min\left\{\frac{\mu_i - c_i(t)}{\epsilon}, W d_i \beta_i(t)\right\} - \min\left\{\frac{c_i(t)}{\epsilon}, l_i\right\} \tag{2.25d}$$

$$u_i(0) = u_{0i}, \quad c_i(0) = c_{0i} \tag{2.25e}$$

$$\sum_{i \in E} \beta_i(t) = 1, \quad 0 \leq \beta_i(t) \leq 1, \qquad \forall i, t. \tag{2.25f}$$

$$0 \leq \beta_i(t) \leq 1 \tag{2.25g}$$

$$W \cdot \beta_i(t) \in \mathbb{Z}_0^+, \tag{2.25h}$$

$$0 \leq c_i(t) \leq \mu_i, \quad u_i(t) \geq 0. \tag{2.25i}$$

The constraints consist of the coupled ODE-system, given by the buffer level equation (2.25b) and (2.25c), the capacity drop (2.25d) (or (1.23), depending on the model version) and the control constraints (2.25f). Furthermore, we need to prescribe initial conditions (2.25e) and the control conditions (2.25f), (2.25g) and (2.25h). If not said otherwise, we choose as initial condition empty buffers (i.e. $u_{0i} = 0$) and full capacities (i.e. $c_{0,i} = \mu_i$). Constraint (2.25i) represents additional non-negativity and box constraints of the state variables representing capacity and buffer level.

III. Discretising constraints. We choose a uniform discrete time grid

$$\mathcal{T} = \{t : t = 0, \ldots, n_t\}$$

of the underlying time horizon $[0, T]$ where n_t denotes the number of grid points. The step-size is defined via $\Delta t = \frac{T}{n_t}$.

Then, we discretise the system of ordinary differential equations (2.25b) - (2.25d) using the explicit Euler scheme.

Remark 2.3.1. *For the step-size Δt, the condition*

$$\Delta t := \min\{2\tau_i, 2\epsilon\}, \quad \forall i \in E$$

must be satisfied when dealing with stiff problems.

As an intermediate result, we get the following optimisation problem $\forall i, t \in \mathcal{T}$:

$$\max \sum_{i \in E^{out}} \sum_{t \in \mathcal{T}} f_i^t \cdot \Delta t$$

s. t.

$$u_i^{t+1} = u_i^t + \Delta t \cdot [[B \cdot f]_i + f_{ext,i}^t - f_i^t] \tag{2.26a}$$

$$c_i^{t+1} = c_i^t + \Delta t \cdot [D_i^t - R_i^t] \tag{2.26b}$$

$$\sum_{i \in E} \beta_i^t = 1 \tag{2.26c}$$

$$u_i^0 = u_{0i}, \quad c_i^0 = c_{0i} \tag{2.26d}$$

$$0 \leq \beta_i^t \leq 1 \tag{2.26e}$$

$$W \cdot \beta_i(t) \in \mathbb{Z}_0^+. \tag{2.26f}$$

$$0 \leq c_i^t \leq \mu_i, \quad u_i^t \geq 0, \quad , \tag{2.26g}$$

with $R_i^t := \min\{\frac{c_i^t}{\epsilon}, l_i\}$, $D_i^t := \min\{\frac{\mu_i - c_i^t}{\epsilon}, W d_i \beta_i^t\}$ and $f_i^t := \min\{c_i^t, \frac{u_i^t}{\tau_i}\}$. Note, that for the flow dependent capacity decrease as presented in (1.23), equation (2.26b) changes to

$$c_i^{t+1} = c_i^t + \Delta t \cdot [D_i^t - l_i \cdot f_i^t]. \tag{2.27}$$

IV. Applying linearisation techniques. In order to arrive at a linear MIP we linearise the min-terms in R_i^t, D_i^t and f_i^t with respect to c_i^t, u_i^t and β_i^t as shown in Lemma 2.2.3.

In this way, we get for R_i^t the following constraints:

$$l_i \cdot \kappa_i^t \leq R_i^t \leq l_i \tag{2.28a}$$

$$\frac{c_i^t}{\epsilon} - \overline{M} \cdot \kappa_i^t \leq R_i^t \leq \frac{c_i^t}{\epsilon}, \tag{2.28b}$$

where $\overline{M} := \frac{\mu_i}{\epsilon}$ and κ_i^t is binary, i.e. $\kappa_i^t \in \{0, 1\}$. By applying the same method to f_i^t, we end up with

$$g_i^t \leq f_i^t \leq c_i^t \tag{2.29a}$$

$$\frac{u_i^t}{\tau_i} - M \xi_i^t \leq f_i^t \leq \frac{u_i^t}{\tau_i}, \tag{2.29b}$$

where M is a sufficiently large constant, ξ_i^t are additional binary variables and

$$g_i^t := c_i^t \cdot \xi_i^t. \tag{2.30}$$

We linearise $c_i^t \cdot \xi_i^t$ as shown in equations (2.5) - (2.8). Hence, with the new linearisation variable $g_i^t \in \mathbb{R}$, (2.30) can be described by

$$0 \leq g_i^t \leq \mu_i \xi_i^t \tag{2.31a}$$

$$c_i^t - \mu_i(1 - \xi_i^t) \leq g_i^t \leq c_i^t. \tag{2.31b}$$

Following Lemma 2.2.3 and taking computational runtime into account, we need an estimate for the constant M in (2.29). More precisely, it is important to choose M as tight as possible. Hence, let M depend on i and t and make sure that

$$M_i^t \geq \frac{u_i^t}{\tau_i}, \tag{2.32}$$

holds $\forall i,\, t \in \mathcal{T}$. Therefore, we consider (2.26a) in order to derive an upper bound for u_i^t:

$$u_i^t \leq u_i^{t-1} + \Delta t \cdot [B \cdot \mu]_i + \Delta t \cdot f_{ext,i}^{t-1},$$

where μ denotes a vector-valued function with entries μ_i for each machine. From our initial conditions we know that $u_i^0 = u_{0,i}$. By iteration, we get

$$u_i^t \leq t \cdot \Delta t \cdot [B \cdot \mu]_i + \Delta t \cdot \sum_{\bar{t}=0}^{t-1} f_{ext,i}^{\bar{t}} + u_{0,i}$$

and thus

$$M_i^t := \frac{1}{\tau_i} t \cdot \Delta t \cdot [B \cdot \mu]_i + \frac{1}{\tau_i} \Delta t \cdot \sum_{\bar{t}=0}^{t-1} f_{ext,i}^{\bar{t}} + \frac{u_{0,i}}{\tau_i}. \tag{2.33}$$

Remark 2.3.2. *In this way, we have also gained an upper bound for u_i^t, namely*

$$0 \leq u_i^t \leq \tau_i \cdot M_i^t, \quad \forall i,\, t \in \mathcal{T}. \tag{2.34}$$

Note, that it is advantageous to keep box constraints as tight as possible, since this might lead to smaller Branch & Bound trees which reduce the runtime.

The missing linearisation of D_i^t is done analogously. This leads to the following additional constraints:

$$W d_i \cdot h_i^t \leq D_i^t \leq W d_i \beta_i^t \tag{2.35a}$$

$$\frac{\mu_i - c_i}{\epsilon} - \frac{\mu_i}{\epsilon} \cdot \gamma_i^t \leq D_i^t \leq \frac{\mu_i - c_i^t}{\epsilon} \tag{2.35b}$$

$$0 \leq h_i^t \leq \gamma_i^t \tag{2.35c}$$

$$\beta_i^t - (1 - \gamma_i^t) \leq h_i^t \leq \beta_i^t, \tag{2.35d}$$

where $\gamma_i^t \in \{0,1\}$ and $h_i^t, \in \mathbb{R}$ is a new set of linearisation variables describing the nonlinearity $\beta_i^t \cdot \gamma_i^t$.

Finally, we specify box and binary constraints for all new variables:

$$0 \leq h_i^t \leq 1,\ 0 \leq g_i^t \leq \mu_i, \tag{2.36a}$$

$$0 \leq f_i^t \leq \mu_i,\ 0 \leq R_i^t \leq l_i,\ 0 \leq D_i^t \leq W \cdot d_i, \tag{2.36b}$$

$$\kappa_i^t, \gamma_i^t, \xi_i^t \in \{0,1\}\ \forall i, t \in \mathcal{T}. \tag{2.36c}$$

In summary, the complete mixed-integer formulation reads

$$\max \sum_{i \in E^{out}} \sum_{t \in \mathcal{T}} f_i^t \cdot \Delta t \tag{2.37}$$

s. t.

(2.26), (2.28), (2.29), (2.31), (2.34), (2.35), (2.36).

The optimisation problem consists of eight sets of different continuous variables $(c, u, \beta, f, g, h, R, D)$ and two to three sets of binaries $((\kappa,)\gamma, \xi)$ depending on the chosen version of capacity modelling. All variables depend on the number of edges and time-steps. Hence, the problem size is $\mathcal{O}(n_t \cdot |E|)$.

V. Avoiding fluctuations of the optimal solution. In this application, it is meaningful to introduce further restrictions on the time evolution of the distribution functions β_i. It does not make sense for the workers to change their position too frequently. In order to avoid strong fluctuations, we apply method A of Subsection 2.2.3 by previously fixing the possible switching times $0 < \tilde{t}_1 < \ldots < \tilde{t}_L < n_t$ of the workers. Thereafter, the same control variable $\beta_i^{\tilde{t}_j}$ is used for time-steps $t : \{\tilde{t}_j \leq t < \tilde{t}_{j+1}\}$.

2.3.2 Steady State Analysis

Since for DTN-MIP optimisation times gets unacceptably large as soon as we increase the time horizon T, it is advisable to use another approach, when questions about the long term behaviour of the system are of interest. Furthermore, it is usually desired to obtain a steady state shortly after the start-up of a productions system. In this subsection we show, how a steady state analysis can be done for given the DTN-MIP. This analysis provides us with information about the necessary number of workers to get a stable flow considering the average through-flow the network for a long time horizon. Additionally, it is easy to find out, how many workers we need in order to obtain a specific production flow. If the specific application is flexible with respect to the flow distribution at branching points, we can also optimise the flow distribution inside the network in order to obtain the most efficient setting. Hence, the computation methods of the desired workers and the flow distribution can be used as a preliminary technique to fix the model parameters W and B, before optimizing the corresponding DTN-MIP. Another advantage of the steady state analysis is that it serves as a tool to derive a heuristic for incumbent solutions: We can use the optimal worker distribution of the steady state case as a promising starting distribution. From this it is possible to compute a feasible starting solution to speed up the start of the optimisation algorithm.

Definition 2.3.3. *A solution of (2.25) is called steady state solution, if* $\frac{du_i}{dt} = 0$ *and* $\frac{dc_i}{dt} = 0$ *and* $\beta_i(t)$ *is independent on* t $\forall i \in E$.

Hence, we drop the time index t in the steady state case.

Time independent capacities. We compute the capacities c_i for the steady state solution. The capacities in equilibrium are computed by

$$\min\{\frac{\mu_i - c_i}{\epsilon}, Wd_i\beta_i\} - \min\{\frac{c_i}{\epsilon}, l_i\} = 0 \quad \forall i. \tag{2.38}$$

The steady state c_i can be determined in the following way:

$$c_i = \begin{cases} \mu_i - \epsilon l_i, & \text{if } \mu_i \geq 2\epsilon l_i \quad \wedge \quad \beta_i \geq \frac{l_i}{Wd_i} \quad \text{(Case 1.1)} \\ \epsilon W d_i \beta_i & \text{if } \text{—— " ——} \quad \wedge \quad \beta_i < \frac{l_i}{Wd_i} \quad \text{(Case 1.2)} \\ \frac{1}{2}\mu_i, & \text{if } \mu_i < 2\epsilon l_i \quad \wedge \quad \beta_i \geq \frac{\mu_i}{2\epsilon W d_i} \quad \text{(Case 2.1)} \\ \epsilon W d_i \beta_i & \text{if } \text{—— " ——} \quad \wedge \quad \beta_i < \frac{\mu_i}{2\epsilon W d_i} \quad \text{(Case 2.2)} \end{cases} \tag{2.39}$$

It can be checked that this choice of c_i fulfills (2.38) by considering every case separately. Considering the limit process $\epsilon \to 0$, we get: Either $c_i = \mu_i$, if there are enough workers to balance the breakdown rate (Case 1.1), or $c_i = 0$ (Case 1.2). Case 2.1 and Case 2.2 will never occur, if μ_i is positive.

Time independent flow. Next, we investigate the buffer levels u_i in the ODE-constraint (2.25b) - (2.25c). In steady state, we have

$$[B \cdot f]_i + f_{ext,i} - f_i = 0, \quad \forall i. \tag{2.40}$$

Since c_i as well as u_i have to be constant in steady state, $f_i := \min\{c_i, \frac{u_i}{\tau_i}\}$ is also constant. This means, if we find variables f_i such that

$$[B \cdot f]_i + f_{ext,i} - f_i = 0 \quad \text{and} \tag{2.41}$$

$$0 \leq f_i \leq c_i \tag{2.42}$$

hold, we can set $u_i := f_i \cdot \tau_i$. In that way $f_i = \frac{u_i}{\tau_i} \leq c_i$ holds, and thus (2.25c) is automatically fulfilled. Apparently, equation (2.41) is only true, if the external inflow f_{ext} is constant as well.

Remark 2.3.4. *Enforcing the conservation of mass, cf. (1.18) in the steady state case, we get*

$$\int_0^t \sum_{i \in E} f_{ext,i} \, d\tilde{t} = \int_0^t \sum_{i \in E^{out}} f_i \, d\tilde{t}.$$

Since f_{ext} and f are time independent

$$t \cdot \sum_{i \in E} f_{ext,i} = t \cdot \sum_{i \in E^{out}} f_i, \tag{2.43}$$

i.e. the total external inflow matches the total outflow.

Maximizing the outflow. In the original optimisation problem (2.25), we obtain a worker distribution such that the outflow of the network is maximal. The same can be done in the stationary case. Unlike in (2.25), for the steady state solution holds (2.43). Hence, there would be nothing to optimise, if we previously prescribe the external inflow. For this reason we leave the inflow as control.

External inflow is only possible at certain edges $e \in E^{in} \subset E$.

We reformulate (2.41) and end up with the following constraints:

$$[B \cdot f]_i - f_i + f_{ext,i} = 0 \qquad \forall i \in E^{in} \tag{2.44a}$$

$$[B \cdot f]_i - f_i = 0 \qquad \forall i \notin E^{in}. \tag{2.44b}$$

Consequently, the steady state optimisation problem reads:

$$\begin{aligned} &\max \sum_{i \in E^{out}} f_i \\ &\text{s. t.} \\ &\text{(2.25f), (2.25h), (2.25i), (2.39), (2.42), (2.44)} \end{aligned} \tag{2.45}$$

In order to solve (2.45) with respect to the worker distribution β and external inflow f_{ext}, we linearise (2.39) using the techniques presented in Subsection 2.2.2 and obtain the following linear MIP:

$$\max \sum_{i \in E^{out}} f_i \tag{2.46a}$$

$$\text{such that } \forall i \in E$$

$$[B \cdot f]_j - f_j + f_{ext,j} = 0 \qquad \forall j \in E^{in} \tag{2.46b}$$

$$[B \cdot f]_j - f_j = 0 \qquad \forall j \notin E^{in} \tag{2.46c}$$

$$-d_i W(1-\delta_i) \leq l_i - W d_i \beta_i \leq l_i \delta_i \tag{2.46d}$$

$$-\mu_i \delta_i \leq c_i - \mu_i + \epsilon l_i \leq (\mu_i + \epsilon l_i) \cdot \delta_i \tag{2.46e}$$

$$-\epsilon W d_i (1-\delta_i) \leq c_i - \epsilon W d_i \beta_i \leq \mu_i (1-\delta_i) \tag{2.46f}$$

$$\sum_{i \in E} \beta_i = 1 \tag{2.46g}$$

$$W \cdot \beta_i \in \mathbb{Z}_0^+ \tag{2.46h}$$

$$0 \leq \beta_i \leq 1, \quad 0 \leq f_i \leq c_i, \quad 0 \leq c_i \leq \mu_i \tag{2.46i}$$

$$\delta_i \in \{0,1\}. \tag{2.46j}$$

Constraints (2.46d) to (2.46f) ensure that c is set according to (2.39). (2.46d) has the effect that δ_i is set to 0, when $\beta_i \geq \frac{l_i}{W d_i}$; otherwise it is set to one. (2.46e) guarantees that c_i is set to $\mu_i - \epsilon l_i$, when $\delta_i = 0$. In the same way (2.46f) ensures that c_i is set to $\epsilon W d_i \beta_i$ in the case $\delta_i = 1$.

The optimisation problem (2.46) enables us to compute the long term behaviour of the DTN. When we compute the maximal outflow for different number of workers W, we find out, how many workers are needed on average to guarantee a desired production outflow considering a long term production.

Remark 2.3.5. *For the modified model (1.23), the dynamics of the capacity is directly connected to f. For this reason, it is not possible to compute the capacities for a given worker distribution a-priory, as in (2.39). Nevertheless, we can derive a steady state model. In this case equations (2.46d) to (2.46f) have to be replaced by*

$$D_i - l_i \cdot f_i = 0, \tag{2.47}$$

where $D_i = \min\{\frac{\mu_i - c_i}{\epsilon}, W \cdot d_i \cdot \beta_i\}$ and linearised as in (2.35). In the case, we want to set ϵ to zero, D_i in (2.47) can simply be replaced by $W \cdot d_i \cdot \beta_i$, since the upper bound of c_i is already guaranteed by (2.46i).

Relation to max flow problems. At this point, it is rather simple to include an additional optimisation task, namely the optimisation of the flow distribution at branching nodes (nodes with more than one outgoing edge). As we have seen, matrix B prescribes the behaviour of the flow at vertices. However, we could instead use an incidence matrix, which only describes the incoming and outgoing edges of a vertex, without fixing the distribution rates. This makes the model more flexible and leads to larger flows as shown later in Section 3.1.

The incidence matrix K is constructed as follows. Given a network with n edges and m vertices, we have $K \in \mathbb{Z}^{m \times n}$, whose elements are set in the following way:

$$k_{v,i} = \begin{cases} 1, & \text{if } i \text{ is incoming edge of } v \\ -1, & \text{if } i \text{ is outgoing edge of } v \\ 0, & \text{else.} \end{cases}$$

The corresponding incidence matrix K of the exemplary network shown in Figure 2.8 is given by

$$K = \begin{pmatrix} -1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & -1 & 0 \\ 0 & 0 & 1 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

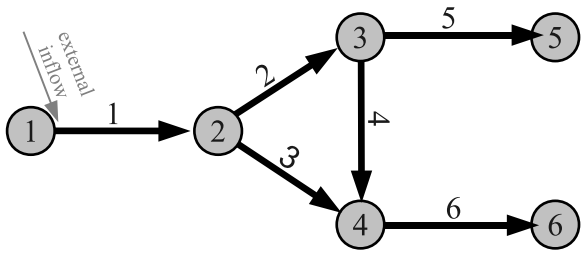


Figure 2.8: Example of a network, $V_{in} = \{v_1\}$ and $V_{out} = \{v_5, v_6\}$.

Consequently, it is possible to include the issue of optimizing the flow distribution by exchanging constraint (2.44) with

$$[K \cdot f]_v = 0 \qquad \forall v \in V \backslash (V_{in} \cup V_{out}) \tag{2.48a}$$

$$[K \cdot f]_v + f_{ext,v} = 0 \qquad \forall i \in V_{in}. \tag{2.48b}$$

Hence, an improved steady state optimisation problem is

$$\max \sum_{i \in E^{out}} f_i \tag{2.49}$$

s. t.

(2.25f), (2.25h), (2.25i), (2.39), (2.42), (2.48)

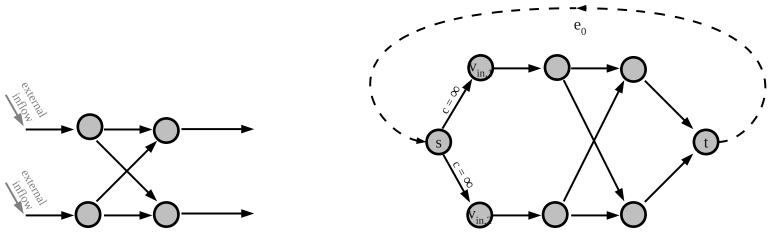
In summary, we note that model (2.45) yields the optimal steady state solution for fixed distribution parameters B, whereas model (2.49) additionally computes the optimal routing of goods. Another difference to the previous model (2.45) is that the external inflow is specified at vertices, and not at edges. Since we usually want the inflow to enter the network at edges without predecessors, we can easily assign the external inflow to the startvertices without changing the setting.

Remark 2.3.6. *It is also possible to use the incidence matrix K for the dynamic model (2.37). But this might lead to highly fluctuating flow distributions in the optimal solution. Since the rates do not explicitly appear as variables in the MIP, they cannot be restricted to be constant in time, cf. [37, 42].*

Choosing a worker distribution β and computing the capacities c according to (2.39), we end up with a well known problem of graph theory, the Maximum Flow Problem (MFP). In the sequel, we will use theoretical results from MFPs to prove the existence of a solution to (2.49).

Lemma 2.3.7. *Given a network $G = (V, E)$ with properties $l_i, d_i > 0$ and $\mu_i, \epsilon \geq 0$, $\forall i \in E$, there exists a feasible solution of (2.49) with $\sum_{i \in E^{out}} f_i = \sum_{v \in V_{in}} f_{ext,v} \geq 0$.*

Proof. Let β be an arbitrary worker distribution, satisfying (2.25f) - (2.25c). The upper bound of the flow is given by (2.39) and satisfies $c_i \geq 0$, $\forall i \in E$. The network can be transformed in the following way: We can imagine the external inflow as edges from a source vertex s to the point where the external inflow is supposed to enter the network. The upper bound c of these edges is set to infinity. In the same way we can add an extra sink vertex t where all outflow edges are led to. Furthermore, we add an artificial edge e_0 from the sink to the source node, which represents the total through-flow, cf. Figure 2.9.



(a) (b)

Figure 2.9: Transformed network.

From (2.42) we know that the lower bound of the flow in each edge is 0. This setting fulfills the conditions for the existence of a feasible flow circulation stated in Hoffman's circulation theorem, see [6], Theorem 3.8.2. Taking the properties of a flow for vertex s and t into account as defined in graph theory, it directly follows that $f_{e_0} = \sum_{i \in E^{out}} f_i = \sum_{v \in V_{in}} f_{ext,v}$ holds, due to the construction of the transformed network. □

2.4 Application II: Traffic Networks - Optimal Traffic Light Setting

Several approaches on finding optimal signal timing on road networks can be found in the literature, see [14, 96], amongst others. Based on cell transmission models, which are an approximation to the Lighthill, Witham and Richards model, mixed-integer formulations and heuristic solution approaches have been developed, see [8, 53, 75, 76, 77]. These models can be applied to small junctions and aim for an optimal cycle length of the signal timing. Different to these approaches, we do not especially aim for an optimal cycle length, but want to optimise the switching times with respect to previously known statistical boundary flows which might underly strong changes during different times of the day. Additionally, we want to optimise road networks containing complex junctions containing several lanes for different turning directions. For several traffic models investigation has been done for complex urban intersections, as for example in [39].

Based on the DTN model presented in Section 1.3, we derive a model for traffic lights at junctions and derive a linear mixed integer optimisation problem that enables us to optimise the traffic light setting in order to obtain the maximal through-flow the network.

Modelling of complex junctions. First of all, we introduce a model for complex traffic light junctions, where different lanes for different turning direction are used. We model each of these lanes by a separate edge. We have given a flow distributin matrix d, see Definition 1.1.7. Hence, $d_{i,j}$ represents the percentage of traffic that is going from road i to road j. $d_{i,j}$ is set to zero for all invalid directions. As example, see Figure 2.10, the corresponding distribution is found in Table 2.1.

Now, we derive a DTN-MIP based on the traffic network model of Section 1.3. We follow the strategy of Subsection 2.2.1. We indicate in parenthesis (I) to (VI) to which step of the strategy we refer to.

2.4.1 Control Variables and Constraints (I)

In this context, the control variables are given by the traffic light setting.

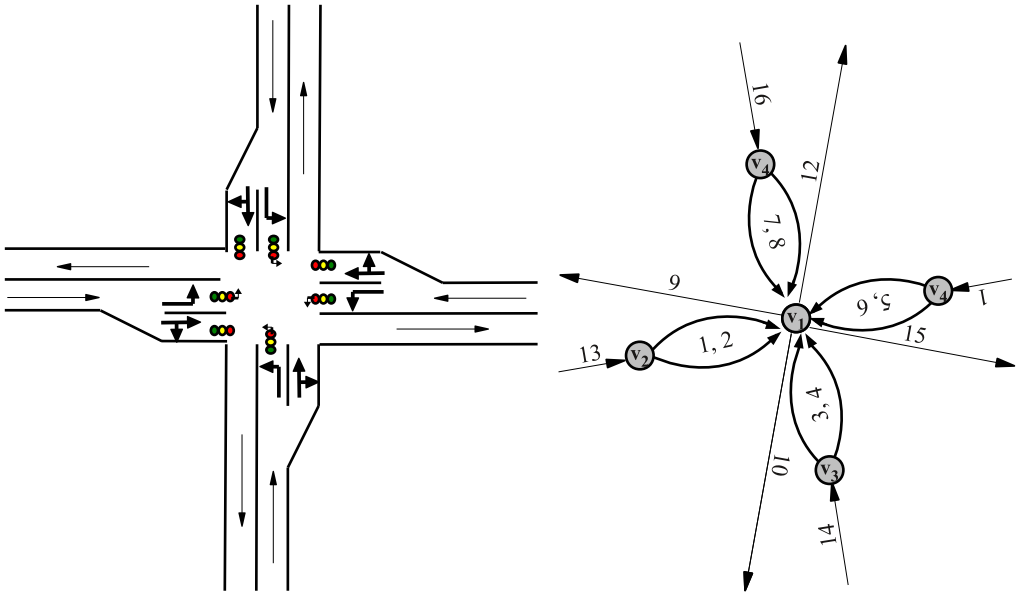


Figure 2.10: Crossover, where each lane for different turning directions is modelled by a separate edge.

	9	10	11	12
1	0	0	0	$d_{1,12}$
2	0	$d_{2,10}$	$d_{2,11}$	0
3	$d_{3,9}$	0	0	0
4	0	0	$d_{4,11}$	$d_{4,12}$
5	0	$d_{5,10}$	0	0
6	$d_{6,9}$	0	0	$d_{6,12}$
7	0	0	$d_{7,11}$	0
8	$d_{8,9}$	$d_{8,10}$	0	0

Table 2.1: Distribution parameters for vertex v_1 of 8x4-junction.

Modelling of traffic lights. We model a traffic light at the end of a road i by piecewise constant functions $A_i : t \mapsto \mathbb{B}$. When $A_i(t) = 0$ for some t, it means that the traffic light is red at this point in time and otherwise it is green.

The traffic light is included into the model by adding the following constraints for all incoming roads i:

$$\hat{f}_i(t) \leq A_i(t) \cdot \hat{F}_i(t), \quad \forall i \in \delta_v^{in}$$

When we consider a road network that also contains junctions without traffic light, we set the corresponding traffic light variable A_i to one.

Secure setting. We assume that the traffic lights are set in a save way. This means that traffic to an outgoing road cannot come from more than one incoming road simultaneously. Thus, a secure traffic light setting should satisfy the following constraint:

$$\sum_{i \in \delta_v^{in} : d_{ij} > 0} A_i \leq 1, \ \forall j \in \delta_v^{out}. \tag{2.50}$$

Depending on the specific case an even stronger restriction might be required. Imagine a big junction, where left-turning vehicles should only have green light, when the straightforward driving opposing traffic has red light, see Figure 2.11.

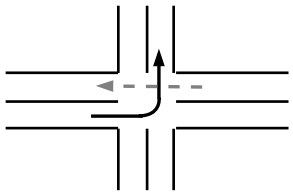


Figure 2.11: Examining secure traffic light settings.

For that reason we introduce the following notation:

Definition 2.4.1. *A **secure set** $S \subset E$ is the index set of traffic lights that must not be green simultaneously. The superset $\mathbb{S}$ contains all secure sets of a given network, $\mathbb{S} := \{S_k, k = 1, \dots |\mathbb{S}|\}$.*

We can guarantee a secure traffic light setting by respecting the following constraints:

$$\sum_{i \in S_k} A_i \leq 1, \ \forall S_k \in \mathcal{S}. \tag{2.51}$$

When the secure sets are chosen reasonably, constraint (2.51) includes (2.50).

The description of $\mathcal{S}$ is not unique. The easiest way to create $\mathcal{S}$ for a given network is the following:

- Take one traffic light i and check one by one, which other traffic light j must not be green at the time as i.
- Create a secure set for each pair – and take in mind not to count a pair twice.

If we apply this strategy to the junction shown in Figure 2.10, we end up with 20 secure sets, each containing two elements:

$$\begin{aligned}\mathcal{S} = \{&\{1,3\}_1, \{1,4\}_2, \{1,6\}_3, \{1,7\}_4, \{1,8\}_5, \{2,3\}_6, \{2,4\}_7, \\ &\{2,5\}_8, \{2,7\}_9, \{2,8\}_{10}, \{3,5\}_{11}, \{3,6\}_{12}, \{3,8\}_{13}, \{4,5\}_{14}, \\ &\{4,6\}_{15}, \{4,7\}_{16}, \{5,7\}_{17}, \{5,8\}_{18}, \{6,7\}_{19}, \{6,8\}_{20}\}\end{aligned} \tag{2.52}$$

However, this is not the best formulation, compare Remark 2.2.2. To clarify this concept, let us take a deeper look at the relaxed formulation: When we neglect the binary restrictions on the variables A_i, the control constraints yield:

$$\sum_{i \in S_k} A_i \leq 1, \ \forall S_k \in \mathcal{S} \tag{2.53a}$$

$$0 \leq A_i \leq 1, \ \forall i \in E. \tag{2.53b}$$

The feasible region of (2.53) is far from being the convex hull of the integer programming problem

$$\sum_{i \in S_k} A_i \leq 1, \ \forall S_k \in \mathcal{S} \tag{2.54a}$$

$$A_i \in \mathbb{B}, \ \forall i \in E. \tag{2.54b}$$

A better formulation of the secure sets can be derived by combining as many of the element pairs of the above constructed sets S_k as possible. If we look carefully at the given example, we observe that there are four groups of four pairwise different roads

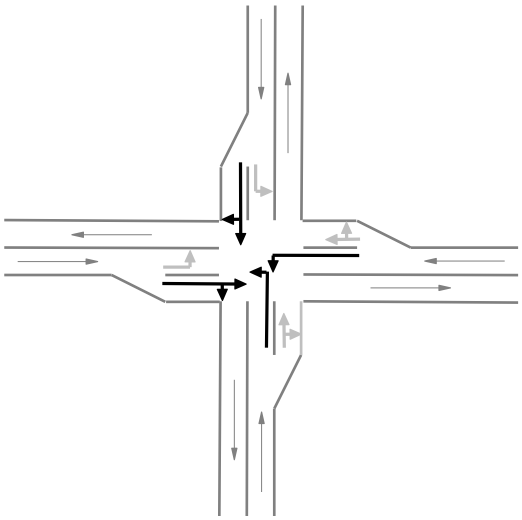


Figure 2.12: Roads whose traffic lights cannot be green at the same time.

which must not be green simultaneously for each combination of the lights contained in the set. One of them is depicted in Figure 2.12. The others are obtained by rotating the setting.

In this way, we obtain an alternative formulation $\tilde{\mathcal{S}}$, given by:

$$\begin{aligned}\tilde{\mathcal{S}} = \{&\{2,4,5,7\}_1,\ \{1,4,6,7\}_2,\\ &\{1,3,6,8\}_3,\ \{2,3,5,8\}_4\}.\end{aligned} \tag{2.55}$$

Let P be the polyhedron described by $\mathcal{S}$ and $\tilde{P}$ the polyhedron described by $\tilde{\mathcal{S}}$, i.e.

$$P := \{x \in \mathbb{R}^8 : \sum_{i \in S_k} x_i \leq 1, \forall S_k \in \mathcal{S} \ \wedge\ 0 \leq x_i \leq 1, \forall i = 1, \ldots, 8\} \tag{2.56}$$

and

$$\tilde{P} := \{x \in \mathbb{R}^8 : \sum_{i \in \tilde{S}_k} x_i \leq 1, \forall \tilde{S}_k \in \tilde{\mathcal{S}} \ \wedge\ 0 \leq x_i \leq 1, \forall i = 1, \ldots, 8\}. \tag{2.57}$$

Claim 2.4.2. *$\tilde{P}$ is a better formulation for (2.54) than P, i.e.*

$$\textit{i):}\ \{x \in \mathbb{B}^8 : \sum_{i \in S_k} x_i \leq 1, S_k \in \mathcal{S}\} = \{x \in \mathbb{B}^8 : \sum_{i \in \tilde{S}_k} x_i \leq 1, \tilde{S}_k \in \tilde{\mathcal{S}}\} \tag{2.58a}$$

$$\textit{and ii):}\ \tilde{P} \subsetneqq P. \tag{2.58b}$$

Proof. (2.58a) holds due to construction. In both cases, the only integer feasible points are given by

$$\mathcal{X} = \left\{ \begin{pmatrix} 0\\0\\0\\0\\0\\0\\0\\0 \end{pmatrix}, \begin{pmatrix} 1\\0\\0\\0\\0\\0\\0\\0 \end{pmatrix}, \begin{pmatrix} 0\\1\\0\\0\\0\\0\\0\\0 \end{pmatrix}, \dots, \begin{pmatrix} 0\\0\\0\\0\\0\\0\\0\\1 \end{pmatrix}, \right.$$

$$\left. \begin{pmatrix} 1\\1\\0\\0\\0\\0\\0\\0 \end{pmatrix}, \begin{pmatrix} 1\\0\\0\\0\\1\\0\\0\\0 \end{pmatrix}, \begin{pmatrix} 0\\1\\0\\0\\0\\1\\0\\0 \end{pmatrix}, \begin{pmatrix} 0\\0\\1\\1\\0\\0\\0\\0 \end{pmatrix}, \begin{pmatrix} 0\\0\\1\\0\\0\\0\\1\\0 \end{pmatrix}, \begin{pmatrix} 0\\0\\0\\0\\1\\0\\0\\1 \end{pmatrix}, \begin{pmatrix} 0\\0\\0\\0\\1\\1\\0\\0 \end{pmatrix}, \begin{pmatrix} 0\\0\\0\\0\\0\\0\\1\\1 \end{pmatrix} \right\} \quad (2.59)$$

With respect to (2.58b), we first show that $\tilde{P} \subset P$:

Choose an arbitrary $x \in \tilde{P}$. Thus, $0 \leq x_i \leq 1$ holds for all $i = 1, \dots, 8$. Taking the order of the secure sets as stated in (2.52) and (2.55) into account, we have

$$\sum_{i \in \tilde{S}_1} x_i \leq 1 \;\Rightarrow\; \sum_{i \in S_j} x_i \leq 1, \text{ for secure sets } S_j \in \mathcal{S}, \text{ with } j = 7, 8, 9, 14, 16, 17,$$

$$\sum_{i \in \tilde{S}_2} x_i \leq 1 \;\Rightarrow\; \sum_{i \in S_j} x_i \leq 1, \text{ for secure sets } S_j \in \mathcal{S}, \text{ with } j = 2, 3, 4, 15, 16, 19,$$

$$\sum_{i \in \tilde{S}_3} x_i \leq 1 \;\Rightarrow\; \sum_{i \in S_j} x_i \leq 1, \text{ for secure sets } S_j \in \mathcal{S}, \text{ with } j = 1, 3, 5, 12, 13, 20,$$

$$\sum_{i \in \tilde{S}_1} x_i \leq 1 \;\Rightarrow\; \sum_{i \in S_j} x_i \leq 1, \text{ for secure sets } S_j \in \mathcal{S}, \text{ with } j = 6, 8, 10, 11, 13, 18.$$

$\Rightarrow \tilde{P} \subset P$.

Now, we show that $\tilde{P} \subsetneq P$, i.e. $\exists x \in P : x \notin \tilde{P}$:

Choose $\hat{x} = (0\ 0.5\ 0\ 0.5\ 0.5\ 0\ 0\ 0)^T$. $\hat{x}$ fulfills

$$\sum_{i \in \S_k} \hat{x}_i \leq 1 \ \forall S_k \in \mathcal{S} \ \Rightarrow \ \hat{x} \in P.$$

But we have

$$\sum_{i \in \tilde{S}_1} \hat{x}_i = 1.5 > 1 \ \Rightarrow \ \hat{x} \notin \tilde{P}.$$

This proves the claim. □

The next subsection represents the second step of the strategy presented in Subsection 2.2.1. An objective function is formulated turning the DTN into a continuous optimisation problem.

2.4.2 Objective Function and Continuous Formulation of Optimisation Problem (II)

Our goal is to find a traffic light setting that enables the traffic participants to drive smoothly through the road network while encountering as few congestion as possible. For this reason we aim to maximise the flow overall the whole network. Considering the continuous notation, this means, we want to maximise the integral over time and space of the flow plus the integral over time of the coupling flow at junctions, i.e.

$$\max \sum_{i \in E} \Big(\int_0^T \int_0^{L_i} f(\rho_i(x,t)) dx dt + \int_0^T \hat{f}_i(t) dt \Big). \tag{2.60}$$

The constraints of the optimisation problem consist on the one hand of the flow function, the evolution of density along the roads and coupling conditions at the junctions, compare DTN model in Section 1.3. On the other hand, we need the above derived control constraints for the traffic light setting.

i) Flow. As stated in the derivation of the DTN model in Subsection 1.3, $\rho : (x,t) \mapsto \rho(x,t) \in [0, \rho^{max}] \subset \mathbb{R}^+$ denotes the density of cars, $x \in [0, L_i] \subset \mathbb{R}^+$ describes the location on road i with length L_i and $t \in (0,T) \subset \mathbb{R}$ denotes the time horizon.

For every road we specify a maximal density ρ_i^{max} and triangular flow functions f_i, given by (1.26). The prescribed flow function holds along the roads as well as for the

coupling at the junctions, i.e

flow along the roads:

$$f_i(x,t) = f_i(\rho_i(x,t)), \qquad \forall i \in E,\ x \in [0,\ L_i] \tag{2.61a}$$

coupling flow:

$$\hat{f}_i(t) = f(\hat{\rho}_i(t)), \qquad \forall i \in E \backslash E^{out} \tag{2.61b}$$

$$\bar{f}_i(t) = f(\bar{\rho}_i(t)), \qquad \forall i \in E \backslash E^{in}. \tag{2.61c}$$

ii) Density evolution along the roads. Along the roads, the continuity equations holds:

$$\begin{cases} \partial_t \rho_i + \partial_x f_i(\rho) = 0, \\ \rho_i(x,0) = \rho_i^0(x), \end{cases} \tag{2.62}$$

iii) Coupling. As in [19] we choose the coupling in a way that maximal possible flow is achieved at the junction with respect to (1.34), (1.35) and conservation of flow. For all junctions v with traffic lights, the coupling flow $\{\hat{f}_i, i \in \delta_v^{in}; \bar{f}_j, j \in \delta_v^{out}\}$ is the optimal solution of the following optimisation problem:

$$\max \sum_{i \in \delta_v^{in}} \gamma_i(t) \tag{2.63a}$$

such that

$$\gamma_j(t) = \sum_{i \in \delta_v^{in}} d_{ij}(t)\gamma_i(t)\ \forall j \in \delta_v^{out} \tag{2.63b}$$

$$0 \leq \gamma_i(t) \leq A_i(t) \cdot \hat{F}_i(t) \tag{2.63c}$$

$$0 \leq \gamma_j(t) \leq \bar{F}_j(t). \tag{2.63d}$$

Note, that these coupling optimisation problems are now constraints of our optimisation problem. Later (in Subsection 2.4.4) we show how to handle these nested optimisation problems and manage to rewrite them as linear constraints of the DTN-MIP without loosing any information or accuracy.

Additionally to the coupling flow, we need the following side constraints:

maximal possible coupling flow:

$$\bar{F}_i(t) := \begin{cases} f_i^{max}, & \text{if } 0 \le \rho_{0,i}(t) \le \rho_i^* \\ f_{0,i}(t), & \text{else} \end{cases} \qquad \forall i \in E \backslash E^{in} \tag{2.64a}$$

$$\hat{F}_i(t) := \begin{cases} f_{n,i}(t), & \text{if } 0 \le \rho_{n,i}(t) \le \rho_i^* \\ f_i^{max}, & \text{else} \end{cases} \qquad \forall i \in E \backslash E^{out} \tag{2.64b}$$

coupling density:

$$\bar{\rho}_j(t) \in \begin{cases} [0, \rho_j^*], & \text{if } 0 \le \rho_{0,j}(t) \le \rho_j^* \\ \{\rho_j^t(0)\} \cup [0, \tau(\rho_{0,j}(t))[, & \text{else} \end{cases} \qquad \forall i \in E \backslash E^{in} \tag{2.64c}$$

$$\hat{\rho}_i(t) \in \begin{cases} \{\rho_{n_i i}(t)\} \cup]\tau(\rho_{n_i i}(t)), \rho_i^{max}], & \text{if } 0 \le \rho_{n_i i}(t) \le \rho_i^* \\ [\rho_i^*, \rho_i^{max}] & \text{else} \end{cases} \qquad \forall i \in E \backslash E^{out} \tag{2.64d}$$

iv) Traffic lights. For the traffic light settings we need to prescribe the secure sets S_k. Then, the following constraints are required:

$$\sum_{i \in S_k} A_i(t) \le 1, \qquad \forall S_k \in \mathcal{S} \tag{2.65}$$

As explained later in Subsection 2.4.5, we can optionally include constraints (2.19) and (2.99b) in order to limit high fluctuations of the switching times.

2.4.3 Discretisation (III)

As in Section 1.4, we introduce discretisation grids. The discrete time grid is given by $\mathcal{T} = \{t : t = 0, \dots, n_t\}$ with time-step size Δt and number of time-steps $n_t := \lceil \frac{t}{\Delta t} \rceil$. T discrete spatial grid is given by $\mathcal{K} = \{k : k = 0, \dots, n_i\}$, where the spatial size is referred to as Δx and the number of space-steps is given $n_i := \lceil \frac{L_i}{\Delta x} \rceil$.

Depending on the numerical schemes used, we get requirements on the time-step size depending on the space-step size, as for example the CFL-condition, see (1.74).

The discrete formulation of the objective function (2.60) is

$$\max \sum_t \sum_{i \in E} \sum_{k=0}^{n_i} f(\rho_{k,i}^t) \Delta t \Delta x + \max \sum_t \sum_{i \in E} \hat{f}_i^t \Delta t. \tag{2.66}$$

We rewrite the constraints of the previous subsection using a discrete formulation of the variables, which are given by:

$$\text{control variables:} \quad A_i^t \qquad \in \mathbb{B} \tag{2.67a}$$

$$\text{state variables:} \quad 0 \leq f_{k,i}^t,\, \hat{f}_i^t,\, \bar{f}_i^t,\, \hat{F}_i^t,\, \bar{F}_i^t \leq f_i^{max}, \qquad \in \mathbb{R} \tag{2.67b}$$

$$0 \leq \rho_{k,i}^t,\, \hat{\rho}_i^t,\, \bar{\rho}_i^t \leq \rho_i^{max}, \qquad \in \mathbb{R} \tag{2.67c}$$

$$\forall i \in E,\, k \in \mathcal{K}_i,\, t \in \mathcal{T}.$$

For the discretisation of the PDE (2.62) we choose a numerical scheme $\mathcal{H}$, as presented in Section 1.4. We start with the initial density values $\rho_{k,i}^0$ and obtain the density values for the time-steps iteratively:

$\forall t = 1, \dots T-1$:

inner grid points of the road:

$$\rho_{k,i}^{t+1} = \mathcal{H}(\rho_{k+1,i}^t,\, \rho_{k,i}^t,\, \rho_{k-1,i}^t,\, f_{k+1,i}^t,\, f_{k,i}^t,\, f_{k-1,i}^t), \quad \forall i \in E,\, k = 1, \dots, n_i - 1 \tag{2.68a}$$

left boundary cell:

$$\rho_{0,i}^{t+1} = \mathcal{H}(\rho_{1,i}^t,\, \rho_{0,i}^t,\, \bar{\rho}_i^t,\, f_{1,i}^t,\, f_{0,i}^t,\, \bar{f}_i^t), \qquad \forall i \in E \backslash E^{in} \tag{2.68b}$$

right boundary cell:

$$\rho_{n,i}^{t+1} = \mathcal{H}(\hat{\rho}_i^t,\, \rho_{n,i}^t,\, \rho_{n-1,i}^t,\, \hat{f}_i^t,\, f_{n,i}^t,\, f_{n-1,i}^t), \qquad \forall i \in E \backslash E^{out} \tag{2.68c}$$

where $\mathcal{H}$ denotes the numerical scheme.

Remark 2.4.3. *For the sake of simplicity, we omitted the double index, i.e. we write* $\rho_{n_i,i}^t$ *instead of* $\rho_{n,i}^t$ *and so on.*

In the following subsection, we comment more detailed on the choice of the scheme $\mathcal{H}$, keeping in mind the linearisability of the whole optimisation problem.

2.4.4 Linearizing Constraints (IV)

To obtain a linear MIP we now apply the linearisation techniques as shown in Subsection 2.2.2.

i) Flow. For the linearisation of (2.61) we follow the steps described for linearizing the triangular flow function, see page 73. Consequently, we obtain the following set of linear constraints:

$$f^t_{k,i} = 2\lambda\tilde{\rho}^t_{k,i} - \rho^*_{k,i}2\lambda\kappa^t_{k,i} - \lambda\rho^t_{k,i} + \rho^*_{k,i}2\lambda \tag{2.69a}$$

$$0 \leq \rho^*_{k,i}\kappa^t_{k,i} - \tilde{\rho}^t_{k,i} + \frac{1}{2}\rho^t_{k,i} - \frac{1}{2}\rho^*_{k,i} \tag{2.69b}$$

$$0 \;\leq\; \tilde{\rho}^t_{k,i} \;\leq\; \rho^{max}_{k,i} \cdot \kappa^t_{k,i} \tag{2.69c}$$

$$\rho^t_{k,i} - \rho^{max}_{k,i}(1 - \kappa^t_{k,i}) \;\leq\; \tilde{\rho}^t_{k,i} \;\leq\; \rho^t_{k,i} \tag{2.69d}$$

$$0 \leq \rho^t_{k,i} \leq \rho^{max}_{k,i} \tag{2.69e}$$

$$\kappa^t_{k,i} \in \mathbb{B} \tag{2.69f}$$

$$\forall i \in E,\, k \in \mathcal{K}_i,\, t \in \mathcal{T}. \tag{2.69g}$$

The linearised expressions for (2.61b) and (2.61c) are derived analogously.

ii) Density evolution along the roads. At this point, we use a numerical scheme which leads to a linear connection of the state variables $\rho^t_{k,i}$ and $f^t_{k,i}$, as for example Lax-Friedrich-Scheme. Alternatively, it is also possible to introduce state variables $M^t_{k,i}$ that express the space derivative of the density and use the Hamilton-Jacobi Scheme that is derived in Subsection 1.4.2. Here, we will follow another approach: As we will see in the next paragraph, we can omit a complex linearisation of (2.64d) and (2.64c) when we use a numerical scheme that computes the boundary flow without involving the boundary density; i.e. we apply a scheme of the form

$$\rho^{t+1}_{k,i} = \mathcal{H}(\rho^t_{k,i},\, f^t_{k+1,i},\, f^t_{k,i},\, f^t_{k-1,i})$$

for $k \in \{0, n_i\}$. The staggered Lax-Friedrich-Scheme [61, 65], described in Subsection 1.4.1, equations (1.68) - (1.70), fulfills exactly these requirements.

iii) Coupling. The solution of the maximisation problem (2.63) is a constraint of our optimisation problem. In the discrete formulation, (2.63) has to be solved separately for every time-step t. In the following explanations, we will skip the time index.

Linearisation of the coupling flow. Consider a traffic light junction with an arbitrary amount of incoming and outgoing roads:

Let a set of incoming roads δ_v^{in} and a set of outgoing roads δ_v^{out} be given. Furthermore, let parameters $0 \leq d_{ij} \leq 1$ which fulfill $\sum_{j \in \delta_v^{out}} d_{ij} = 1$, upper bounds $F_i \geq 0$ for all $i \in \delta_v^{in}$ and $F_j \geq 0$ as well as traffic light parameters $A_i \in \mathbb{B}$ for all $i \in \delta_v^{in}$ and $j \in \delta_v^{out}$ be given.

In order to obtain the coupling flow $\hat{f}_i$ for all i, we have to find the maximal feasible flow at the junction. Hence, we have to solve the (LP):

$$\max \sum_{i \in \delta_v^{in}} \gamma_i \tag{2.70}$$

such that

$$\begin{aligned} &\sum_{i \in \delta_v^{in}} d_{ij}\gamma_i = \gamma_j, && \forall j \in \delta_v^{out} \\ &0 \leq \gamma_i \leq A_i \hat{F}_i, && \forall i \in \delta_v^{in} \\ &0 \leq \gamma_j \leq \bar{F}_j, && \forall j \in \delta_v^{out} \end{aligned} \tag{2.71}$$

Lemma 2.4.4. *If*

$$\begin{aligned} \hat{f}_i &= \min\{A_i \hat{F}_i, \underbrace{\frac{1}{d_{ij}} F_j - \sum_{k \in \delta_v^{in} \setminus i} \frac{d_{kj}}{d_{ij}} \hat{f}_k, ...}_{\forall j \in \{\delta_v^{out} : d_{ij} > 0\}}\} \\ \bar{f}_j &= \sum_{i \in \delta_v^{in}} d_{ij}\gamma_i = \gamma_j; \quad \forall j \in \delta_v^{out} \end{aligned} \tag{2.72}$$

holds for all $i \in \delta_v^{in}$ *and*

$$\bar{f}_j = \sum_{i \in \delta_v^{in}} d_{ij}\gamma_i, \ \forall j \in \delta_v^{out} \tag{2.73}$$

then $\{\hat{f}_i,\ i \in \delta_v^{in}\}$, *and* $\{\bar{f}_j,\ j \in \delta_v^{out}\}$, *is an optimal solution of (2.70).*

Proof. Since the flow of the outgoing roads γ_j does not appear in the objective function, (2.70) can be rewritten in the more condensed form

$$\max \sum_{i \in \delta_v^{in}} \gamma_i \tag{2.74}$$

such that

$$\sum_{i \in \delta_v^{in}} d_{ij} \gamma_i \leq \bar{F}_j, \quad \forall j \in \delta_v^{out} \tag{2.75}$$

$$0 \leq \gamma_i \leq A_i \hat{F}_i, \quad \forall i \in \delta_v^{in}. \tag{2.76}$$

We can transform (2.75) in terms of γ_i, which yields

$$\gamma_i \leq \frac{1}{a_{ij}} \bar{F}_j - \sum_{k \in \delta_v^{in} \setminus \{i\}} \frac{d_{ik}}{d_{ij}} \cdot \gamma_k, \ \forall i \in \delta_v^{in}, \ j \in \{\delta_v^{out} : \ d_{ij} \neq 0\}. \tag{2.77}$$

In this way, we get all conditions with respect to the upper bound of γ_i. Thus, γ_i is feasible for (2.74) if and only if it fulfills

$$0 \leq \gamma_i \leq \min\{A_i F_i, \underbrace{\frac{1}{d_{ij}} F_j - \sum_{k \in \delta_v^{in} \setminus i} \frac{d_{kj}}{d_{ij}} \gamma_k, ...\}}_{\forall j \in \{\delta_v^{out} : d_{ij} > 0\}}, \ \forall i \in \delta_v^{in}. \tag{2.78}$$

The feasible region of the linear program (2.74) is a polytope (i.e a bounded polyhedron) due to construction. It is not empty, since $\gamma_i = 0$ for all i is a feasible solution of (2.74). According to the fundamental theorem of linear programming, see for example [89], page 39, an optimal solution of (2.74) is an extreme point of the feasible region. An extreme point is defined as a point that cannot be expressed as a convex combination of any other two distinct points of the feasible region, see for example [89, 94]. Due to construction, $\hat{f}$ as stated in (2.79) has for all indices i an active constraint. Hence, it is an extreme point of the feasible region of (2.74). Together with (2.78) this yields that $\hat{f}$ is optimal for (2.74). □

Lemma 2.4.5. *In consideration of the constraint (2.51) and (2.71), equation (2.72) reduces to*

$$\hat{f}_i = \min\{A_i \hat{F}_i, \underbrace{\frac{1}{d_{ij}} \bar{F}_j, ...}_{\forall j \in \{\delta_v^{out} : d_{ij} > 0\}}\}. \tag{2.79}$$

Proof. Due to (2.51) for each outgoing road j at most one traffic light parameter is set to 1 for all roads that lead traffic to j. That means if $A_i = 1$, then $A_j = 0$ for all

$j : d_{ij} > 0$. In this case equation (2.72) reduces to (2.79). If $A = 0$, then the minimum of (2.79) is zero as well as the minimum of the terms of (2.72). Hence, for both cases, the claim is fulfilled. □

Due to Lemma 2.4.5, we can can replace the maximisation problem (2.63) by the minimum-expression (2.79) for every time-step $t \in \mathcal{T}$. These expressions can be linearised iteratively using the Lemma 2.2.3 and Lemma 2.2.4. For example, for the junction shown in Figure 2.10, there are not more than three terms per minimum expressions different from zero:

$$\hat{f}_i^t = \min\left\{A_i^t \hat{F}_i^t,\ \frac{1}{d_{i,j}} \bar{F}_j^t\right\} \tag{2.80}$$
$$\text{for } (i,j) \in \big\{(1,12),\ (3,9),\ (5,10),\ (7,11)\big\},$$

$$\hat{f}_i^t = \min\left\{A_i^t \hat{F}_i^t,\ \frac{1}{d_{i,j_1}} \bar{F}_{j_1}^t,\ \frac{1}{d_{i,j_2}} \bar{F}_{j_2}^t\right\} \tag{2.81}$$
$$\text{for } (i,j_1,j_2) \in \big\{(2,10,11),\ (4,11,12),\ (6,9,12),\ (8,9,10)\big\}.$$

Furthermore, dispersing junctions represent the lane split in front of the traffic light junction (as derived in Subsection 1.3.3). Here we get

$$\hat{f}_i = \min\left\{\hat{F}_i^t,\ \frac{1}{d_{i,j_1}} \bar{F}_{j_1}^t,\ \frac{1}{d_{i,j_2}} \bar{F}_{j_2}^t\right\} \tag{2.82}$$
$$\text{for } (i,j_1,j_2) \in \big\{(13,1,2),\ (14,3,4),\ (15,5,6),\ (16,7,8)\big\}.$$

Then, we linearise (2.80) - (2.82) as shown in Lemma 2.2.3 and Lemma 2.2.4 and additionally linearise expression $A_i^t \cdot F_i^t$ according to equations (2.5) - (2.8). This leads us to the following set of linear constraints for the network shown in Figure 2.10:

$\forall t \in \mathcal{T}$ and for $(i,j) \in \big\{(1,12),\ (3,9),\ (5,10)\,(7,11)\big\}$ we get:

$$\tilde{G}_i^t \le \hat{f}_i^t \le \hat{G}_i^t \tag{2.83a}$$
$$\frac{1}{d_{i,j}} \bar{F}_j^t - \frac{1}{d_{i,j}} f_j^{max} \beta_i^t \le \hat{f}_i^t \le \frac{1}{d_{i,j}} \bar{F}_j^t \tag{2.83b}$$
$$0 \le \tilde{G}_i^t \le f_i^{max} \beta_i^t \tag{2.83c}$$
$$\hat{G}_i^t - f_i^{max}(1 - \beta_i^t) \le \tilde{G}_i^t \le \hat{G}_i^t \tag{2.83d}$$

$\forall t \in \mathcal{T}$ and for $(i, j_1, j_2) \in$
$\big\{(13,1,2),\ (14,3,4),\ (15,5,6),\ (16,7,8),\ (2,10,11),\ (4,11,12),\ (6,12,9),\ (8,9,10)\big\}$
we get:

$$\tilde{G}_i^t \leq e_i^t \leq \hat{G}_i^t \tag{2.84a}$$
$$\frac{1}{d_{i,j_1}} \bar{F}_{j_1}^t - \frac{1}{d_{i,j_1}} f_{j_1}^{max} \beta_i^t \leq e_i^t \leq \frac{1}{d_{i,j_1}} \bar{F}_{j_1}^t \tag{2.84b}$$
$$\tilde{e}_i^t \leq \hat{f}_i^t \leq \hat{e}_i^t \tag{2.84c}$$
$$\frac{1}{d_{i,j_2}} \bar{F}_{j_2}^t - \frac{1}{d_{i,j_2}} f_{j_2}^{max} \eta_i^t \leq \hat{f}_i^t \leq \frac{1}{d_{i,j_2}} \bar{F}_{j_2}^t \tag{2.84d}$$
$$0 \leq \tilde{G}_i^t \leq f_i^{max} \beta_i^t \tag{2.84e}$$
$$\hat{G}_i^t - f_i^{max}(1 - \beta_i^t) \leq \tilde{G}_i^t \leq \hat{G}_i^t \tag{2.84f}$$
$$0 \leq \tilde{e}_i^t \leq f_i^{max} \eta_i^t \tag{2.84g}$$
$$\hat{e}_i^t - f_i^{max}(1 - \eta_i^t) \leq \tilde{e}_i^t \leq \hat{e}_i^t. \tag{2.84h}$$

For $i \in \big\{1, \dots 8\big\}$ we additionally need:

$$0 \leq G_i^t \leq f_i^{max} A_i^t \tag{2.85a}$$
$$\hat{M}_i^t - f_i^{max}(1 - A_i^t) \leq G_i^t \leq \hat{M}_i^t \tag{2.85b}$$

with

$$\hat{f}_i^t,\ \hat{F}_i^t,\ \bar{F}_{j_1}^t,\ \bar{F}_{j_2}^t,\ \bar{F}_j^t\ \in \mathbb{R}_0^+ \tag{2.86}$$

and linearisation variables

$$\beta_i^t,\ \eta_i^t \in \mathbb{B}, \tag{2.87a}$$
$$G_i^t,\ \tilde{G}_i^t,\ e_i^t, \tilde{e}_i^t \in \mathbb{R}_0^+, \tag{2.87b}$$

where G_i^t represents $A_i^t \cdot \hat{f}_i^t$ for $i = 1, \dots, 8$, and $G_i^t = \hat{M}_i^t$ for roads 13-16, which are without traffic light. Furthermore we have $\tilde{G}_i^t := \beta_i^t \cdot \hat{F}_i^t$, $e_i^t = \min\{\hat{F}_i^t,\ \frac{1}{d_{i,j_1}} \bar{F}_i^t\}$ and $\tilde{e}_i^t = \eta_i^t \cdot e_i^t$.

Linearisation of F_i. As before, we use the binaries $\kappa^t_{n,i}$ (and $\kappa^t_{0,i}$) which are set to 1, when the corresponding density value is smaller or equal than ρ^*_i and to 0, otherwise. As proposed by [50], we can reduce the number of constraints, when we already transform the resulting variables $f^t_{0,i}$ and $f^t_{n,i}$ according to (2.69). This yields

$$\begin{aligned}\hat{F}^t_i &= \kappa^t_{n,i} \cdot f^t_{n,i} + (1-\kappa^t_{n,i}) \cdot f^{max}_i \\ &\overset{(2.15a)}{=} \kappa^t_{n,i}\Big(\kappa^t_{n,i}\lambda_i\rho^t_{n,i} + (1-\kappa^t_{n,i})(\lambda_i(2\rho^*_i - \rho^t_{n,i}))\Big) + (1-\kappa^t_{n,i}) \cdot f^{max}_i \\ &= \kappa^t_{n,i}\lambda_i\rho^t_{n,i} + (1-\kappa^t_{n,i}) \cdot f^{max}_i, \end{aligned} \tag{2.88}$$

because $(\kappa^t_{n,i})^2 = \kappa^t_{n,i}$, since it is binary.

Analogously, we get

$$\begin{aligned}\bar{F}^t_i &= \kappa^t_{0,i} f^{max}_i - (1-\kappa^t_{0,i}) f^t_{0,i} \\ &\overset{(2.25c)}{=} \kappa^t_{0,i} f^{max}_i + (1-\kappa^t_{0,i})\Big(\kappa^t_{0,i}\lambda_i\rho^t_{0,i} + (1-\kappa^t_{0,i})(\lambda_i(2\rho^*_i - \rho^t_{0,i}))\Big) \\ &= 2f^{max}_i - \kappa^t_{0,i} f^{max}_i - \lambda_i\rho^t_{0,i} + \lambda_i\kappa^t_{0,i}\rho^t_i \end{aligned} \tag{2.89}$$

(2.88) and (2.89) together with the constraints (2.29) form the linear equivalence to (2.29), (2.64b) and (2.64a).

Now, we introduce variables $\tilde{f}^t_{0,i}$ and $\tilde{f}^t_{n,i}$ representing $\kappa^t_{0,i} \cdot f^t_{0,i}$ and $\kappa^t_{n,i} \cdot f^t_{n,i}$ and add the corresponding required linear constraints analogously as in (2.5) to (2.8).

Coupling density. The equations leading from the boundary density $\rho^t_{0,i}$ and $\rho^t_{n,i}$ to the coupling density $\hat{\rho}^t_i$ and $\bar{\rho}^t_i$, (2.64d) and (2.64c), are also linearisable:

Consider the upper part of equation (2.64d). We assume that ρ is known. We have to linearise

$$\begin{aligned} & y \in \{\rho\} \cup]\tau_\rho, \rho^{max}] \\ \Leftrightarrow \quad & y = \rho \wedge \tau_\rho < y \leq 1. \end{aligned} \tag{2.90}$$

We use $g \in]\tau_\rho, \rho^{max}]$ and sets

$$y = \xi\rho + (1-\xi)g$$

with $\xi \in \mathbb{B}$. Now we linearise $\tilde{g} := \xi \cdot g$ by

$$\begin{aligned} 0 \leq \tilde{g} &\leq \rho^{max} \cdot \xi \\ g - \rho^{max}(1-\xi) \leq \tilde{g} &\leq g, \end{aligned}$$

respectively. Hence, we get the linear constraint

$$y = \xi\rho + g - \tilde{g}. \tag{2.91}$$

In summary, we can replace (2.90) by

$$y = \xi\rho + g - \tilde{g} \tag{2.92a}$$

$$0 \leq \tilde{g} \leq 1 \cdot \xi \tag{2.92b}$$

$$g - \rho^{max}(1 - \xi) \leq \tilde{g} \leq g \tag{2.92c}$$

$$\xi \in \mathbb{B} \tag{2.92d}$$

$$\tau_\rho < g \leq \rho^{max} \tag{2.92e}$$

Since strict inequalities, such as (2.92e), are difficult to handle for MIP-solvers, we introduce a very small tolerance value $0 < \epsilon \ll 1$, and replace (2.92e) by $\tau_\rho + \epsilon \leq g \leq \rho^{max}$.

Analogously, we can linearise the set-constraint of (2.64c).

Taking account the techniques to linearise the if-else-construction of (2.64d) and (2.64c) and applying (2.92).

For all junctions $v \in V$ and for all $t \in \mathcal{T}$ we obtain the following set of constraints.

For the coupling density of incoming roads $i \in \delta_v^{in}$:

$$\hat{\rho}_i^t = \tilde{z}_i^t + y_i^t - \tilde{y}_i^t \tag{2.93a}$$

$$0 \leq \frac{1}{2}\rho_i^* - \frac{1}{2}\rho_{n,i}^t - \rho_i^*\zeta_i^t + \tilde{\rho}_{n,i}^t \tag{2.93b}$$

$$0 \leq \tilde{y}_i^t \leq \rho_i^{max} \cdot \zeta_i^t \tag{2.93c}$$

$$y_i^t - \rho_i^{max}(1 - \zeta_i^t) \leq \tilde{y}_i^t \leq y_i^t \tag{2.93d}$$

$$0 \leq \tilde{z}_i^t \leq \rho_i^{max} \cdot \zeta_i^t \tag{2.93e}$$

$$z_i^t - \rho_i^{max}(1 - \zeta_i^t) \leq \tilde{z}_i^t \leq z_i^t \tag{2.93f}$$

$$0 \leq \tilde{\rho}_{n,i}^t \leq \rho_i^{max} \cdot \zeta_i^t \tag{2.93g}$$

$$\rho_{n,i}^t - \rho_i^{max}(1 - \zeta_i^t) \leq \tilde{\rho}_{n,i}^t \leq \rho_{n,i}^t \tag{2.93h}$$

$$y_i^t = \mathring{\rho}_{n_i}^t + g_i^t - \tilde{g}_i^t \tag{2.93i}$$

$$0 \leq \mathring{\rho}_{n,i}^t \leq \rho_i^{max} \cdot \xi_i^t \tag{2.93j}$$

$$\rho_{n,i}^t - \rho_i^{max}(1 - \xi_i^t) \leq \mathring{\rho}_{n,i}^t \leq \rho_{n,i}^t \tag{2.93k}$$

$$0 \leq \tilde{g}_i^t \leq \rho_i^{max} \cdot \xi_i^t \tag{2.93l}$$

$$g_i^t - \rho^{max}(1 - \xi_i^t) \leq \tilde{g}_i^t \leq g_i^t \tag{2.93m}$$

$$2\rho_i^* - \rho_{n_i i}^t + \epsilon \leq g_i^t \leq \rho_i^{max} \tag{2.93n}$$

$$\rho_i^* \leq z_i^t \leq \rho_i^{max} \tag{2.93o}$$

For the coupling density of outgoing roads $j \in \delta_v^{out}$:

$$\bar{\rho}_j^t = \tilde{z}_j^t + y_j^t - \tilde{y}_j^t \tag{2.94a}$$
$$0 \leq \frac{1}{2}\rho_j^* - \frac{1}{2}\rho_{0,j}^t - \tilde{\rho}_{0,j}^t + \rho_j^* \zeta_i^t \tag{2.94b}$$
$$0 \leq \tilde{y}_j^t \leq \rho_j^{max} \cdot \zeta_j^t \tag{2.94c}$$
$$y_j^t - \rho_j^{max}(1 - \zeta_j^t) \leq \tilde{y}_j^t \leq y_j^t \tag{2.94d}$$
$$0 \leq \tilde{z}_j^t \leq \rho_j^{max} \cdot \zeta_j^t \tag{2.94e}$$
$$z_j^t - \rho_j^{max}(1 - \zeta_j^t) \leq \tilde{z}_j^t \leq z_j^t \tag{2.94f}$$
$$0 \leq \tilde{\rho}_{0,j}^t \leq \rho_j^{max} \cdot \rho_{0,j}^t \tag{2.94g}$$
$$\rho_{0,j}^t - \rho_j^{max}(1 - \rho_{0,j}^t) \leq \tilde{\rho}_{0,j}^t \leq \rho_{0,j}^t \tag{2.94h}$$
$$y_j^t = \mathring{\rho}_{0,j}^t + g_j^t - \tilde{g}_j^t \tag{2.94i}$$
$$0 \leq \tilde{g}_j^t \leq (2\rho_j^* - \rho_{oj}^t) \cdot \xi_j^t - \epsilon \tag{2.94j}$$
$$g_j^t - \rho^{max}(1 - \xi_j^t) \leq \tilde{g}_j^t \leq g_j^t \tag{2.94k}$$
$$0 \leq \mathring{\rho}_{0,j}^t \leq \rho_j^{max} \cdot \rho_{0,j}^t \tag{2.94l}$$
$$\rho_{0,j}^t - \rho_j^{max}(1 - \rho_{0,j}^t) \leq \mathring{\rho}_{0,j}^t \leq \rho_{0,j}^t \tag{2.94m}$$
$$0 \leq z_j^t \leq \rho_j^* \tag{2.94n}$$

with state variables

$$\hat{\rho}_i^t,\, \bar{\rho}_j^t,\, \rho_{n,i}^t,\, \rho_{0,j}^t \in \mathbb{R}_0^+, \tag{2.95}$$

linearisation variables

$$z_i^t,\, z_j^t,\, \tilde{z}_i^t,\, \tilde{z}_j^t,\, y_i^t,\, y_j^t,\, \tilde{y}_i^t,\, \tilde{y}_j^t,\, \tilde{\rho}_{n,i}^t,\, \tilde{\rho}_{0,j}^t,\, \mathring{\rho}_{n,i}^t,\, \mathring{\rho}_{0,j}^t,\, g_i^t,\, g_j^t,\, \tilde{g}_i^t,\, \tilde{g}_j^t \in \mathbb{R}_0^+ \tag{2.96}$$

and binaries

$$\zeta_i^t,\, \zeta_j^t,\, \xi_i^t,\, \xi_j^t \in \mathbb{B} \tag{2.97}$$

for all $i \in \delta_v^{in}$, $j \in \delta_v^{out}$, $v \in V$.

Resulting DTN-MIP. Altogether, we get a linear mixed integer optimisation Problem of the form

$$\begin{cases} \max (2.66) \\ \text{such that} \\ (2.67),\ (2.69),\ (1.68) \text{ - } (1.70),\ (2.83) \text{ - } (2.87),\ (2.88),\ (2.89), \\ \big((2.93) \text{ - } (2.97)\big). \end{cases} \tag{2.98}$$

In this formulation, the numerical scheme (1.68) - (1.70) can be replaced by any other easily linearisable numerical scheme $\mathcal{H}$.

However, the tolerance parameter ϵ that is needed for the linearisaton of the coupling density can lead to numerical instabilities. In the worst case, it can happen that the optimisation algorithm does not find a feasible solution at all, because of rounding errors.

As mentioned earlier, we can apply the strategy proposed by [50]. We completely omit the computation of the coupling densities (2.93) to (2.94), when we use a numerical scheme for the evolution of density on roads that does not need the coupling density in order to compute the boundary density, see Subsection 1.4.1. In this way, it is sufficient to only consider the boundary flow $\hat{f}_i^t$ and $\bar{f}_i^t$ to get all necessary information. To get this clearer, we refer to have a look at the forward solver, that is able to compute all model and linearisation variables once the traffic light setting is fixed, see Algorithm 5.

2.4.5 Additional Requirements on Switching Times (V)

It is desirable to avoid traffic light settings with highly frequent switching times, since this would not be applicable in real applications.

We follow approach **B** described in Subsection 2.2.3. In that way the switching times itself are subject to the optimisation process. In order to obtain a smooth flow, we prescribe a lower bound for the green phase, such that several cars have the chance to cross the junction. Furthermore, we want to spare any traffic member to wait for extremely long time in front of a red light, even if the road, he is coming from, is not very busy. As explained in Subsection 2.2.3, part **B**, we can meet these requirements by including additional constraints.

Let $L_g \in \mathbb{N}$ be the minimal number of time-steps, a traffic light is allowed to be green and U_r the maximal number of consecutive time-steps, a traffic light is allowed to be red.

Algorithm 5: Forward solver, to compute a feasbile solution for a prescribed traffic light setting.

/* Input: Traffic network model with initial conditions $\rho^0_{k,i} \forall i \in E, k \in \mathcal{K}_i$, outer boundary conditions $\rho^t_{0,i}, \forall i \in \mathcal{E}^{in}$, $t \in \mathcal{T}$ and feasible traffic light setting A^t_i, $\forall i \in E$, $t \in \mathcal{T}$. */

/* Output: An integer feasible solution, i.e. corresponding values for model and linearisation variables */

begin

  forall the $t = 0, \dots, nt$ **do**

    /* Compute state variables */

    Compute $\bar{F}^t_i$ and $\hat{F}^t_i$ according to (2.64a) and (2.64b), $\forall i \in E$.

    Compute coupling flow $\hat{f}^t_i$ and $\bar{f}^t_i$ according to (2.79) and (2.73), $\forall i \in E$.

    Compute flow $f^t_{k,i}$ according to (2.69), $\forall i \in E$, $k \in \mathcal{K}_i$.

    Apply numerical scheme for the roads for the next time-step to compute $\rho^{t+1}_{k,i}$ using (1.68) to (1.70). /* Compute linearisation variables */

    Set linearisation variables (2.87), $\tilde{f}^t_{k,i}$, etc. to the corresponding values, $\forall i \in E$, $k \in \mathcal{K}_i$.

According to equations (2.17) and (2.22), we have to add the following constraints for all roads i which possess a traffic light to our DTN-MIP:

$$\sum_{l=t+1}^{t+L_g} A_i^l \geq L_g(-A_i^t + A_i^{t+1}), \qquad \forall t \leq n_t - L_g \tag{2.99a}$$

$$\sum_{l=t+1}^{t+U_r+1} A_i^l \geq 1, \qquad \forall t \leq n_t - U_r - 1. \tag{2.99b}$$

2.4.6 Speeding up the Optimisation Algorithm (VI)

We use the idea described in Subsection 2.2.4 to speed up the optimisation procedure. We apply a bounding heuristic to compute integer feasible solution for each subnode of the tree (already starting with the root node), see Algorithm 4. To compute feasible control variables as done in line 5, we derive an algorithm which is adapted to the needs of the DTN-MIP (2.98). If we do not consider additional requirements on switching times, as described in Subsection 2.4.5, we use a greedy heuristic to find potentially good settings of control variables: We try to set those traffic light variables to one, which are close to one in the relaxed solution, as long as we do not violate the control condition (2.51). For more details see Algorithm 6. Having obtained the control variables, we run the forward solver, Algorithm 5, in order to get an integer feasible solution of (2.98) for the considered subnode in the Branch & Bound tree.

If we also want to include requirements on switching times, such as (2.99a) and (2.99b), it is a bit more involved to find a feasible setting of the traffic lights. This is due to the fact that we can not consider each time-step separately anymore. Instead of deriving a heuristic algorithm for this situation, we solve an additional optimisation problem, containing only the control conditions (2.51), (2.99a) and (2.99b). We want to use an objective function that supports good solutions for the original DTN-MIP (2.98). We follow a similar approach than in Algorithm 6: We consider the values of the traffic light parameters of the optimal relaxed solution, and formulate an objective function that rewards setting those variables to one, whose relaxed values are closest to one:

$$\max \sum_{t \in \mathcal{T}, i \in I_t^*} \tilde{A}_i^t \cdot A_i^t, \tag{2.100}$$

where I_t^* is the set of indices whose control variables have not been fixed during the branching process (and which are not integer feasible in the relaxed solution) and $\tilde{A}_i^t$ are the values of the optimal solution for the relaxed problem. Note, that $\tilde{A}_i^t$ are known values in this context and serve as coefficients. Hence, the objective function is linear. The whole procedure is described in Algorithm 7.

Algorithm 6: Compute Controls I.

```
/* Input: A DTN-MIP of the form (2.98), a set of fixed control
   variables A_i^t, t ∈ T, i ∈ Ĩ_t ⊂ E, relaxed solution with controls
   Ã_i^t ∈ ℝ.                                                    */
/* Output: A feasible setting of control variables
   A_i^t ∈ 𝔹, ∀(i,t) ∈ E × T with respect to (2.51)               */
1  begin
2      for t = 1, ..., nt do
           /* Consider the index set whose control variables are not
              fixed yet                                          */
3          Set I_t^* := E\Ĩ_t
4          while I_t^* ≠ ∅ do
               /* Find the traffic light variable closest to 1  */
5              find j = argmax_{i∈I_t^*} Ã_i^t.
6              if A_j^t = 1 does not violate constraint (2.51) then
7                  Set A_j^t = 1.
8              else
9                  Set A_j^t = 0.
10             Set Ĩ_t = Ĩ_t ∪ j. Set I_t^* = I_t^*\j.
11     Return control variables A_i^t, ∀(i,t) ∈ E × T.
```

In Chapter 3 we apply these Bounding Heuristics in the way they are illustrated in Figure 2.7. The improvement of the optimisation procedure compared to the use of a simple starting heuristic (cf. Figure 2.5) is remarkable. For more details, see Figure 3.44(a), 3.45(a) and 3.46(a).

Algorithm 7: Compute Controls II.

/* Input: A DTN-MIP of the form (2.98), a set of fixed control variables A_i^t, $t \in \mathcal{T}$, $i \in \tilde{I}_t \subset E$, relaxed solution with controls $\tilde{A}_i^t \in \mathbb{R}$. */

/* Output: A feasible setting of control variables $A_i^t \in \mathbb{B}$, $\forall (i,t) \in E \times \mathcal{T}$ with respect to (2.51), (2.99a) and (2.99b). */

1 **begin**

2 **for** $t = 1, \ldots, nt$ **do**

/* Consider the index set whose control variables are not fixed yet */

3 Set $I_t^* := E \backslash \tilde{I}_t$

4 Solve coupling IP:

$$\max \sum_{t \in \mathcal{T}, i \in I_t^*} \tilde{A}_i^t \cdot A_i^t$$

such that (2.51), (2.99a) (2.99b)

$A_i^t \in \mathbb{B}$, $\forall t \in \mathcal{T}$, $i \in I_t^*$

5 Return control variables A_i^t, $\forall (i,t) \in E \times \mathcal{T}$.

3

Results

In this chapter we present various results on the models derived in the course of this work. Section 3.1 is dedicated to considerations on the production network model with dynamic capacities, see Subsection 1.2.2 for the DTN and Section 2.3 for the corresponding DTN-MIP. We illustrate the behaviour of capacities, buffers and production flow and the effects of worker changes. Furthermore we analyse grid size dependencies on solutions and verify the conservation of mass (cf. Lemma 1.2.2). Next, we provide a detailed study on the steady state model, see Subsection 2.3.2, for a branched network and point out how we can exploit these results for the corresponding dynamic DTN-MIP. Finally, a real world example demonstrates the functionality of the model and shows, how production flow can be gained by skillfully appointing workers in under-staffed situations. Section 3.2 considers the traffic flow model, derived in Section 1.3. Firstly, simulation of the novel Hamilton-Jacobi-Algorithm (cf. Algorithm 1) are shown, compared to other schemes and used to derive car trajectories. Secondly, the DTN-MIP on traffic light optimisation, see Section 2.4, equation (2.98), is considered and improvements for optimal traffic light settings are pointed out. We also discuss the necessity of additional restrictions on switching times, as explained in Subsection 2.4.5 and compare the corresponding solutions. Thirdly, tuning techniques for the optimisation algorithm, as shown in Figure 2.5 and 2.7 are applied and compared. All computations are performed on a PC equipped with 16GB Ram, Intel(R) Xeon(R) CPU 5160 @ 3.00GHz.

3.1 Application I: Optimal Worker Scheduling for Production Networks

For computational experiments we use two different approaches. For the first small test case discussed in Subsection 3.1.1, we implemented the optimisation problem (2.25) in Matlab 7.5 using the function *fmincon*, which is a solver for nonlinear optimisation problems, see [87]. This approach works quite well as long as the test cases are small. The disadvantage of this solver is that it often gets stuck in local optima and it does not allow to restrict the worker distribution to integer workers. For these reasons we derived the mixed integer formulation (2.37) which can be used by Cplex 12.1.0, a commercial solver for linear mixed integer problems developed by IBM, formerly Ilog, see [23]. It uses a Branch & Cut algorithm providing the user with currently found primal as well as dual bounds during the optimisation process. In the case that the optimality gap tends to zero, the user can be sure that the provided solution is indeed globally optimal. Furthermore, this method has the advantage that we can easily restrict the workers to integer numbers, which is indeed meaningful for real world applications. In Subsection 3.1.2 this method is applied to a branched network where also steady state studies are carried out. Subsection 3.1.3 deals with a real-world example as originally introduced in [42, 47].

3.1.1 Model Behaviour on Processor Chain

Initially, we sketch the impact of numerical parameters on the result as well as introduce the modelling aspect of worker changes during the time horizon. Therefore, we take a small test example. We consider two machines in a row with a fixed parameter setting, see Figure 3.1 and 3.2.

Note that the breakdown parameter l is set to a relatively high value compared to the maximal processing capacity in order to better work out the effects how the worker distributions influence the overall outflow. Nevertheless, the length of a time unit can be interpreted according to the application and typically comprises a much larger period than one single production step. If we choose smaller breakdown rates, we have to set the time horizon to a much larger value to see significant effects, which also enlarges the computation time.

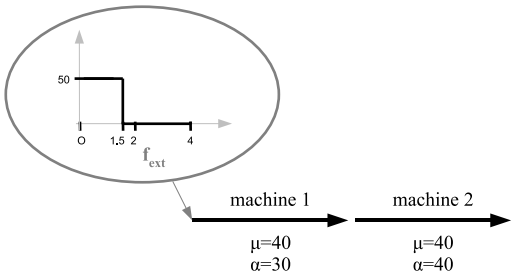


Figure 3.1: Two serial processors.

time horizon:	$T = 4$
throughput time:	$\tau = 0.25$
workers:	$W = 40$
Total inflow: $\int_0^T f_{ext}(\tilde{t})d\tilde{t} = \sum_{t\in\mathcal{T}} f_{ext}^t \Delta t$	=75 parts

Figure 3.2: Parameter setting.

Remark 3.1.1. *Note, that we do not use the linear MIP formulation in this subsection, but solve the optimisation problem via the first approach: by nonlinear, gradient-based optimisation methods, which does not allow for integer restrictions on workers.*

The time horizon is $T = 4$, the throughput times at buffers are $\tau_i = 0.25$ for every machine and 40 workers are available. The external inflow enters the network at the first machine. In the first 1.5 time units, we have an inflow of 50 and thus $\int_0^T f_{ext}(\tilde{t})d\tilde{t} = 75$ parts are introduced into the system. Furthermore, here and in all following examples, the repair times r_i are set to 1 for all edges.

As initial condition, we set c_{0i} to full capacities and assume empty queues in the beginning (i.e. $u_{0i} = 0$). Furthermore, we provide a start solution where the workers are equally distributed among the edges, i.e. we have 20 workers at each machine.

Numerical Investigations

We perform a simulation assuming the worker distribution rate β to be constant for the whole time horizon. We compute the objective function value (2.66) for different values of β, using the Matlab routine *fmincon* with explicit Euler discretization for the ODE-constraints. We let β_1 go from 0 to 1 in steps of length 0.001. In Figure 3.3, we compare the simulation results for different time grids. We can observe that the optimal objective function value tends to the same value, even for coarse time grids.

In this setting, the maximal outflow of 44.47 units is achieved, if 13.92 workers are sent to the first machine and 26.08 to the second one. As we can see in Table 3.1 and in accordance with Figure 3.3, the conservation of mass, as stated in Lemma 1.2.2, is kept with an accuracy that depends on the time grid size.

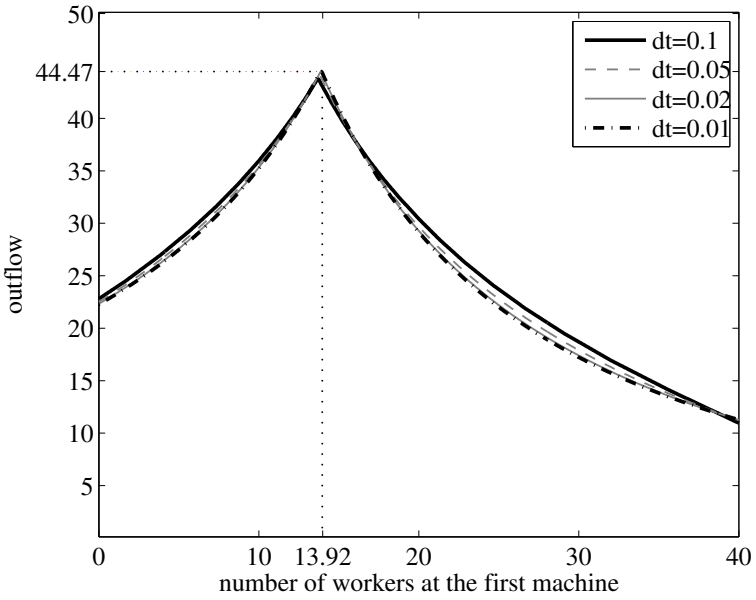


Figure 3.3: Comparison of simulation using different time grid sizes.

Δt	opt. worker distr.	max outflow	final queues	$\sum$
0.1	[13.716, 26.284]	43.756	31.457	75.213
0.05	[13.803, 26.197]	44.163	30.888	75.050
0.02	[13.871, 26.129]	44.453	30.558	75.011
0.01	[13.921, 26.079]	44.471	30.532	75.003

Table 3.1: Verifying conservation of mass. The total inflow is $\sum_{t\in\mathcal{T}} f^t_{ext}\Delta t = 75$.

Worker Changes

In a next step, we illustrate the modelling aspect of worker changes. We have the option, to vary the worker schedule at certain points in time. We allow the workers to change their position once in the middle of the time horizon. We fix the time grid size to $\Delta t = 0.01$ and simulate the objective function value varying β_1^t from 0 to 1 with step width 0.001 β_2 of the second machine automatically varies since $\beta_2^t = 1 - \beta_1^t$. Since β_1^t has two values (one for each time period), we end up with a 3D-plot showing the objective function value for all combinations of $\beta_1^t, t \in [0; 2]$ and $\beta_1^t, t \in (2, 4]$. The result is depicted in Figure 3.4.

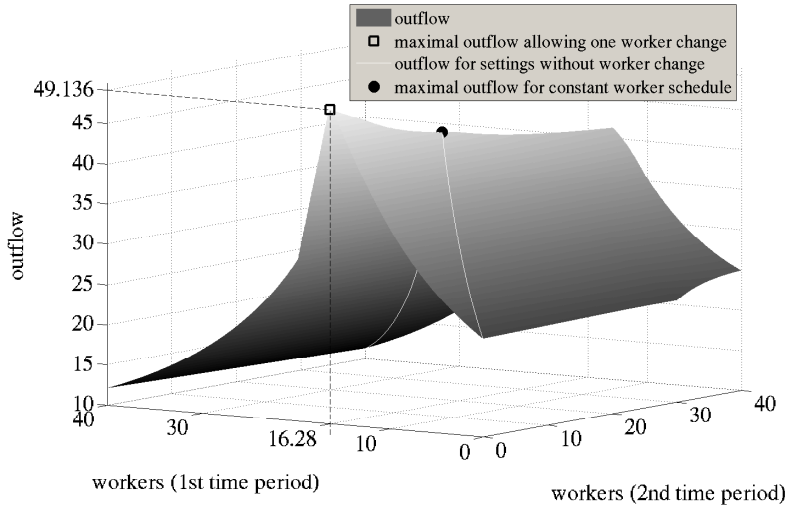
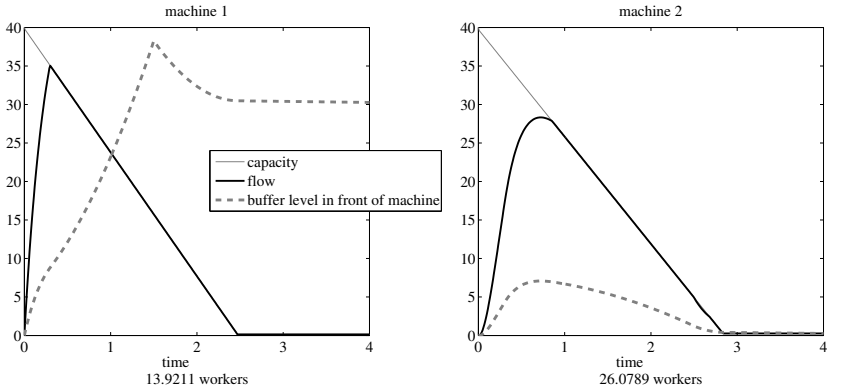


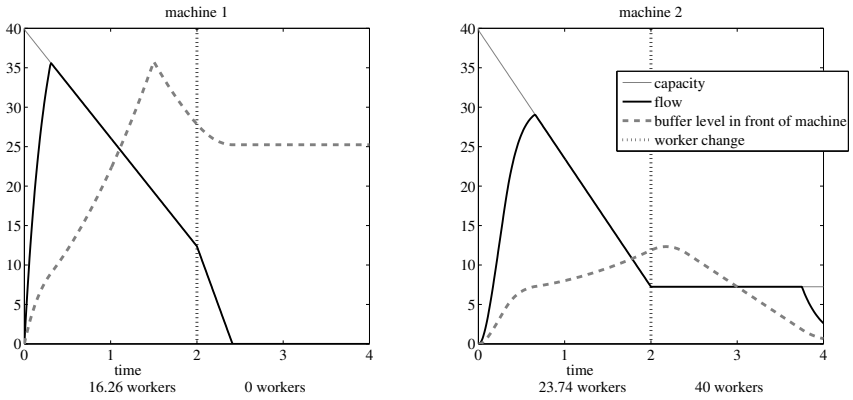
Figure 3.4: Outflow depending on the number of workers at the first machine, including one worker change in the middle of the time horizon.

Obviously, at a first result, allowing one worker change within the time horizon leads to an improvement of the optimal solution (49.14 units compared to 44.47 units). To understand this behavior, we shall have a closer look at the evolution of flow, capacities and buffer levels as well, which are plotted in Figure 3.5(a).

In all these plots we observe that the number of workers at a machine influence the slope of the capacity evolution. Unless the capacity is neither 0 nor has reached its maximal level μ, it can be described by a (piecewise) straight line with slope $W\beta_i r_i - l_i$ (cf. equation (2.25d)). Since the breakdown rate of the second machine is 40, we need



(a) Constant worker schedule.



(b) One worker change in the middle of the time horizon.

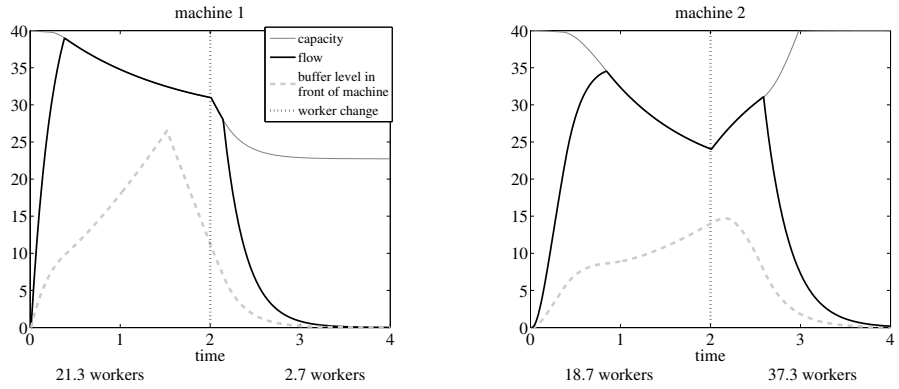
Figure 3.5: Optimal solution for serial processors.

all 40 workers to keep the capacity at the same level. This happens in the time period after the workers have changed, see Figure 3.5(b). In this way, the flow is sustained leading to a larger total outflow value compared to a fixed worker schedule.

Remark 3.1.2. *It is not always the case that a unique maximum is reached. On the contrary, in more complicated settings many local maxima may occur. In such cases the* fmincon *solver is not reliable anymore since it often gets stuck in local optima. Another drawback of* fmincon *is that we cannot stick to schedules with integer workers.*

Flow-dependent Capacity Behaviour

Now, we consider the modified model, using equation (1.23). To compare the qualitative behaviour of both models, we use the same testcase as before. Note, that this time, the capacity loss is proportional to the through-going flow. For that reason, we choose the breakdown parameter l in a way that the magnitude of the capacity loss is roughly comparable to that of the previously discussed setting. Namely, we set $l_1 = 0.75$ and $l_2 = 1$. The results are shown in Figure 3.6.

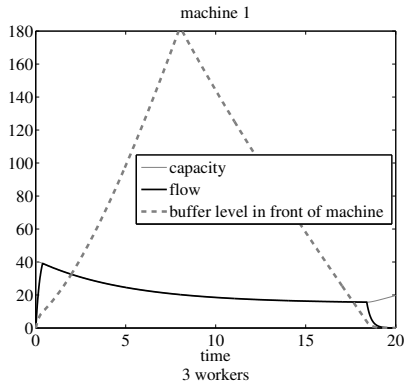


(a) (b)

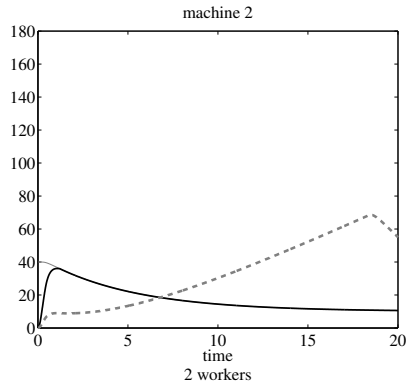
Figure 3.6: Modified model: optimal solution assuming one worker change in the middle of the time horizon.

Usually abrasion effects are not as drastic and only perceptible after a longer time period. For that reason we want to compare the observed behaviour of the model with a more realistic parameter setting. We set the breakdown parameter l to 0.2 for both

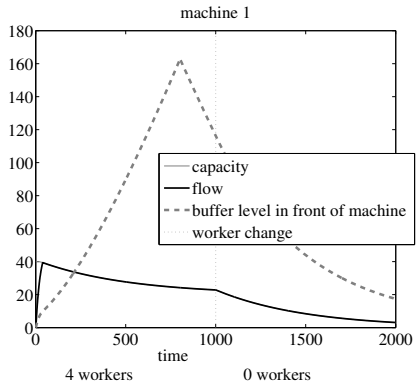
machines. This means that we have a capacity loss of 20% compared to the throughflow. Furthermore, we extend the given time horizon to $T = 20$ and set the external inflow to 50 parts per unit time for the first 8 time units. The available number of workers is $W = 5$. This time, we additionally restrict the number of repair workers to integer numbers, which is done by using Cplex [23]. The results are shown in Figure 3.7.



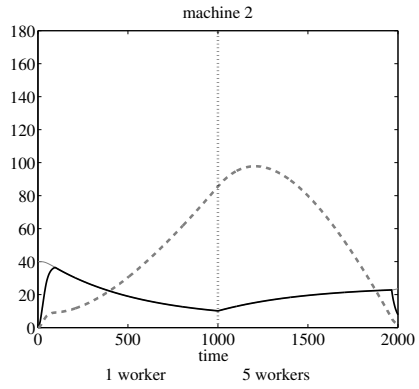
(a) Constant worker schedule, machine 1.



(b) Constant worker schedule, machine 2, total outflow = 345,64 parts.



(c) One worker change within the time horizon, machine 1.



(d) One worker change within the time horizon, machine 2, total outflow = 381.29 parts.

Figure 3.7: Optimal solution - considering a larger time horizon with more subtle breakdown rates.

Again, the worker change leads to an increase of total outflow, which is in this case about 10%.

Note, that the runtime increases drastically, when longer time periods T are chosen. In the previous example where T has been set to 4, the optimisation takes less than 2 seconds, whereas it takes more than 9 minutes in the last testcase with $T = 20$.

3.1.2 Production Networks

So far, the serial processor test case is a nice example to get insight and feeling for the dynamics involved in the repair worker assignment model. After this numerical experiments mainly computed in Matlab, we now have a different focus. First of all, we analyse the steady state problem (2.45) in Subsection 2.3.2 and point out, in which way the obtained information can be exploited for the dynamic model (2.37). The models, formulated as linear MIPs (2.46), are solved by Cplex [23]. We extend our studies to a more general network with 12 edges, as shown in Figure 3.8 and restrict the worker distributions to integer values only due to the easier applicability to real world problems. We allow external inflow for the first two edges and are interested in maximizing the outflow at edges 11 and 12.

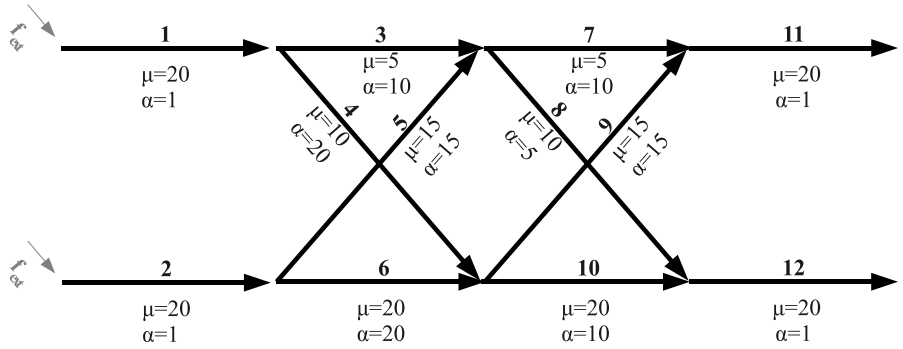


Figure 3.8: Branched network with 12 edges where $r_i = 1$ for all machines.

Steady state studies

Before we prescribe the external inflow and compute the optimal solution of the dynamic model (2.37), we first have a deeper look at the steady state solutions, described in Subsection 2.3.2. Different from the dynamic model, the external inflow of the steady

state model (2.45) is not given a priori, but is maximised simultaneously with the outflow.

Analysis on the amount of available repair workers

From case 1.1 of equation (2.39) with $\epsilon = 0$ we can deduce that we need at least $\frac{l_i}{r_i}$ workers in order to keep the capacity of machine i to its maximal level. In our setting r_i is set to one for all i and breakdown rates l sum up to 109. This means that at least 109 workers are necessary avoid capacity drops. Since employing workers is expensive, it is rewarding to check, how we can cope with less manpower.

The question arises, how many workers we would at least need to get a steady state through-flow greater than zero. In the case that we do not previously fix the flow distribution at the nodes as explained in Subsection 2.3.2, we can find the answer in the following way: Assume that the maximal capacity μ_i is greater than zero for all machines. As explained before, the capacity of a machine can only be sustained, if at least $\frac{l_i}{r_i}$ workers are allocated to it. We can find the least manpower consuming path through the network by using a standard shortest path algorithm such as Djikstra algorithm, for more details see [51] and [66] amongst others. The through-flow of this path is bounded by its bottleneck, which is the machine with the smallest capacity. For our testcase, we need at least 17 workers contributed along the shortest path to get a steady state solution greater than zero. In this case, the through-flow is 5 parts per time unit, see Figure 3.9.

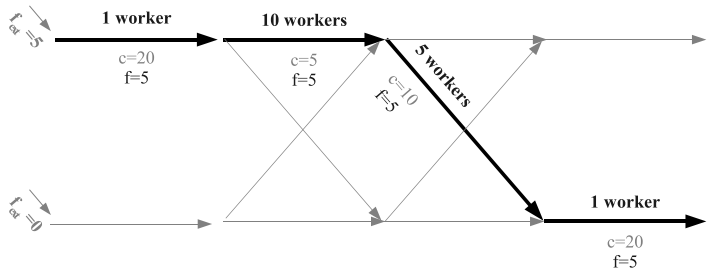


Figure 3.9: The least manpower consuming steady state solution greater than zero. The resulting through-flow is 5 parts per unit time requiring a minimum of 17 workers.

When we previously fix the distribution behaviour of the flow as in Subsection

2.3.2, for example to equal distribution between the succeeding edges, we need a lot more workers to get a positive through-flow. This is due to the fact that once an edge transmits a flow, all its succeeding edges must also have a capacity greater than zero, such that the flow can be distributed in the prescribed way. Note, that in steady state, solutions do not allow for increasing buffers. For our testcase, we need at least 73 workers to get a positive steady state flow. The resulting through-flow is 10 parts per unit time. For details, see Figure 3.10.

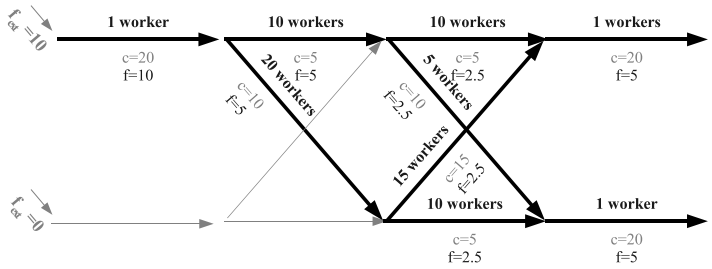


Figure 3.10: The least manpower consuming steady state solution greater than zero, for equally distributed flow at branching nodes. The resulting through-flow is 10 parts per unit time and the necessary number of workers is $W = 73$.

Moreover, it is interesting to compute the maximal steady state solution, when we have no capacity drop. If the flow distribution at branching nodes is not previously fixed, we can find the solution via the Ford-Fulkerson-Algorithm [6] using the maximal capacities $c_i = \mu_i$ as upper bounds. The result is shown in Figure 3.11.

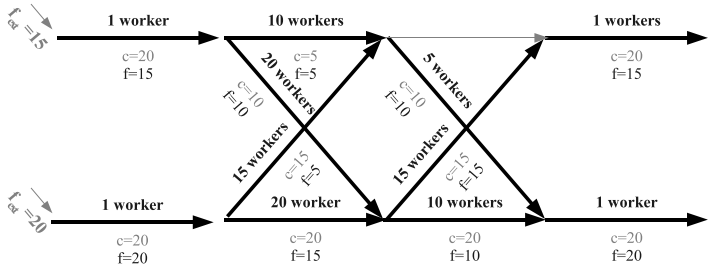


Figure 3.11: The maximal static through-flow when all capacities are at their maximal level.

This gives us an upper bound for the maximal through-flow. In our case, it is 35 parts per time unit.

Under-staffed settings

From the above analysis we know that finding the optimal worker distribution is only interesting in the case that we have less than 109 workers available. Otherwise, we can always distribute the workers in a way that no capacity loss occurs.

In the following we consider two scenarios where the total number of workers is set to 30 ($\rightarrow$ highly under-staffed) and to 100 ($\rightarrow$ slightly under-staffed) respectively.

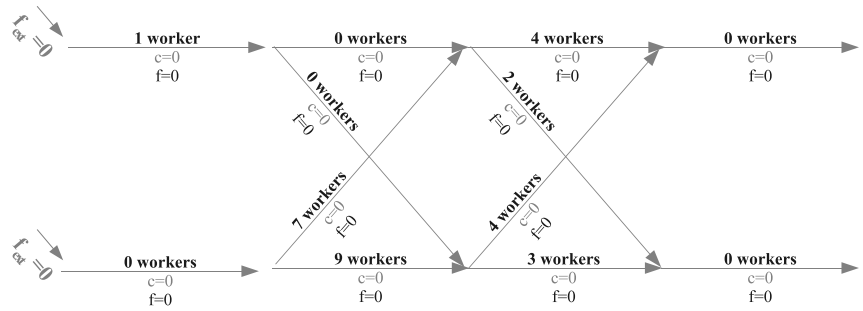
Moreover, we compare both versions of the steady state optimisation problem (2.45): First we use the flow distribution matrix d and thus a fixed flow distribution at branching nodes, and for the second run we exchange d with the incidence matrix K, see (2.48), leading to variable flow distributions that are subject to the optimisation process.

The resulting maximal through-flow of the different settings is depicted in Figure 3.12 for 30 workers and in Figure 3.13 for 100 workers.

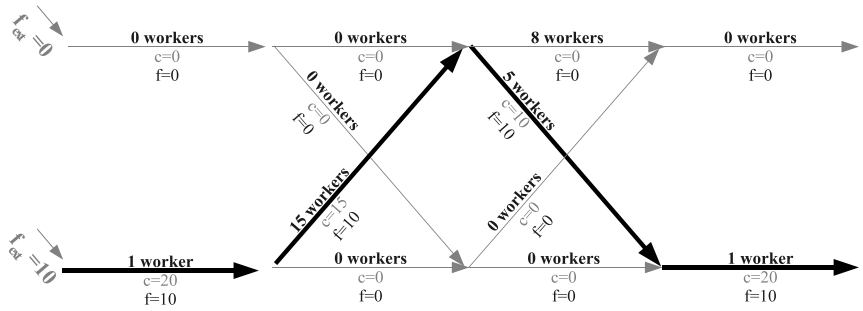
In the highly under-staffed scenario with fixed flow distribution (see Figure 3.12(a)), it is not possible to allocate the workers in a way to obtain positive solution. All machines are out of order and no flow is able to go through. However, if we leave the distribution of flow up to optimisation, we can find a solution where a through-flow of 10 parts per time unit can be provided, on the only functioning path through the network (see Figure 3.12(b)).

As expected, we get a much better solution, when we increase the number of workers to 100 (see Figure 3.13). Now, the setting with fixed flow distribution allows a maximal through-flow of 20 (see Figure 3.13(a)), whereas the additional optimisation of the flow distribution increases the through-flow to 35 (see Figure 3.13(b)). As shown above, this is already the upper bound of steady through-flow with respect to the number of repair workers.

The steady state solutions can be useful for the dynamic model (2.37). As explained in the sequel, the steady state analysis provides us with a qualitatively good start solution for the dynamic MIP (2.37), leading to significant runtime reductions of the optimisation procedure. Furthermore, we can observe that optimisation of the flow distribution at branching nodes leads to a considerable gain of outflow. This does not

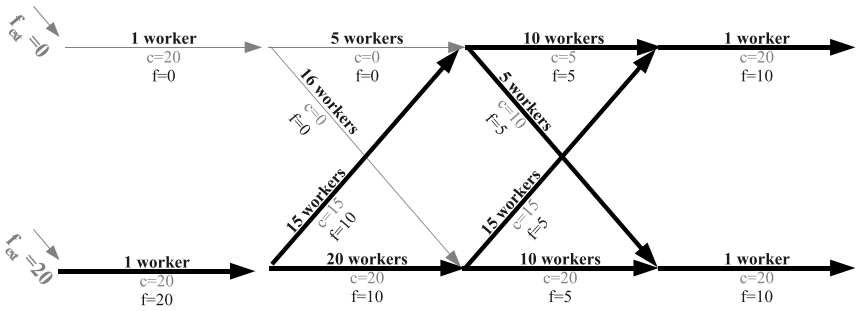


(a) Fixed flow distribution.

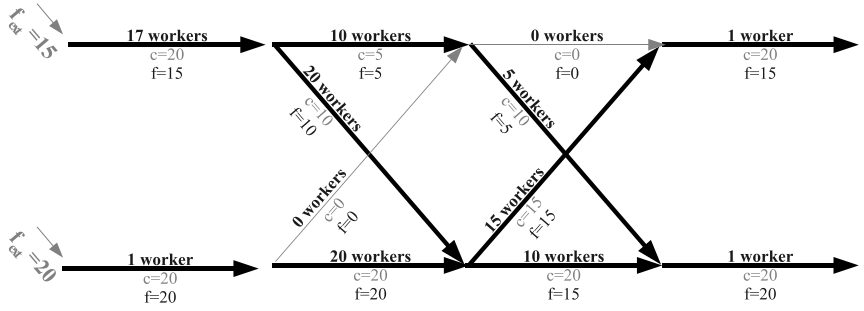


(b) Optimised flow distribution.

Figure 3.12: Maximal through-flow, scenario with 30 workers.



(a) Fixed flow distribution.



(b) Optimised flow distribution.

Figure 3.13: Maximal through-flow, scenario with 100 workers.

only hold for the steady state case but also for the dynamic setting, as described in the sequel.

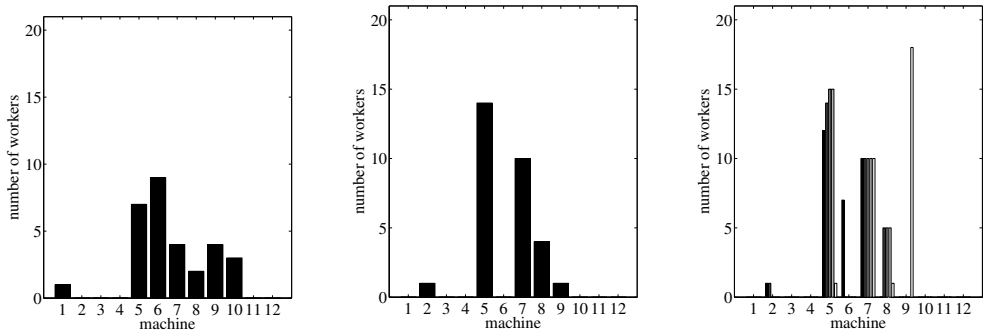
Dynamic repair model

Now, we move on to the dynamic repair model (2.37). As before, we use the flow distribution matrix d that divides the flow in equal shares among the succeeding edges at branching nodes.

Different to the steady state model, we have to fix the external inflow function in the dynamic setting. We choose $f_{ext} \equiv 20$ for edge 1 as well as for edge 2. The time horizon T is set to 5 and the time grid size to $\Delta t = 0.1$. As initial conditions, the network is empty, i.e. buffers and flows are equal to zero for $t = 0$ and the capacities are set to its maximal values $c_i^0 = \mu_i$. As in the previous subsection, we again consider the highly under-staffed setting with 30 workers as well as the slightly under-staffed one where 100 repair workers are available.

Due to the high complexity of the dynamic problem (2.37) it is advisable to provide a start solution in order to speed up computation time. A feasible start solution can easily be computed by fixing the worker assignment for all machines and computing the forward solutions for the capacity and buffer conditions according to (2.26b) and (2.26a). This procedure is explained in Subsection 2.2.4, Figure 2.5. The overall outflow after the time horizon is 10.27 parts, when 30 workers are equally distributed among the machines. Using this setting as start solution, optimisation takes 504.55 seconds. However, if we use the optimal solution for the steady state case, depicted in Figure 3.14(a), the resulting outflow is 16.52 parts. When we use this solution as a start for optimisation, the computation time reduces to 316.08 seconds, see Table 3.2. The optimal worker distribution is shown in Figure 3.14(b) and leads to an outflow of 41.88 parts. If we allow position changes of the repair workers after each time unit, the optimisation time strongly increases to more than 3 days. The optimal solution is to assign many workers towards the end of the network in the last time period (see Figure 3.14(c), machine 9). This leads to an augmentation of outflow in the final time period and to an overall outflow of 42.73 parts.

The outflow behaviour throughout the time horizon is depicted in Figure 3.16(a) for different worker distributions.



(a) Optimal worker distribution of the steady state case.

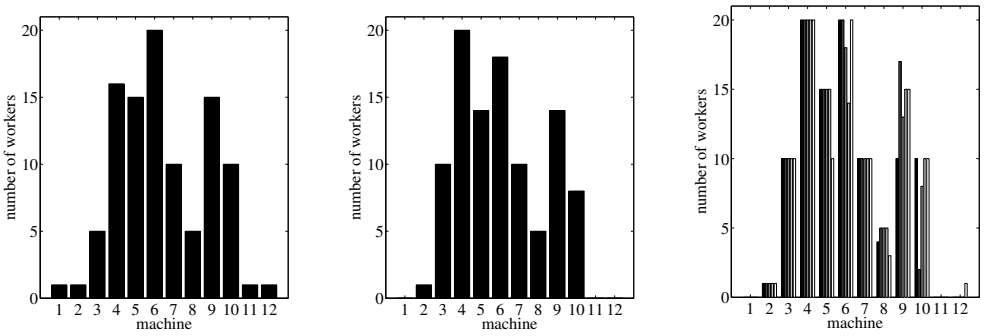
(b) Optimal solution for the dynamic model.

(c) Optimal solution for the dynamic model allowing the workers to change position after each time unit.

Figure 3.14: Optimal worker distribution for the highly under-staffed setting, i.e. $W = 30$.

The same investigation has been done for the case that 100 repair workers are available, see Figure 3.15 and 3.16(b). Again, optimisation time can highly be reduced by using the steady state optimal solution, leading to a run time of 569.53 seconds, which is a third of the runtime, when the start solution is given by equally distributed workers (namely 1794.47 seconds). However, the computation time is unexpectedly short, when we allow position changes of the workers, only 154.74 seconds, see Table 3.2. An explanation for this phenomenon gives the comparison of the worker distribution shown in Figure 3.15. It is conspicuous that the optimal worker distribution of the steady state model, shown in Figure 3.15(a) is already quite similar to the optimal solution with and without worker changes, see Figures 3.15(b) and 3.15(c).

For the modified model where the breakdown rate is proportional to the throughflow the machines, the behaviour is similar. We use the same branched network as before with breakdown parameters given by $l = [0.1\ 0.1\ 0.5\ 1\ 1\ 1\ 1\ 0.2\ 0.75\ 0.5\ 0.1\ 0.1]$. The optimal worker distribution with and without worker changes is shown in Figure 3.17. Here, the following output can be achieved for the different scenarios (a)-(c):
(a) outflow of steady state optimal startsolution: 83.211 parts.
(b) outflow of optimal solution without worker change: 112.233 parts.
(c) outflow of optimal solution with worker change: 115.208 parts.

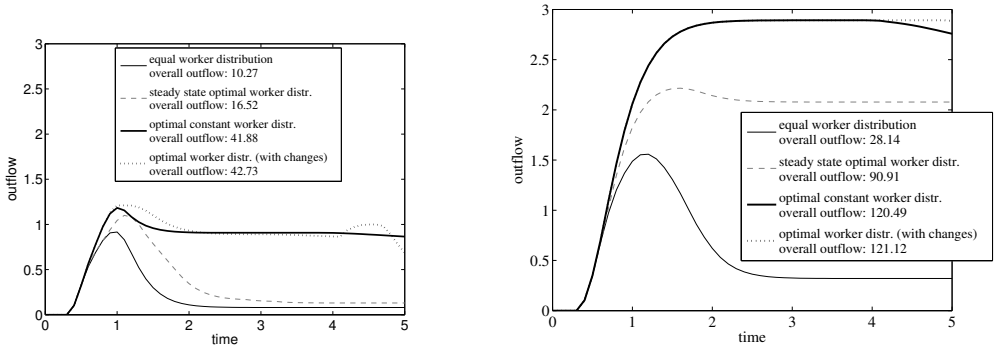


(a) Optimal worker distribution of the steady state case.

(b) Optimal solution for the dynamic model

(c) Optimal solution for the dynamic model allowing the workers to change position after each time unit.

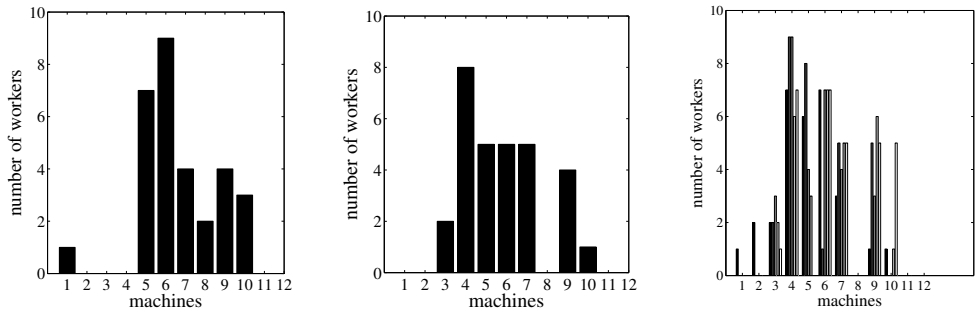
Figure 3.15: Optimal worker distribution for the slightly under-staffed setting, i.e. $W = 100$.



(a)

(b)

Figure 3.16: Outthrough-flowout the time horizon for the slightly under-staffed setting comparing different worker assignments.



(a) Optimal worker distribution for the steady state case.

(b) Optimal solution for the dynamic model.

(c) Optimal solution for the dynamic model allowing the workers to change position after each time unit.

Figure 3.17: Modified model using 30 repair workers.

In Table 3.2 the computation times of the different optimisation runs are listed. Due to the smaller number of binary variables in the modified model, the complexity of the corresponding MIP is smaller. For that reason much less computation time is needed.

# workers	worker changes	start solution	original model:	modified model:
30	no	equal distr.	504.55 s	65.69 s
30	no	steady state opt.	316.08 s	18.57 s
30	yes	equal distr.	$>$ 3 days	231.71 s
30	yes	steady state opt.	$>$ 3 days	411.27 s
100	no	equal distr.	1794.47 s	8.14 s
100	no	steady state opt.	569.53 s	18.62 s
100	yes	equal distr.	65.57 h	23 h
100	yes	steady state opt.	154.74 s	46.71 s

Table 3.2: Optimization time comparison.

Changing the flow distribution at branching nodes

In the sequel, we will use an important observation concerning the previously described steady state analysis. Remember that the steady through-flow can be significantly improved, when the flow distribution at branching nodes is not a-priori fixed. A straightforward idea would be to include this flexibility as well into the dynamic model (2.37) by exchanging the flow distribution matrix d by the incidence matrix K analogously as done for the steady case in Subsection 2.3.2. However, this ansatz encloses a significant drawback. The distribution rates of the flow do not appear explicitly as parameters in the formulation of the problem. For that reason it is not possible to restrict to constant distribution rates, when the incidence matrix is used. Consequently, we can not avoid the undesired effect that solutions contain highly fluctuating flow distributions. We prefer to track another idea. We use the optimised flow distribution of the steady state case for the dynamic model (2.37) by adapting matrix d accordingly.

Step 1: Compute the steady state solution with variable flow distribution (2.48).

Step 2: Construct the flow distribution matrix d according to the distribution of the steady state obtained in step 1.

Step 3: Solve the dynamic repair model (2.37) using d and taking the optimal worker distribution of step 1 as start solution.

In Table 3.3 the corresponding optimisation results are listed.

# workers	changes allowed?	optimisation time	opt. gap	outflow of. start sol.	optimal outflow	improvement to previous flow distr.
30	no	3207.46 s	0 %	42.56	45.05	7.56 %
	yes	$>$ 3 days	2.33 %	42.56	$\in [49.56, 50.71]$	$>$ 15.98 %
100	no	0.84 s	0 %	126.28	126.28	4.80 %
	yes	0.84 s	0 %	126.28	126.28	4.26 %

Table 3.3: Optimization results for the dynamic repair problem using optimal flow distribution rates of the steady state analysis.

The last column of Table 3.3 shows the considerable gain of outflow by using the optimised matrix d instead of equal flow distribution. Furthermore, it is interesting to

have a look at the computation time. For the highly under-staffed setting optimisation takes notedly longer. When allowing worker changes, the optimality gap of the algorithm could not even be completely closed after three days. On the other hand, the gain of outflow is noteworthy, especially when workers are allowed to change their position after each time unit. When 100 workers are available, the optimal steady state solution turns out to be already optimal for the dynamic model, even for the case in which we allow worker changes. Hence, the optimisation time is with 0.84 seconds extremely short.

A comparison between the obtained outflow using different settings is illustrated in Figure 3.18.

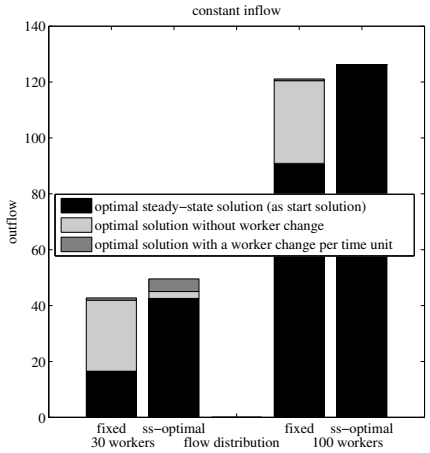


Figure 3.18: Comparison of outflow of different settings.

The two bars on the left show the total outflow, when 30 repair workers are available. The black part indicates how much outflow is obtained by using the worker distribution which is optimal for the steady state model, the light gray part shows the gain of outflow when we us the optimal worker distribution and the dark gray part shows the increment of outflow, when workers are allowed to change their position after each time unit. The bars on the right show the same results for 100 repair workers.

It is remarkable that the optimal steady state worker distribution is already really close to the optimal solution of the dynamic model in the case that we use the flow

distribution that is optimal for the steady state case (in the figure denoted by "ss-optimal").

Summarizing the numerical observations, we can underline the benefit of the steady state analysis. Note that the steady state problem (2.45) is much faster solvable than the far more complex dynamic MIP (2.37) where we need the whole set of variables for each single time step. First of all, the steady state analysis provides us with a qualitatively good starting solution that leads to significant runtime reductions for the optimisation of the dynamic model. Secondly, the additional optimisation of the flow distribution in the steady state case, endows us with valuable information how to increase the outflow of the dynamic model, given that the flow distribution of the corresponding application is adaptable accordingly.

3.1.3 Real World Example: Toothbrushfactory

In this section, we model a stylised real world example for a toothbrush factory considered in [42, 47]. It consists of 12 production units, sketched in Figure 3.19. The production steps are represented by edges of a graph depicted in Figure 3.20. All computations are applied to the modified capacity model, i.e. problem (2.26) where equation (2.26b) is replaced by (2.27).

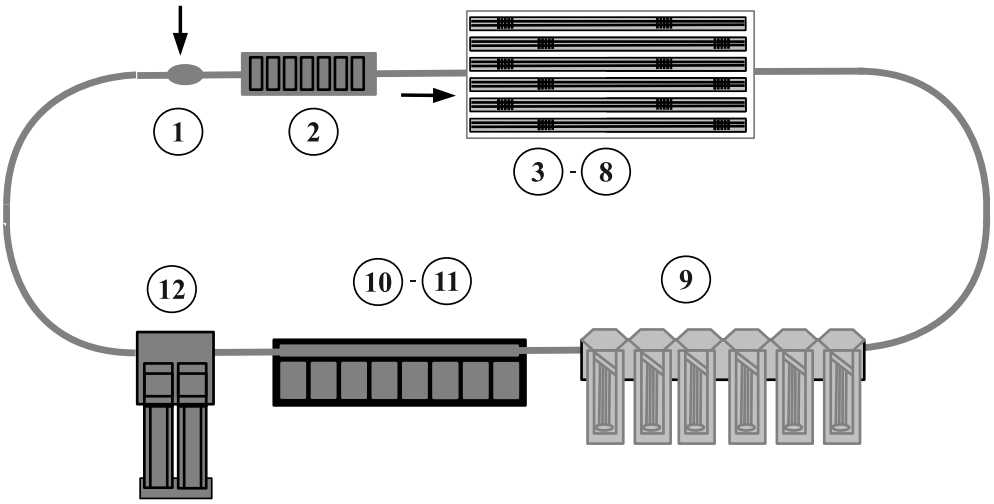


Figure 3.19: Layout of the toothbrushfactory.

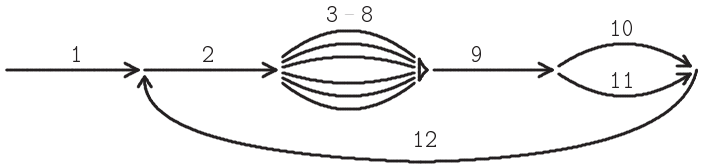


Figure 3.20: Schematic draft of the production steps.

In edge 1 empty pallets are introduced into the system. At edge 2 containers for the toothbrushes are mounted upon the pallets. The actual production of toothbrushes are processed in edges 3 to 9. Up to edge 9, workers are primarily needed to restock production material and to replace worn-out tools. At edge 10 and 11 production workers check the quality of the finished products and sort deficient toothbrushes out. Here, the through-flow depends proportionally on the number of workers, in contrast to the other production steps where primarily machines in operation. To include 10-11, we replace equation (10) by

$$c_i(t) = \min\{\mu_i,\, r_i W \beta_i(t)\}, \text{ for } i \in \{10, 11\}$$

and linearise it as explained in Subsection 2.2.2. For the other production steps we use the flow dependent capacity model (1.23).

In 12 the finished products leave the factory (not shown) and the empty pallets enter the system again at 2. The parameters are listed in Table 3.4.

Edge	Production step	$\mu\,[\frac{\text{parts}}{\text{minute}}]$	τ [minute]	l	r	# workers
1	intake	100	1	0	1	0
2	assembly of pallets	42.6	1	0.05	5.325	1
3-8	thermoforming and transport (parallel)	4	1.25	0.0083	0.1333	2 (altogether)
9	assembly line	42.6	1.05	0.1	4.26	1
10 -11	sorting of deficient items (parallel)	14.4	1	0	6	6 (altogether)
12	emptying pallets	42.6	1	0.01667	1.42	1

Table 3.4: Parameter setting of the toothbrush-factory, scaled to minutes as time unit.

We consider the following setting: In the beginning, the network is empty. In the first 10 minutes 81.1 pallets per minute are introduced into the system at unit 1.

Under standard conditions 11 workers are needed for a stable production, seen in the last column of Table 3.4. In Figure 3.21 the flow behaviour in the first hour of the daily production process is shown. This time is needed to raise the production flow inside the initially empty network until constant cycle of pallets is obtained. On the left we see the inflow of pallets into the system at unit 1. At processor 2 the pallets enter the production cycle. The processor works at full capacity for the first 40 minutes until all incoming pallets from edge 1 are processed. Afterwards the flow reduces to a constant rate. The figure on the right shows processor 12 where finished toothbrushes are taken out. After the first hour a total of 1212 pallets reach processor 12.

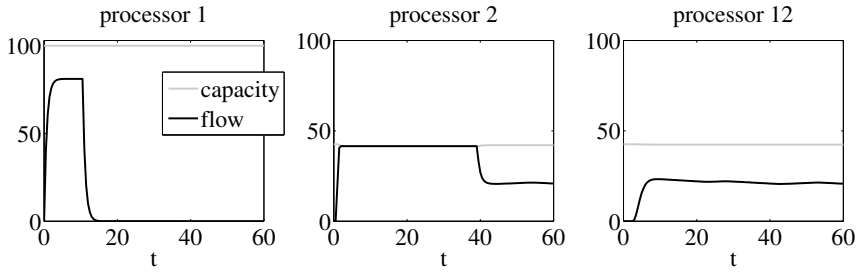


Figure 3.21: Production process for the first hour using standard worker allocation. Obtained output after the first hour: 1211 pallet-loads.

Now, we imagine the following scenario: 4 workers are not available within the first hour. These workers are usually assigned to production units 3, 9, 10 and 11. Without optimisation, the production capacity of the abandoned units would soon decrease. In this example the drastic decrease of capacity at production unit 9 leads to a total decrease of production flow as depicted in Figure 3.22. The output after one hour reduces to 424 pallet-loads.

If we assume that the remaining workers are able to fulfill the tasks of the missing workers as well, they can support the production at the abandoned machines. It is reasonable to assume that they can easily change their position every 20 minutes in order fix capacity losses. We optimise using the previously derived DTN-MIP, (2.37). As result we get an optimal solution for the worker assignment as in Figure 3.23. Due to the lack of workers the production capacity at some machines, as for example at

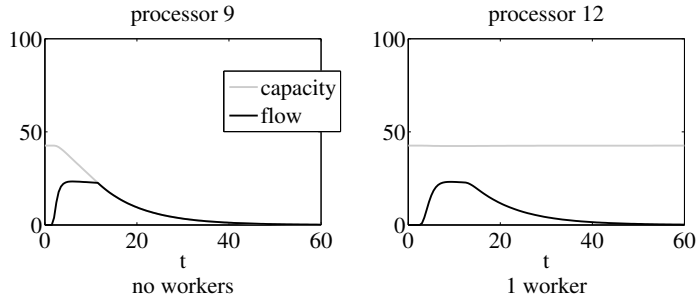


Figure 3.22: Production process with 4 missing workers. The total ouflow reduces to 424 pallet-loads.

processor 12, is reduced. However, the output until then is 1210 pallet loads and thus almost as good as if all processors would have been fully manned.

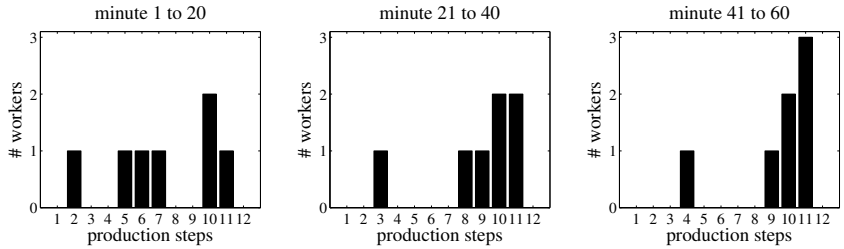


Figure 3.23: Optimal worker assignment using 7 workers.

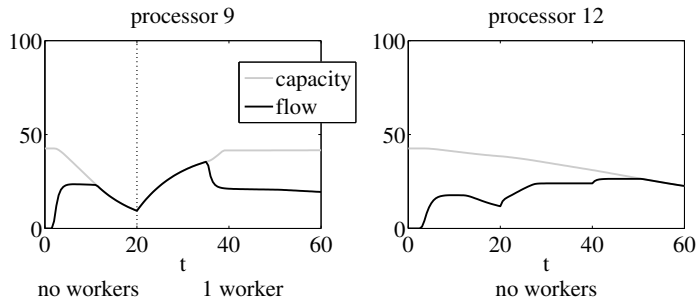


Figure 3.24: Flow of the optimal solution at production unit 9 and 12.

3.2 Application II: Traffic Networks - Optimal Traffic Light Setting

In this section, we consider traffic flow networks. First of all, we model different kind of roundabouts and simulate various scenarios using the Hamilton-Jacobi-Scheme derived in Subsection 1.4.2, see Algorithm 1. Secondly, we consider crossovers with traffic lights and optimise traffic light settings using corresponding DTN-MIPs, see Chapter 2 (2.98). In the end, we will investigate the efficiency of the optimisation procedure and the impact of tuning techniques as described in Subsection 2.2.4.

3.2.1 Simulation of a Roundabout, applying the Hamilton-Jacobi Scheme

Now, we apply Algorithm 1 to several traffic flow situations. We test the introduced simulation method against the Godunov Scheme (cf. Subsection 1.4.1) and describe certain effects.

First of all, we apply the merging and dispersing junction model to a small network consisting of eight roads, see Figure 3.25. The network describes a small traffic circle that has already been examined in [11]. We use the same instance as in [11] where the flow is given by $f(\rho) = \rho(1-\rho)$ and initial as well as boundary data a given as follows:

boundary density of incoming roads:	$\rho_1(x,0) = 0.25,\ \rho_3(x,0) = 0.4$
initial density of incoming roads:	$\rho_1(0,t) = 0.25,\ \rho_3(0,t) = 0.4$
initial density of outgoing roads:	$\rho_2(0,t) = \rho_4(0,t) = 0.5$
initial density of inner circle:	$\rho_i(0,t) = 0.5,\ \forall i = 5,6,7,8$

In [11] this test case is compared for different right-of-way parameters $q \in]0,1[$, determining the proportion of cars coming from each road at merging junctions. The priority rule used in this paper corresponds to $q = 0$.

The graphic of Figure 3.25 shows the traffic density exemplarily for four roads at 4 different points in time. Since the boundary condition is constant, the density evolution reaches an equilibrium and does not change for $t > 5$. The traffic at the inner circle has priority, therefore the roundabout does not get blocked. This is qualitatively the same behaviour as in [11], when a small parameter q is used. Since our model uses strict priorities, the equilibrium state is reached faster, than in [11].

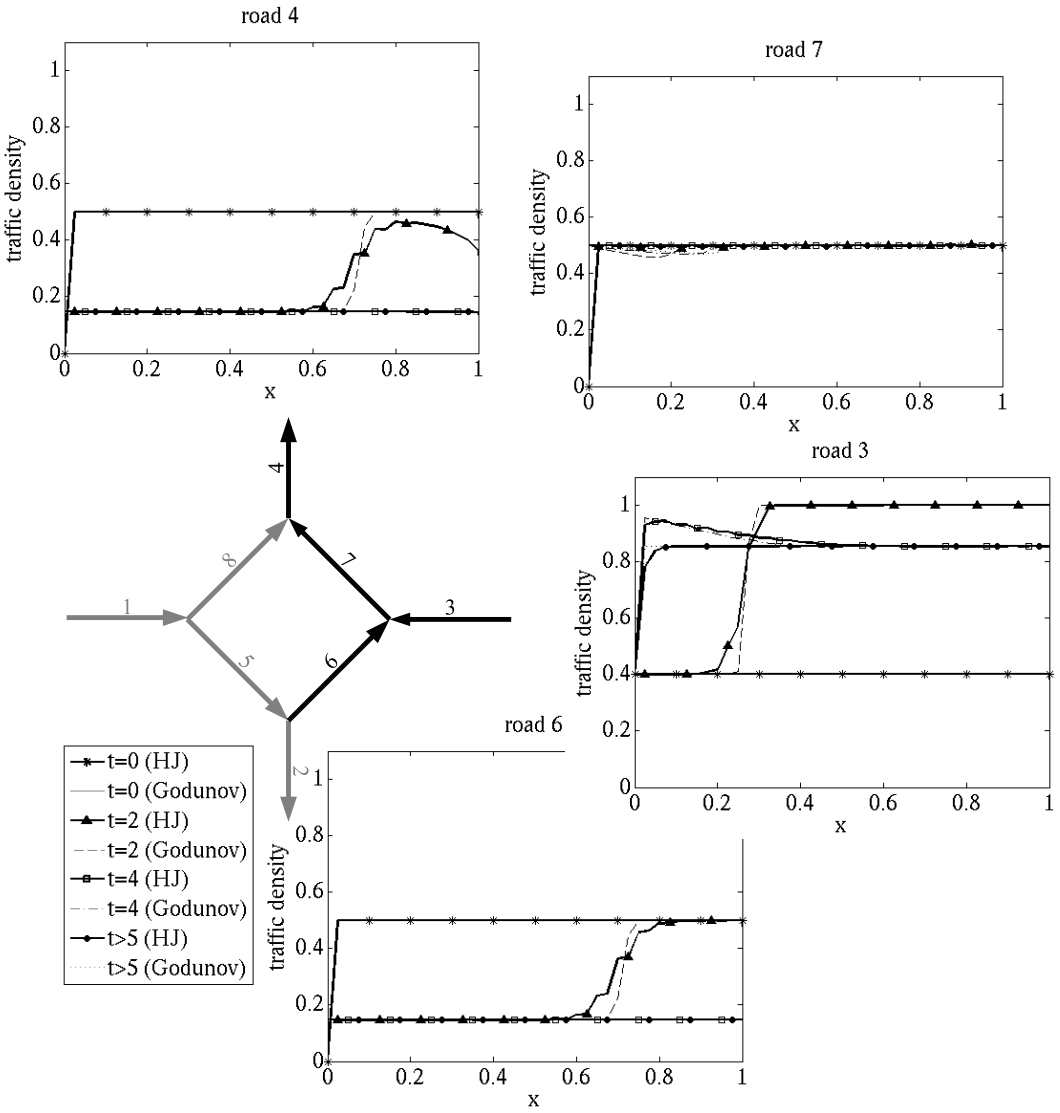
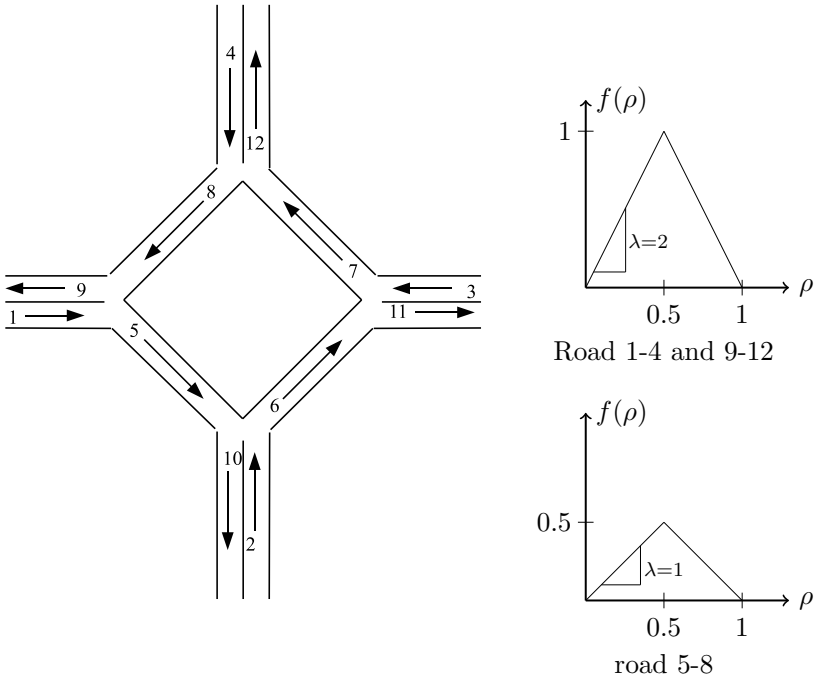


Figure 3.25: Small roundabout. Results for the LWR flow function $f(\rho) = \rho(1-\rho)$ on each road with priority rules at merging junctions.

We use the Hamilton-Jacobi Scheme for simulations and reconstruct the density values as in (1.72). The thin lines show the result obtained by using the Godunov Scheme, which we use as a benchmark. For road 4 at time $t = 2$ it can clearly be seen that a much sharper shock wave is obtained by Godunov. However, the actual density levels are equivalent for both schemes. Since this model is especially derived for instances with piecewise constant initial conditions, the Hamilton-Jacobi Scheme leads to sufficiently precise results.

A more realistic Roundabout

We consider a roundabout composed of four junctions with two incoming and two outgoing roads as derived earlier, which is depicted in Figure 3.26(a).



(a) Roundabout.

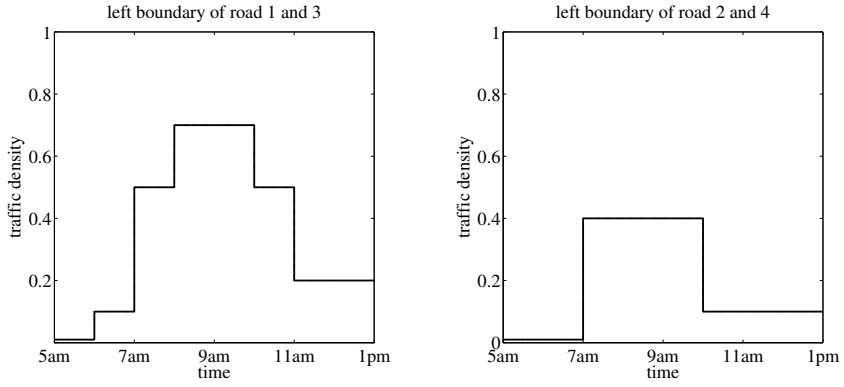
(b) Flow functions for outer and inner roads of the roundabout.

Figure 3.26: Model of a roundabout with different flow functions on different roads.

According to the enumeration in Figure 3.26(a), roads 1 to 4 are leading towards the

inner circle which is composed of roads 5 to 8. Roads 9-12 point out of the roundabout. Usually, drivers cannot drive as fast inside the inner circle as at the mostly broader roads leading in and out of the circle. For this reason, we describe these roads by different triangular flow functions, as depicted in Figure 3.26(b). As before, the traffic density lives in an interval between 0 (no traffic) and 1 (maximal dense traffic). Since we assume that the usual speed of the cars is faster at the outer roads than inside the circle, the corresponding flow function has a steeper slope outside the inner circle.

We prescribe the left boundary data for the incoming roads 1-4. We assume that road 1 and 3 are slightly more busy than roads 2 and 4. For simplicity we use the same boundary data for each road pair. Figure 3.27 gives a detailed overview of the boundary data at an average working day from 5am to 1pm. This is a fictive test setting attempting to tackle the qualitative traffic behaviour taking the morning rush hour into account.



(a) Boundary density of road 1 and 3. (b) Boundary density of road 2 and 4.

Figure 3.27: Incoming traffic data over time.

Figure 3.28 shows the traffic density along the inner circle for exemplary points in time. Since the traffic at the inner roads always has the priority at junctions and outgoing roads are not blocked in our setting, no jams appear inside the roundabout. But you can observe that at the peak time of the rush hour, the traffic density all along the inner circle is at value $\rho^* = 0.5$, which means that the traffic moves with the maximal possible flow.

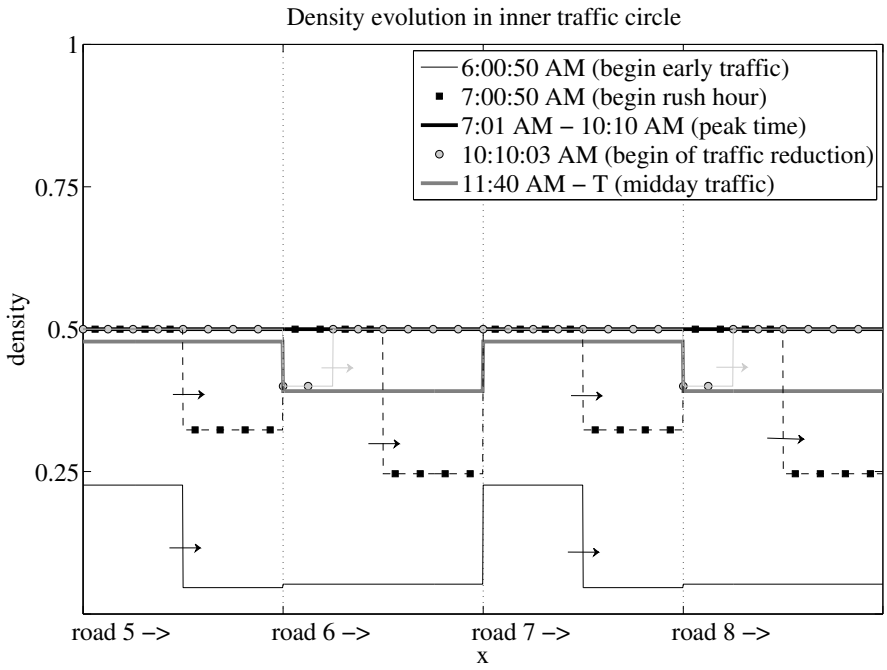


Figure 3.28: Evolution of the traffic density in the inner circle.

However, if we have a closer look at the traffic evolution at a junction, see Figure 3.29, we notice that at the peak time, traffic jams occur at roads leading to the inner circle. Particularly from 7am to shortly after 11am, the traffic entering the roundabout is quite dense. However, since the incoming traffic reduces drastically around 11am (see boundary condition depicted at Figure 3.27(a)) the jam is resolved again a while after the incoming traffic reduces.

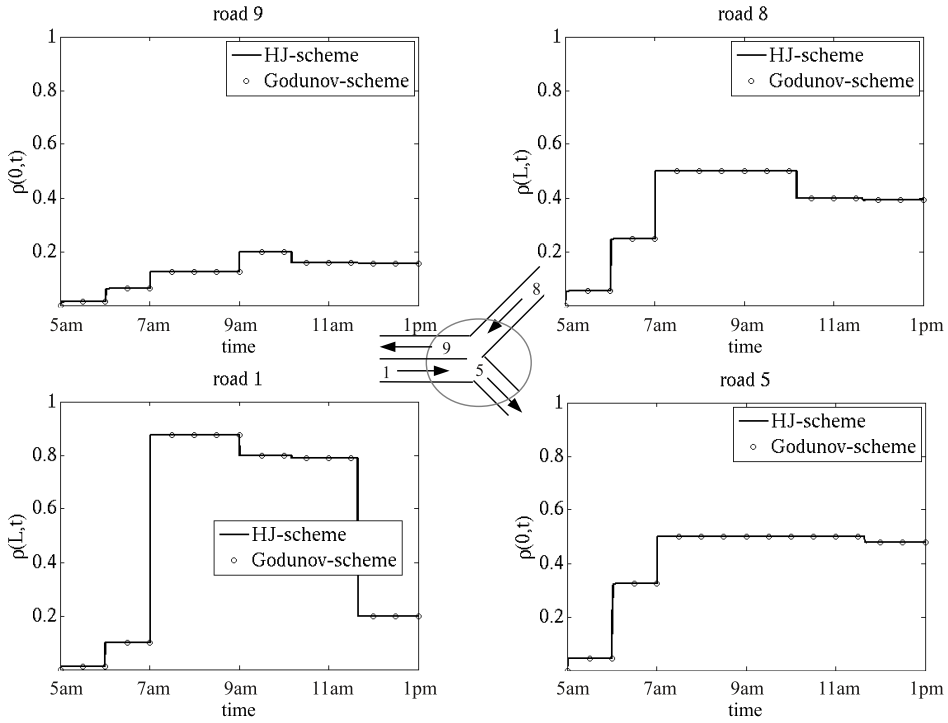


Figure 3.29: Traffic evolution at the junction.

When we compare the Hamilton-Jacobi Scheme with the Godunov Scheme, we observe that for a triangular flow function the results are really precise compared to the use of the functions in Subsection 3.2.1. Thanks to the parameter setting as proposed in Lemma 1.4.2, the shock fronts computed by the Hamilton-Jacobi Scheme are sharp. Furthermore, the trajectories of the fronts are very close to the Godunov solution, due to the artificial shortening of the road which balances out the time delay caused by the use of ghost-cells (compare Remark 1.4.3).

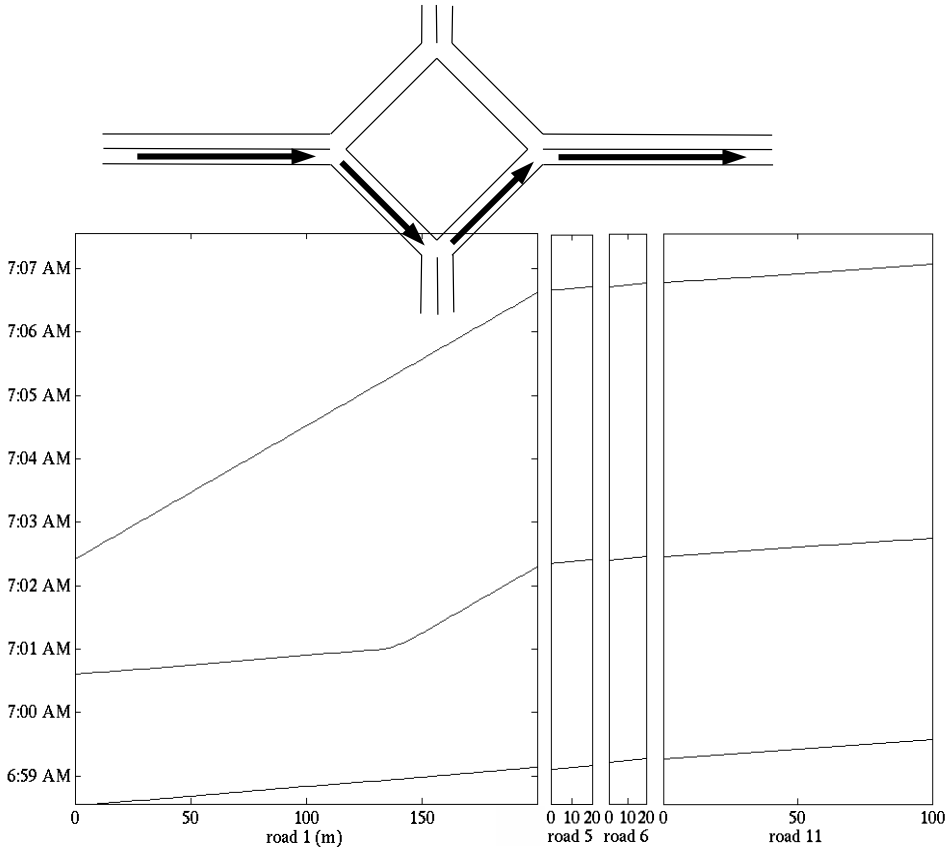


Figure 3.30: Single car tracking for three cars on the above route starting at different times.

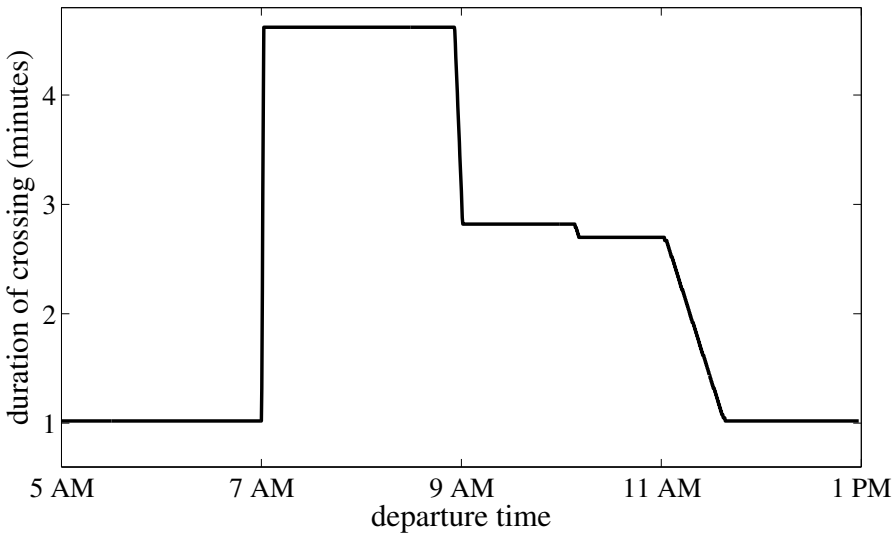


Figure 3.31: Travel time (in minutes) for the route depicted in Figure 3.30, depending on the time when starting the journey.

As we stated in Remark 1.3.10 it is easy to derive the trajectories of cars from the Hamilton-Jacobi formulation, since we only have to track the contour lines of function M. In Figure 3.30, the trajectories of 3 cars moving along the roads 1-5-6-11 are depicted exemplarily. In this example you can see that somebody entering the system before 6:59 am moves freely and leaves the system already about 1 minute later. In contrast to that, another driver, who enters the system only 4 minutes later, already encounters dense traffic on the road and needs more than 4 minutes to move to the end of road 11. The graphic on Figure 3.31 shows the duration of the route 1-5-6-11 depending on the starting time of the journey. While it takes only 1 minute to traverse the route during light traffic times, cars need up to 4.7 minutes between 7 and 9 am. Hence, it takes more than 4 times longer to traverse the given route during the rush-hour.

3.2.2 Traffic Light Optimization

In this section we consider several scenarios including traffic light junctions and use the techniques derived in Chapter 2 to create corresponding DTN-MIPs (2.98). These

problems will be optimised by Cplex [23] in order to obtain optimal traffic light settings. We compare default traffic light settings with optimal solutions and discuss the necessity of additional requirements on switching times, as introduced in Section 2.4.5. Furthermore, we have a deeper look into the optimisation process itself and consider the effects of starting and bounding heuristics for the optimisation time.

Optimal Traffic Light Setting of Crossover

We analyse a crossover as depicted in Figure 3.32. First, we simulate the traffic evolutions for a default traffic light setting. Then we compute the optimal traffic light setting and compare the resulting solutions with and without additional restrictions on the switching time.

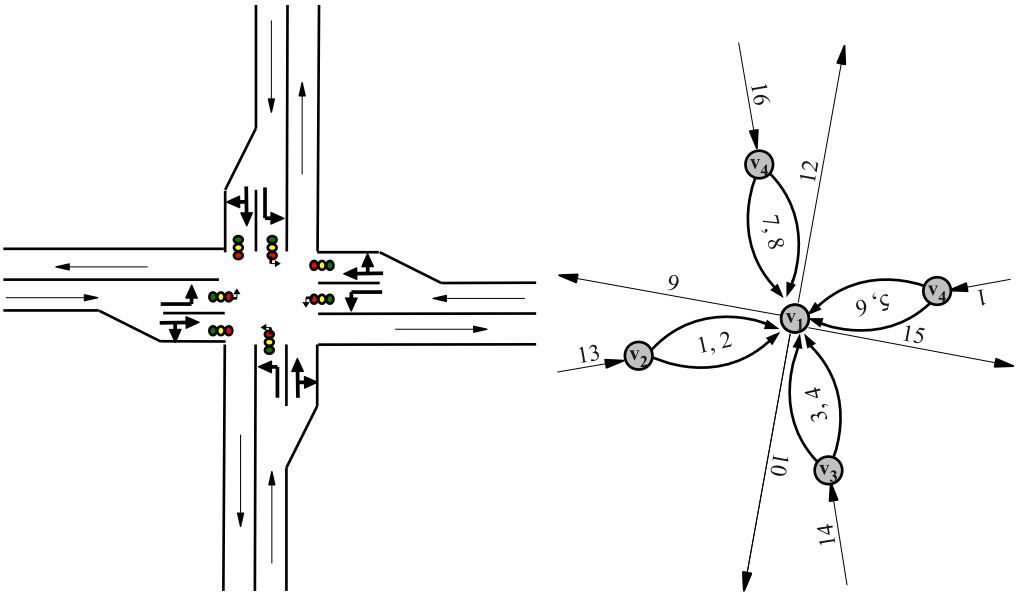
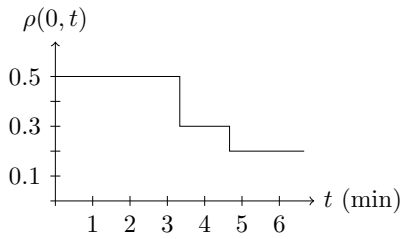


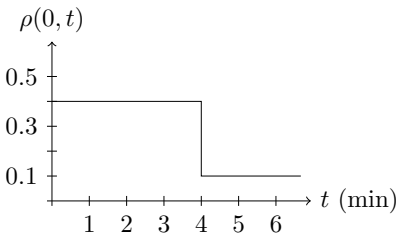
Figure 3.32: Traffic Crossover Each lane for different turning directions is modelled by a separate edge.

Parameter setting and boundary conditions are set according to Figure 3.33 and 3.34.

The default traffic light setting has always green light for all pairs of opposite



(a) Left boundary density of road 13 and road 15.



(b) Left boundary density of road 14 and road 16.

Figure 3.33: Boundary density of incoming roads $i \in E^{in}$.

straight-and-right-turning lanes, as well as for all pairs of opposite left-turning lanes, see Figure 3.35(a). The resulting objective function with the given boundary data is 66.84. To get a feeling on the traffic behaviour in scenarios where traffic lights are used, we refer to Figure 3.36. The density evolution for one part of the setting, namely road 14 plus all succeeding roads is plotted. Blue colour refers to light traffic, whereas yellow colour denotes heavy traffic. Dark red colour indicates a total traffic jam, where cars are standing still.

Now, we optimise the traffic light setting using Cplex [23] on the corresponding DTN-MIP (2.98) and obtain an optimal traffic light setting as shown in Figure 3.35(b). The optimal objective function value results in a objective function value of 96.63, which is a considerable increase of 44.57%. But as we can observe in Figure 3.35(b), the resulting traffic light setting is highly fluctuating and has too long red phases for the left-turning lanes (which are at road 1, 3, 5 and 7). The corresponding density evolution on the roads is shown in Figure 3.37. Compared to the default setting, the traffic jams are significantly reduced. Especially road 14 is free of total jams. However, a new jam appears at road 3, since the left-turning lanes have unacceptable long red phases.

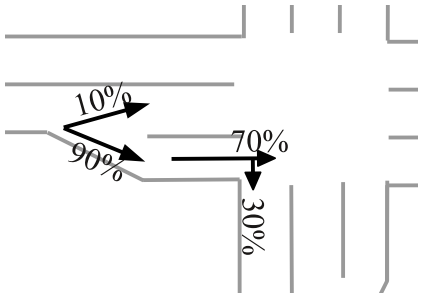
For this reason, we add restrictions on switching times to the model as described in Subsection 2.4.5 . We set the lower bound for each green phase to 12 seconds and the upper bound for each red phase to 80 seconds. After applying the optimisation software, we obtain an optimal solution. The corresponding traffic light setting depicted in Figure 3.35(c). The resulting objective function value is 88.61 which is still an increase of

time horizon:	$T = 10 \mathrel{\widehat{=}} 400$ s
lower bound on green phase:	$L^0 = 0.3 \mathrel{\widehat{=}} 12$ s (optional)
upper bound on red phase:	$U^1 = 2.0 \mathrel{\widehat{=}} 80$ s (optional)
time step size :	$\Delta t = 0.1$
space step size:	$\Delta x = 0.2$

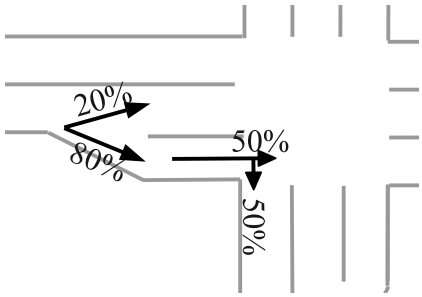
(a)

roads i	left turns 1/3/5/7	straight/right turns 2/4/6/8	outgoing roads 9-12	incoming roads 13-16
parameters for f				
λ	1	1	1	1
ρ^*	0.5	0.5	0.5	0.5
road length L_i	$0.5 \mathrel{\widehat{=}} 0.25$ km	$0.5 \mathrel{\widehat{=}} 0.25$ km	$1 \mathrel{\widehat{=}} 0.5$ km	$2 \mathrel{\widehat{=}} 1.0$ km
initial traffic density $\rho_i(x, 0)$	0.1	0.2	0.1	0.4

(b)



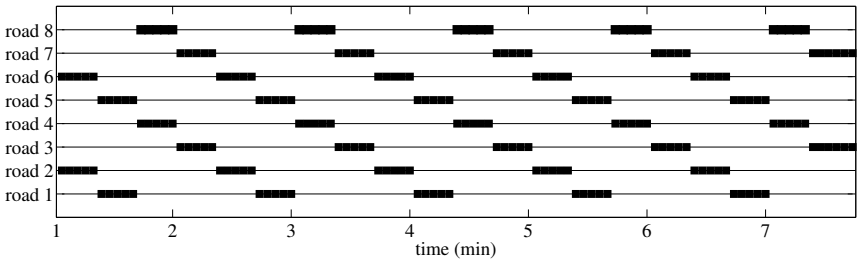
(c) Traffic distribution for each direction during the first half of the time horizon.



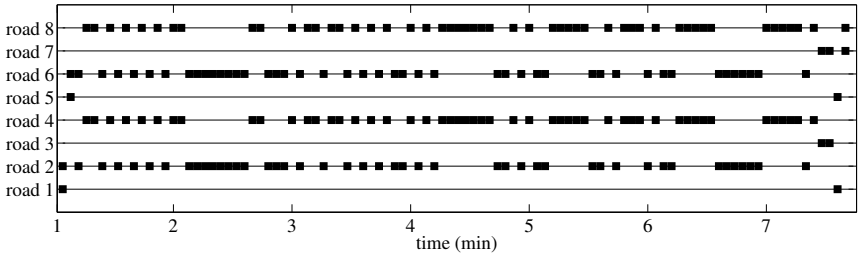
(d) Traffic distribution for each direction during the second half of the time horizon.

Figure 3.34: Parameter setting for crossover.

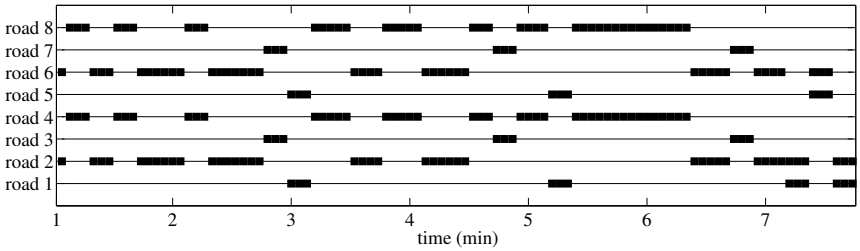
32.58% compared to the default traffic light setting. The density evolution is shown in Figure 3.35(b).



(a) Default traffic light setting, leads to an objective function value of 66.84.



(b) Optimised traffic light setting, leads to an objective function value of 96.63.



(c) Optimised traffic light setting including restrictions on switching time, leads to an objective function value of 88.61.

Figure 3.35: Traffic light settings. The beams indicate the time intervals, when the corresponding traffic lights are green, and the thin lines represent the time intervals, when the traffic light is red.

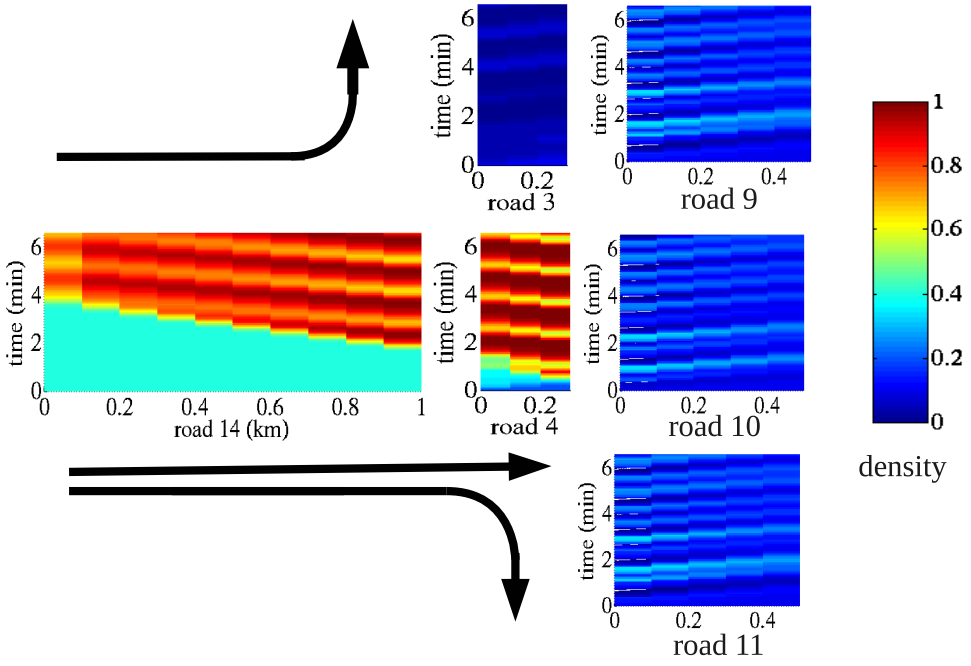


Figure 3.36: Traffic density using default traffic light setting.

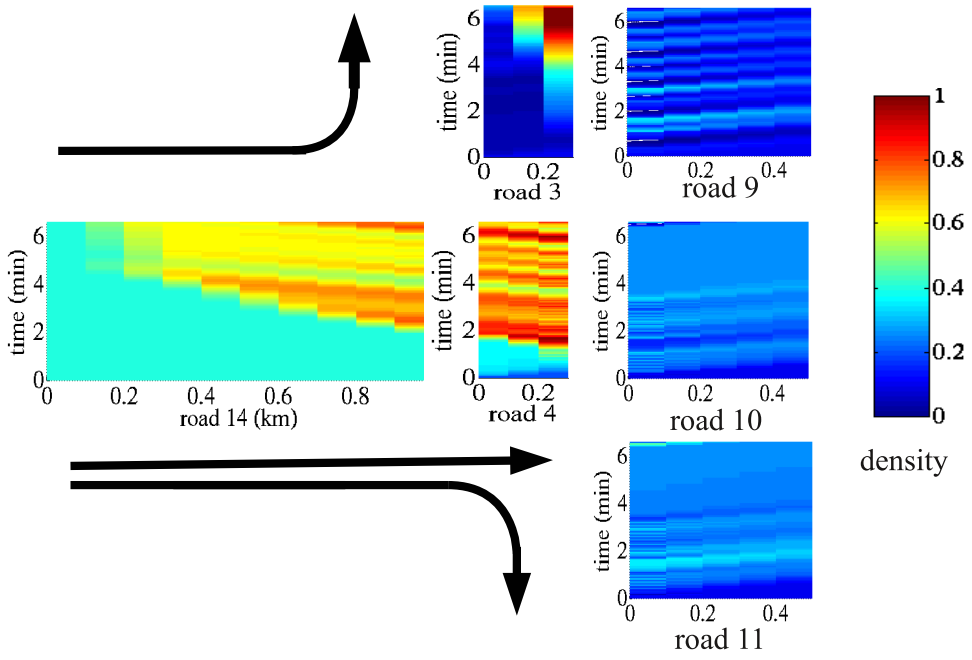


Figure 3.37: Traffic density using optimised traffic light setting.

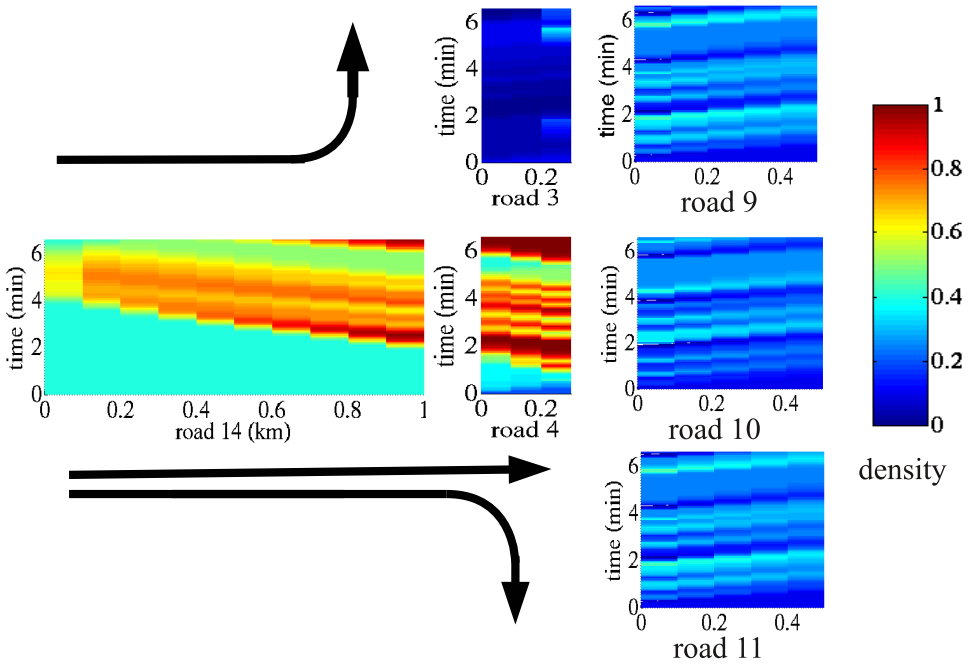
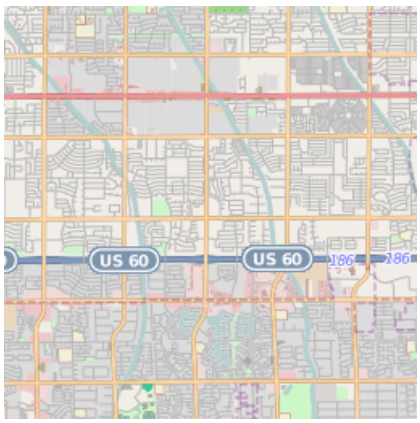


Figure 3.38: Traffic density using optimised traffic light setting including restrictions on switching times.

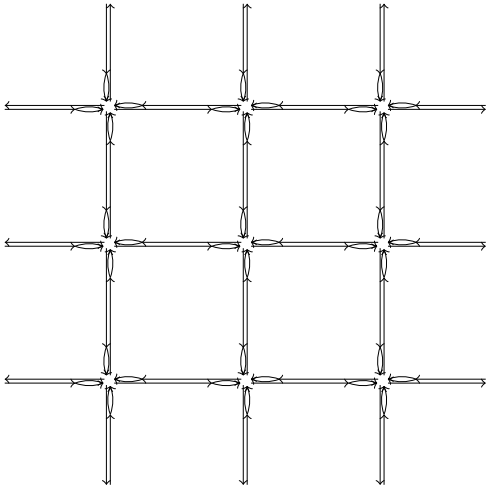
Road Network

We can apply the model techniques to larger road networks. In some cities the main roads are often arranged in chess pattern and the big crossovers have a distance of one mile from each other. As example see a part of a roadmap of Phoenix, cf. Figure 3.39(a). Motivated by these arrangement, we construct similar simplified road networks as a composition of several junctions as described in Figure 3.32, neglecting the small side roads.

In the sequel we will consider a fictive scenario, based on a network, which consists of nine crossovers, altogether assembled of 45 vertices and 120 roads, as shown in Figure 3.39(b).



(a) Examplary road network (part of Phoenix, Arizona), taken from ©OpenStreetMap, http://www.openstreetmap.org



(b) Road network with 120 edges and 45 vertices.

Figure 3.39: Road network.

We assume that for a certain time of the day the main traffic volume is moving from left to right. This is realised by prescribing higher boundary density on the ingoing roads from the left side compared to the other ingoing roads and by choosing the distribution matrix d in a way that cars preferably turn towards roads leading from left to right. Details can be seen in Table 3.5 and Figure 3.40.

First of all, we assume to have the default traffic light setting for each crossover as shown in Figure 3.35(a).

general parameters	time horizon T	time step size Δt	space step size Δx
	$10 \mathrel{\widehat{=}} 400$ s	0.25	0.5
roads	length L	initial density $\rho(x,0)$	left bound. density $\rho(0,x)$
incoming roads from top	2 ($\widehat{=}$1 km)	0.3	0.3
incoming roads from left	2 ($\widehat{=}$1 km)	0.5	0.5
incoming roads from right	2 ($\widehat{=}$1 km)	0.1	0.1
incoming roads from bottom	2 ($\widehat{=}$1 km)	0.3	0.3
left turning lanes	0.5 ($\widehat{=}$0.25 km)	0.1	–
straight/right turning lanes	0.5 ($\widehat{=}$0.25 km)	0.2	–
inner roads	4 ($\widehat{=}$2 km)	0.4	–
outgoing roads	2 ($\widehat{=}$1 km)	0.1	–
for all roads:	$\lambda = 1$	$\rho^* = 0.5$	

Table 3.5: Parameter setting of road network.

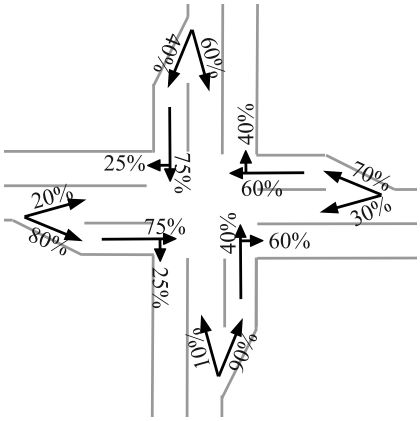
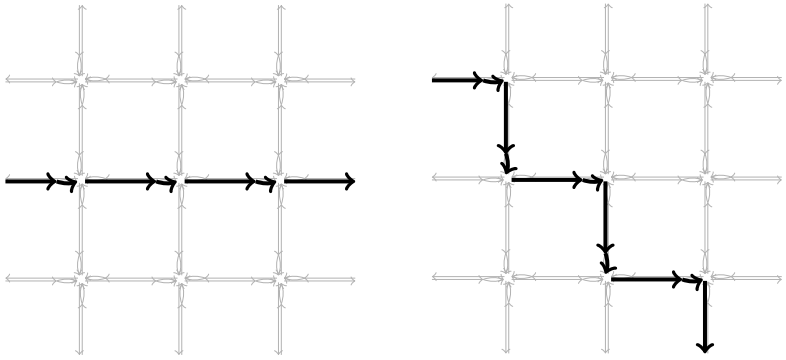


Figure 3.40: Traffic distribution for all crossovers.

We simulate the density evolution on the road with the prescribed parameter setting using the default traffic-light setting as shown in Figure 3.35(a) and use it as a start

solution for the corresponding DTN-MIP (2.98), as illustrated in Figure 2.5. The objective function value of the start solution is 695,2285. After optimisation, we get a solution with objective function value of 804,9520, which is an increase of 15,78%.

In order to get an idea of the improvements of the traffic situation using the optimised traffic light setting, we pick two paths through the network, as depicted in Figure 3.41. Path 1 is crossing the road network from left to right, as shown in Figure 3.41(a) and Path 2 is crossing the network diagonally from the upper left to the lower right corner, as shown in Figure 3.41(b).



(a) Path 1 (b) Path 2

Figure 3.41: Paths through road network.

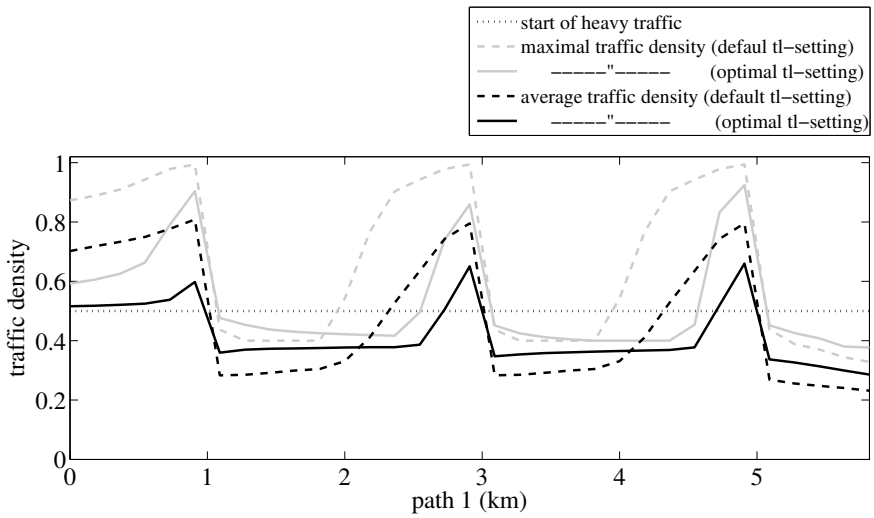
We compare the traffic density along the paths under default and optimised traffic light setting. The black line in Figure 3.42(a) describes the averaged traffic density under optimised conditions and the dashed black line describes the averaged traffic density under default traffic light setting. The gray lines indicate the corresponding maximal values during the whole time horizon. In Figure 3.42(b) the same comparison is done for average and minimal travel velocity v which is computed by

$$v = \frac{f}{\rho}.$$

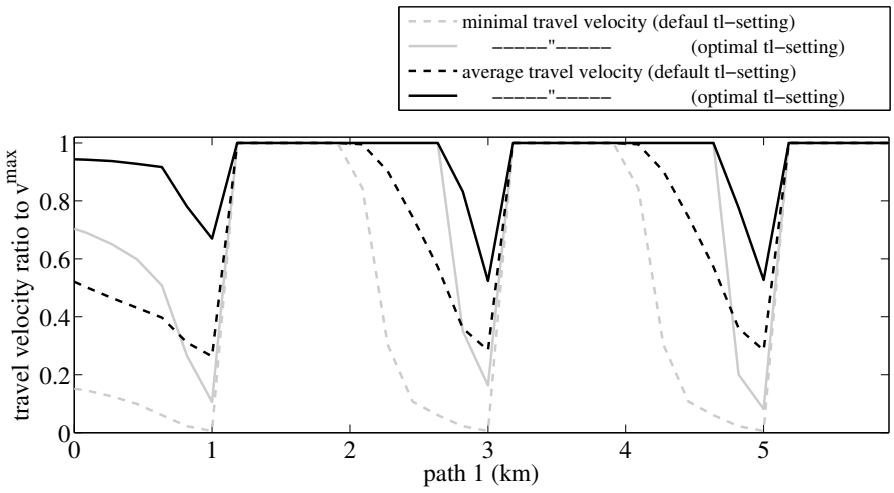
Remark 3.2.1. *If we consider the underlying flow function 1.26, we see that the travel velocity is maximal, as long as the density is smaller or equal than ρ^*. For dense traffic, i.e. for $\rho \geq \rho^*$, it decreases monotonically until it reaches zero for $f(\rho^{max})$.*

Note, the considerable improvement of density, especially in front of the crossovers, which are found at km 1, 3 and 5 on the x-axis. The travel velocity along the roads

is mostly at its maximal value. Cars only have to slow down in front of a crossover (see again km 1, 3 and 5 on the x-axis on Figure 3.42(b)). Again, the travel velocity is considerably higher when the optimised traffic light setting is used.

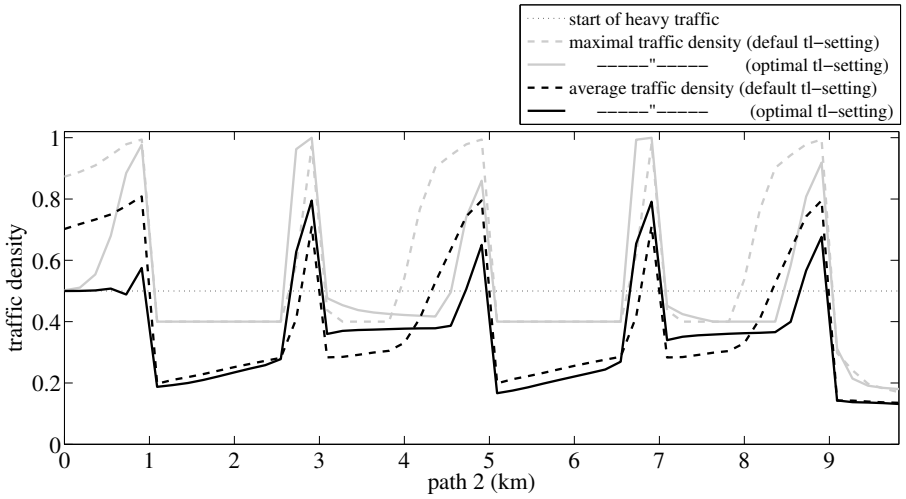


(a) Traffic density.

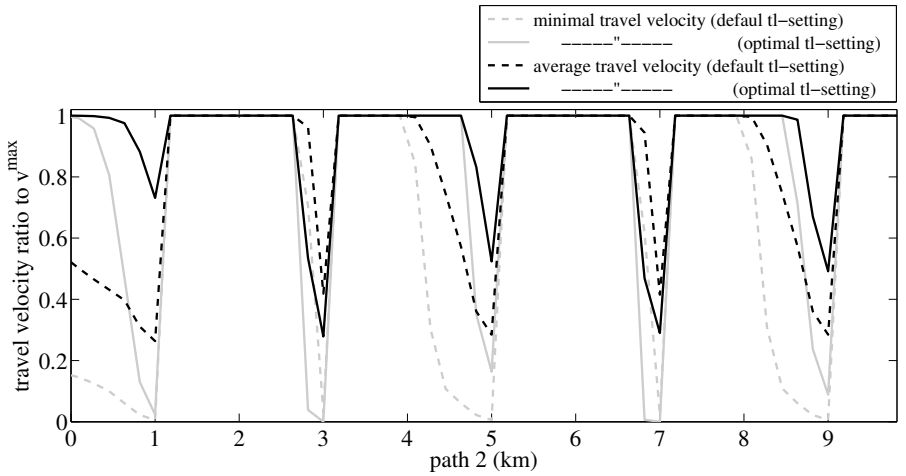


(b) Travel velocity.

Figure 3.42: Traffic evolution along Path 1.



(a) Traffic density.



(b) Travel velocity.

Figure 3.43: Traffic evolution along Path 2.

The same comparisons are done for path 2. The average and maximal traffic density along the path is depicted in Figure 3.43(a) and the average and minimal travel velocities can be seen in Figure 3.43(b). Since the aim of optimisation is to increase the traffic flow globally for the whole network, it can happen that optimised traffic light settings locally lead to slight setbacks, especially for roads which are not corresponding to the direction of the main traffic load. For this reason we can observe slight worsening of traffic densities in front of crossovers coming from a road leading from up to down. In this graph you find them on the x-axis around km 3 and 7. This is also observable for the travel velocity at the same points on the x-axis on Figure 3.43(b). However, these setbacks are more than compensated on the rest of the path.

Optimization Procedure

The main difficulties for the optimisation procedure is the huge problem size of the DTN-MIPs. In comparison to the DTN-MIP resulting from the production network model (2.37), where no detailed modelling along the edges is done, the traffic network also works with space grids along the roads. This additional dimension that has to be discretised results is an even more complex DTN-MIP, where the number of constraints and variables is in $\mathcal{O}(|E| \cdot |n_t| \cdot |n_k|)$.

For this reason, we stick to a rather coarse grid size as $\Delta x = 0.2$ and $\Delta t = 0.1$ for the modelling of the single crossover and $\Delta x = 0.5$ and $\Delta t = 0.25$ for the modelling of the road network.

The resulting MIP for the crossover consist of around $6 \cdot 10^4$ variables, the MIP for the roadnet consists of around $1.3 \cdot 10^5$ variables, which makes it almost impossible to find an optimal solution within acceptable computation time if no tuning techniques such as Algorithm 6 and 6 are applied (see illustration in Figure 2.7. We have a deeper look into the optimisation process, firstly, of the crossover model without additional restrictions on switching time (cf. Figure 3.44); secondly, of the crossover model including switching time restrictions (cf. Figure 3.45) and, thirdly, on the optimisation process of the roadnet (cf. Figure 3.46). All computations are performed on a PC equipped with 16GB Ram, Intel(R) Xeon(R) CPU 5160 @ 3.00GHz.

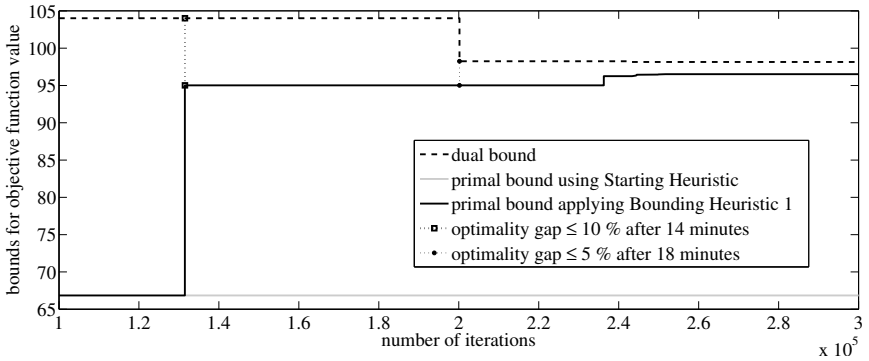
On Figures 3.44(a), 3.45(a) and 3.46(a) the evolution of the primal and dual bounds during the optimisation process is plotted. The number of iterations given on the x-axis refer to small computation units as for example iterations of the simplex method,

when computing the relaxed solution. For this reason the iteration number is roughly proportional to the number of rows in the linear MIP. The dashed black line denotes the dual bound and the black line the primal bound, when a bounding heuristic is used. The gray line represents the primal bound of the optimisation in the case that only a starting heuristic is used and no further bounding heuristics during the Branch & Cut algorithm. As starting solution we choose the default traffic light setting for each crossover as shown in Figure 3.35(a) and compute the remaining variables with the forward solver (cf. Algorithm 3), see illustration in Figure 2.5.

When we apply the bounding heuristics (cf. Algorithm 6 for the models without restrictions on switching times and Algorithm 7 for the crossover model with switching time restrictions), the primal bound improves soon after the start of the optimisation procedure. In Figures 3.44(a), 3.45(a) and 3.46(a) you see the points in time, when the optimality gap falls below 20%, 10% and 5%. As indicated in the corresponding tables, a strong improvement of the optimality gap is already achieved during the examination of the root node, where cutting planes techniques and the bounding heuristics are applied several times before the actual branching starts.

Tables 3.44(b), 3.45(b) and 3.46(b) show that the optimality gap that is obtained after 5 hours runtime cannot be exceedingly improved even after 3 days. This is due to the facts that firstly, many new found feasible solution are not better than the current incumbent. Secondly, the memory consumption slows down iteration time as soon as soon as the Branch & Bound tree gets large. For instance for the crossover model with switching time restrictions, cf. Figure 3.45, we have a gap of 42.75% after 5 hours runtime and still 37.09% after 3 days runtime, when no bounding heuristic is used. The use of the bounding heuristic extremely helps to close the optimality gap fast. As can be seen on Table 3.45(b) it is smaller than 10% after half an hour runtime. On the other scenarios we encounter a similar behaviour.

These results clearly show the importance of bounding heuristics for such large problem sizes in order to obtain reasonable solutions within an acceptable computation time.

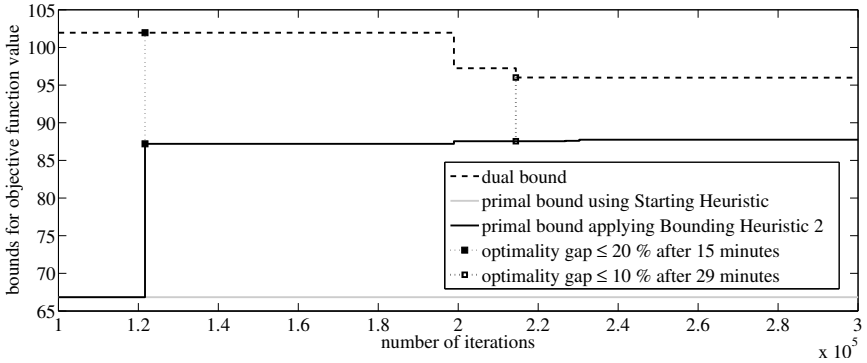


(a) Comparison of the evolution of primal and dual bounds during the optimisation procedure using only Starting Heuristic and using Bounding Heuristic 1.

		Starting Heuristic	Bounding Heuristic 1
after	primal bound	73.7952	96.6212
18000 s	dual bound	97.1187	98.1543
(5 hours)	optimality gap	31.62%	1.59%
after	primal bound	75.0262	96.6331
259200 s	dual bound	97.0759	98.1538
(3 days)	optimality gap	29.39%	1.57%
optimality	# nodes	–	0 (root node)
gap	# iterations	(not obtained)	131290
$\leq 20\%$	elapsed time	–	813 s ($\approx$ 14 m)
optimality	# nodes	–	0 (root node)
gap	# iterations	(not obtained)	131290
$\leq 10\%$	elapsed time	–	813 s ($\approx$ 14 m)
optimality	# nodes	–	0 (root node)
gap	# iterations	(not obtained)	200173
$\leq 5\%$	elapsed time	–	1108 s ($\approx$ 18 m)
improvement of optimised traffic light setting			44.57%

(b)

Figure 3.44: Comparison of optimisation procedures using Starting Heuristic and using Bounding Heuristic 1. Here, we consider the DTN-model for crossover that optimises the traffic light setting without additional requirements on switching times.

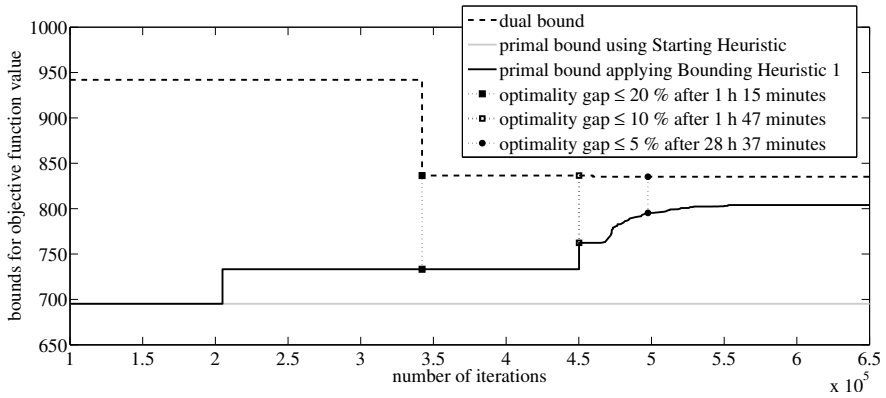


(a) Comparison of the evolution of primal and dual bounds during the optimisation procedure using only Starting Heuristic and using Bounding Heuristic 2.

		Starting Heuristic	Bounding Heuristic 2
after	primal bound	66.8394	87.7438
18000 s	dual bound	95.4144	95.9824
(5 hours)	optimality gap	42.75%	9.39%
after	primal bound	69.1212	88.6135
259200 s	dual bound	94.7592	95.9740
(3 days)	optimality gap	37.09%	8.31%
optimality	# nodes	–	0 (root node)
gap	# iterations	(not obtained)	121648
$\leq 20\%$	elapsed time	–	916 s ($\approx$ 15 m)
optimality	# nodes	–	0 (root node)
gap	# iterations	(not obtained)	214403
$\leq 10\%$	elapsed time	–	1751 s ($\approx$ 29 m)
optimality	# nodes	–	–
gap	# iterations	(not obtained)	(not obtained)
$\leq 5\%$	elapsed time	–	–
improvement of optimised traffic light setting			32.58%

(b)

Figure 3.45: Comparison of optimisation procedures using Starting Heuristic and using Bounding Heuristic 2. Here, we consider the DTN-model for crossover that optimises the traffic light setting including additional requirements on switching times.



(a) Comparison of the evolution of primal and dual bounds during the optimisation procedure using only Starting Heuristic and using Bounding Heuristic 1.

		Starting Heuristic	Bounding Heuristic 1
after	primal bound	695.2285	763.2116
18000 s	dual bound	822.4174	835.1736
(5 hours)	optimality gap	18.29%	9.43%
after	primal bound	708.3082	804.9520
259200 s	dual bound	822.1958	835.1453
(3 days)	optimality gap	16.08%	3.75%
optimality	# nodes	600	0 (root node)
gap	# iterations	678788	342171
$\leq 20\%$	elapsed time	85658 s ($\approx$ 2h 23 m)	4524 s ($\approx$ 1 h 15 m)
optimality	# nodes	–	0 (root node)
gap	# iterations	(not obtained)	450166
$\leq 10\%$	elapsed time	–	6419 s ($\approx$ 1 h 47 m)
optimality	# nodes	–	558
gap	# iterations	(not obtained)	497555
$\leq 5\%$	elapsed time	–	102941 s ($\approx$ 28h 37 m)
improvement of optimised traffic light setting			15.78%

(b)

Figure 3.46: Comparison of optimisation procedures using Starting Heuristic and using Bounding Heuristic 1. Here, we consider the DTN-model for road network.

Conclusion

In this work we provided a general classification of dynamic transportation networks (DTNs), which represent macroscopic PDE/ODE-based descriptions of network flow problems. There is a broad variety of versions depending on the application; for example it is possible to model buffers, to describe the evolution of density by conservation laws and to model different kinds of coupling conditions. Afterwards we considered optimisation techniques. We discussed the advantages of mixed integer optimisation and presented a general strategy how DTNs can be transformed into linear MIPs. Furthermore, we showed how the knowledge of the problem structure can be used to introduce bounding heuristics which are extremely efficient to speed up the optimisation procedure. Within this frame, we presented specific models with application in production and traffic.

The first is a novel production model for the time-changing repair worker assignment. The main idea is to keep the system performance optimal whenever machines have failed and must be repaired. In general, available workers are limited and therefore a decision has to be made on which machines are repaired first. The resulting optimisation question is how the optimal worker scheduling looks like to maximise the production flow. This issue has been intensively analysed and numerical case studies comparing fixed and time-changing schedules have been performed. As we have seen, the numerical results demonstrate the different opportunities of our modelling approach.

With respect to the second application, we considered the LWR-based traffic flow network model [19]. We showed how coupling conditions of several junction types can be transformed into easily linearisable min-terms. We introduced a numerical framework for the Hamilton-Jacobi formulation of traffic flow and showed how this

correctly resolves the dynamics at the junction. We presented simulations for a roundabout and compared them with existing results and computed travel times for certain routes through the network depending on the starting time of the travel. Moreover, we modelled traffic light settings for LWR-based traffic flow networks that can easily be adapted to arbitrary junction types and network topologies and discussed requirements for secure traffic light settings. We showed the necessity of additional requirements on the switching time rate to avoid inapplicably frequent fluctuations which appear when mixed integer optimisation techniques are used, and solved this problem with previously derived techniques. Furthermore, we developed a bounding heuristic to speed up the optimisation process. The resulting improvements for the optimisation procedure are remarkable and indicate the potential of combining simulation techniques with Branch & Bound procedures.

Altogether, this work illustrates, how the combination of various different mathematical fields – in our case coupled PDE/ODE-systems, numerical computation and discrete optimisation techniques – allow for detailed dynamic network descriptions and reliable optimisation. The remarkable improvements of the optimisation procedure lead to the assumption that there is still a lot of potential hidden in the connection of these fields. One important aspect would be the application of Branch & Bound techniques on DTNs that are not linearisable. A second point is to find ways to allow for finer discretisation grids without the inflation of problem size and optimisation time. A promising approach to obtain both aims is the development of an adapted Branch & Bound procedure including an integrated forward solver to obtain primal bounds and a novel strategy to obtain dual bounds without the necessity of laborious linearisation techniques.

Generally speaking, further research on the intersection of numerical computation and discrete optimisation is a worthwhile task full of of potential.

References

[1] Y. Achdou, F. Camilli, A. Cutri, N. Tchou, et al. Hamilton-jacobi equations on networks. 2010. 3, 40

[2] D. Armbruster, C. de Beer, M. Freitag, T. Jagalski, and C. Ringhofer. Autonomous control of production networks using a pheromone approach. *Physica A: Statistical Mechanics and its applications*, 363(1):104–114, 2006. 1, 5, 13, 18

[3] D. Armbruster, P. Degond, and C. Ringhofer. A model for the dynamics of large queuing networks and supply chains. *SIAM Journal on Applied Mathematics*, pages 896–920, 2006. 1, 2, 5, 16

[4] D. Armbruster, S. Göttlich, and M. Herty. A continuous model for supply chains with finite buffers. 2010. 2, 13, 16

[5] D. Armbruster, D. Marthaler, and C. Ringhofer. Kinetic and fluid model hierarchies for supply chains. *Multiscale Modeling and Simulation*, 2:43–61, 2004. 5

[6] J. Bang-Jensen and G. Gutin. *Digraphs: theory, algorithms and applications*. Springer Verlag, 2009. 92, 127

[7] J. Banks and J.S. Carson. Discrete event system simulation. 1984. 2, 5, 16

[8] C. Beard and A. Ziliaskopoulos. System optimal signal optimization formulation. *Transportation Research Record: Journal of the Transportation Research Board*, 1978(-1):102–112, 2006. 2, 57, 59, 93

[9] G. Bolch. *Queueing networks and Markov chains: modeling and performance evaluation with computer science applications*. Wiley-Blackwell, 2006. 1, 16

[10] A. Bressan. *Hyperbolic systems of conservation laws: the one-dimensional Cauchy problem*, volume 20. Oxford University Press, USA, 2000. 12, 27, 44

[11] G. Bretti, R. Natalini, and B. Piccoli. Numerical approximations of a traffic flow model on networks. *Networks and Heterogeneous Media*, 1(1):57, 2006. 1, 5, 15, 24, 25, 27, 28, 30, 31, 32, 33, 38, 141

[12] G. Bretti and B. Piccoli. A tracking algorithm for car paths on road networks. *SIAM Journal on Applied Dynamical Systems*, 7(2):510–531, 2008. 24, 27, 38, 40

[13] E. Brockfeld, R. Barlovic, A. Schadschneider, and M. Schreckenberg. Optimizing traffic lights in a cellular automaton model for city traffic. *Physical Review E*, 64(5):056132, 2001. 57

[14] T.H. Chang and J.T. Lin. Optimal signal timing for an oversaturated intersection. *Transportation Research Part B: Methodological*, 34(6):471–491, 2000. 93

[15] T.H. Chang and G.Y. Sun. Modeling and optimization of an oversaturated signalized network. *Transportation Research Part B: Methodological*, 38(8):687–707, 2004. 57

[16] H. Chen and D.D. Yao. *Fundamentals of queueing networks: Performance, asymptotics, and optimization*, volume 46. Springer Verlag, 2001. 1, 16

[17] Y. Chitour and B. Piccoli. Traffic circles and timing of traffic lights for cars flow. *Discrete and Continuous Dynamical Systems Series B*, 5(3):599, 2005. 1, 5, 24, 32, 33

[18] C.G. Claudel and A.M. Bayen. Convex formulations of data assimilation problems for a class of hamilton-jacobi equations. *SIAM Journal on Control and Optimization*, 49(2):383, 2011. 1, 6, 25, 40

[19] G.M. Coclite, M. Garavello, and B. Piccoli. Traffic flow on a road network. *SIAM journal on mathematical analysis*, 36(6):1862–1886, 2005. 1, 5, 15, 24, 28, 38, 100, 167

[20] RM Colombo, G. Guerra, M. Herty, and V. Schleper. Optimal control in networks of pipes and canals. *SIAM Journal on Control and Optimization*, 48(3):2032–2050, 2009. 1, 5

[21] R. Corthout, G. Flötteröd, F. Viti, and C.M.J. Tampère. Non-unique flows in macroscopic first-order intersection models. *Transportation Research Part B: Methodological*, 46(3):343–359, 2012. 24

[22] T.J. Cova and J.P. Johnson. A network flow model for lane-based evacuation routing. *Transportation Research Part A: Policy and Practice*, 37(7):579–604, 2003. 1, 2, 57

[23] IBM ILOG CPLEX. Ibm deutschland gmbh, 71137 ehningen. Information available at `http://www-01.ibm.com/software/integration/optimization`; visited on June 2012. 2, 67, 77, 118, 124, 125, 149, 150

[24] C. Daganzo. *A theory of supply chains*. Number 526. Springer Verlag, 2003. 1, 5

[25] C.F. Daganzo. The cell transmission model, part ii: network traffic. *Transportation Research Part B: Methodological*, 29(2):79–93, 1995. 6, 24, 32, 33

[26] C.F. Daganzo. A continuum theory of traffic dynamics for freeways with special lanes. *Transportation Research Part B: Methodological*, 31(2):83–102, 1997. 5

[27] C.F. Daganzo. A variational formulation of kinematic waves: basic theory and complex boundary conditions. *Transportation Research Part B: Methodological*, 39(2):187–196, 2005. 40

[28] C.F. Daganzo. On the variational theory of traffic flow: well-posedness, duality and applications. 2006. 24, 25, 40

[29] R.J. Dakin. A tree-search algorithm for mixed integer programming problems. *The Computer Journal*, 8(3):250–255, 1965. 60

REFERENCES

[30] G.B. Dantzig. *Linear programming and extensions*. Princeton Univ Pr, 1998. 2, 32, 34, 36, 58

[31] C. d'Apice, S. Göttlich, M. Herty, and B. Piccoli. *Modeling, Simulation, and Optimization of Supply Chains: A Continuous Approach*. Society for Industrial Mathematics, 2010. 2, 13, 16

[32] C. D'Apice and R. Manzo. A fluid dynamic model for supply chains. *Networks and Heterogeneous Media*, 1(3):379–398, 2006. 1, 2, 5, 6, 16

[33] C. D'Apice, R. Manzo, et al. Splitting of traffic flows to control congestion in special events. *International Journal of Mathematics and Mathematical Sciences*, 2011, 2011. 24

[34] C. D'Apice, R. Manzo, and B. Piccoli. A fluid dynamic model for telecommunication networks with sources and destinations. *SIAM Journal on Applied Mathematics*, 68(4):981–1003, 2008. 1, 5, 25

[35] C. D'Apice, R. Manzo, and B. Piccoli. Modelling supply networks with partial differential equations. *Quart. Appl. Math*, 67:419–440, 2009. 1, 5

[36] A. Dittel, A. Fügenschuh, S. Göttlich, and M. Herty. Mip presolve techniques for a pde-based supply chain model. *Optimization Methods & Software*, 24(3):427–445, 2009. 3, 18, 43, 59

[37] A. Dittel, S. Göttlich, and U. Ziegler. Optimal design of capacitated production networks. *Optimization and Engineering*, 12(4):583–602, 2011. 2, 13, 18, 21, 43, 57, 59, 77, 82, 91

[38] W. Domschke, L. Häselbarth, and A. Scholl. Wisulexikon operations research. *Publications of Darmstadt Technical University, Institute for Business Studies (BWL)*, 2003. 63

[39] G. Flötteröd and J. Rohde. Operational macroscopic modeling of complex urban road intersections. *Transportation Research Part B: Methodological*, 2011. 24, 93

[40] DR Ford and D.R. Fulkerson. *Flows in networks*. Princeton university press, 2010. 5, 8

[41] L.R. Ford and D.R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8(3):399–404, 1956. 1, 5, 8

[42] A. Fügenschuh, S. Göttlich, M. Herty, A. Klar, and A. Martin. A discrete optimization approach to large scale supply networks based on partial differential equations. *SIAM journal on scientific computing*, 30:1490, 2008. 1, 2, 5, 13, 16, 57, 59, 71, 82, 91, 118, 137

[43] M. Garavello and B. Piccoli. *Traffic flow on networks*. American institute of mathematical sciences Springfield,, USA, 2006. 1, 5, 24, 27, 28

[44] D.C. Gazis. Optimum control of a system of oversaturated intersections. *Operations Research*, pages 815–831, 1964. 57

[45] S.K. Godunov. A difference method for numerical calculation of discontinuous solutions of the equations of hydrodynamics. *Matematicheskii Sbornik*, 89(3):271–306, 1959. 47, 55

[46] S. Göttlich. *Continuous Models for Production Networks Including Optimization Issues*. Verl. Dr. Hut, 2007. 1, 2, 18, 57, 82

[47] S. Göttlich, M. Herty, and A. Klar. Network models for supply chains. *Communications in Mathematical Sciences*, 3(4):545–559, 2005. 1, 2, 5, 16, 21, 44, 118, 137

[48] S. Göttlich, M. Herty, and C. Ringhofer. Optimization of order policies in supply networks. *European Journal of Operational Research*, 202(2):456–465, 2010. 2

[49] S. Göttlich, M. Herty, C. Ringhofer, and U. Ziegler. Production systems with limited repair capacity. 2011. 2, 3, 6, 18, 43, 59, 82

[50] S. Göttlich, O. Kolb, and S. Kühn. Interpretation of the dual of a mip with discrete adjoints. *preprint*, 2012. 108, 112

[51] H. Hamacher and K. Klamroth. *Lineare und Netzwerk-Optimierung: Ein bilinguales Lehrbuch*. Friedrick Vieweg & Son, 2000. 63, 126

[52] H.W. Hamacher and S.A. Tjandra. Mathematical modelling of evacuation problems–a state of the art. *Pedestrian and Evacuation Dynamics*, 2002:227–266, 2002. 1, 2, 5, 57

[53] Q. He, W.H. Lin, H. Liu, and KL Head. Heuristic algorithms to solve 0-1 mixed integer lp formulations for traffic signal control problems. In *Service Operations and Logistics and Informatics (SOLI), 2010 IEEE International Conference on*, pages 118–124. IEEE, 2010. 2, 59, 93

[54] D. Helbing. Production, supply, and traffic systems: A unified description. *Traffic and Granular Flow 03*, pages 173–188, 2005. 1, 5

[55] M. Herty and A. Klar. Modeling, simulation, and optimization of traffic flow networks. *SIAM Journal on Scientific Computing*, 25:1066, 2003. 24, 28, 31

[56] M. Herty and M. Rascle. Coupling conditions for a class of second-order models for traffic flow. *SIAM journal on mathematical analysis*, 38(2):595–616, 2007. 24, 33

[57] M. Herty and C. Ringhofer. Optimization for supply chain models with policies. *Physica A: Statistical Mechanics and its Applications*, 380:651–664, 2007. 2

[58] H. Holden and N.H. Risebro. A mathematical model of traffic flow on a network of unidirectional roads. *SIAM Journal on Mathematical Analysis*, 26:999, 1995. 24, 26, 27

[59] H. Holden and N.H. Risebro. *Front tracking for hyperbolic conservation laws*, volume 152. Springer Verlag, 2011. 27

[60] LINDO SYSTEMS INC. 1415 north dayton street, chicago, il 60642, usa. Information available at http://www.lindo.com; visited on June 2012. 2, 67

[61] G.S. Jiang, D. Levy, C.T. Lin, S. Osher, and E. Tadmor. High-resolution nonoscillatory central schemes with nonstaggered grids for hyperbolic conservation laws. *SIAM Journal on Numerical Analysis*, pages 2147–2168, 1998. 44, 103

[62] J. Kallrath. *Gemischt-ganzzahlige Optimierung: Modellierung in der Praxis: mit Fallstudien aus Chemie, Energiewirtschaft, Metallgewerbe, Produktion und Logistik*. Vieweg+ Teubner, 2002. 2, 59, 70, 71, 73

[63] C.T. Kelley. *Iterative methods for optimization*, volume 18. Society for Industrial Mathematics, 1999. 2, 58

[64] A. Klar, R.D. Kühne, and R. Wegener. *Mathematical models for vehicular traffic*. Arbeitsgruppe Technomathematik, Univ., 1995. 15

[65] O. Kolb. *Simulation and Optimization of Gas and Water Supply Networks, PhD TU Darmstadt*. Dr. Hut Verlag, 2011. 44, 103

[66] S.O. Krumke and H. Noltemeier. *Graphentheoretische Konzepte und Algorithmen*. Vieweg+ Teubner, 2009. 8, 126

[67] A. Kurganov and E. Tadmor. New high-resolution semi-discrete central schemes for hamilton–jacobi equations. *Journal of Computational Physics*, 160(2):720–742, 2000. 49

[68] S. Lämmer and D. Helbing. Self-control of traffic lights and vehicle flows in urban road networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008:P04019, 2008. 57

[69] A.H. Land and A.G. Doig. An automatic method of solving discrete programming problems. *Econometrica: Journal of the Econometric Society*, pages 497–520, 1960. 60

[70] E.L. Lawler and D.E. Wood. Branch-and-bound methods: A survey. *Operations research*, pages 699–719, 1966. 2, 58, 63

[71] P.D. Lax. *Hyperbolic Systems of Conservation Laws and the Mathematical Theory of Shock Wawes*, volume 17. SIAM, 1973. 27

[72] JP Lebacque and MM Khoshyaran. First order macroscopic traffic flow models for networks in the context of dynamic assignment. *Transportation Planning*, pages 119–140, 2004. 1, 5, 24

[73] R.J. LeVeque. *Numerical methods for conservation laws*. Birkhäuser, 1992. 6, 12, 41, 42, 43, 49

[74] M.J. Lighthill and G.B. Whitham. On kinematic waves. ii. a theory of traffic flow on long crowded roads. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 229(1178):317–345, 1955. 3, 23, 24

[75] W.H. Lin and C. Wang. An enhanced 0-1 mixed-integer lp formulation for traffic signal control. *Intelligent Transportation Systems, IEEE Transactions*, 5(4):238–245, 2004. 2, 59, 93

[76] H.K. Lo. A novel traffic signal control formulation. *Transportation Research Part A: Policy and Practice*, 33(6):433–448, 1999. 2, 57, 59, 93

[77] H.K. Lo. A cell-based traffic control formulation: strategies and benefits of dynamic timing plans. *Transportation Science*, 35(2):148–164, 2001. 2, 59, 93

[78] Y. Makigami, GF Newell, and R. Rothery. Three-dimensional representation of traffic flow. *Transportation Science*, 5(3):302–313, 1971. 24, 40

[79] P.E. Mazaré, A.H. Dehwah, C.G. Claudel, and A.M. Bayen. Analytical and grid-free solutions to the lighthill–whitham–richards traffic flow model. *Transportation Research Part B: Methodological*, 2011. 24, 25, 40

[80] K. Moskowitz and L. Newan. Notes on freeway capacity. *Highway Research Record*, 1963. 40

[81] K.G. Murty. Linear programming. 1983. 63

[82] G.L. Nemhauser and L.A. Wolsey. *Integer and combinatorial optimization*, volume 18. Wiley New York, 1988. 2, 58, 63

[83] G.F. Newell. A simplified theory of kinematic waves in highway traffic, part i: General theory, part ii: Queuing at freeway bottlenecks, part iii: Multi-destination flows. *Transportation Research Part B: Methodological*, 27(4):281–313, 1993. 6, 24, 40

[84] J. Nocedal and S.J. Wright. *Numerical optimization*. Number 2. Springer Series in Operations Research, Berlin, 2006. 63

[85] GUROBI Optimization. Box 1001, 3733-1 westheimer rd., houston tx, 77027, usa. Information available at http://www.gurobi.com/; visited on June 2012. 2, 67

[86] H. Prothmann, J. Branke, H. Schmeck, S. Tomforde, F. Rochner, J. Hahner, and C. Muller-Schloer. Organic traffic light control for urban road networks. *International Journal of Autonomous and Adaptive Communications Systems*, 2(3):203–225, 2009. 57

[87] Matlab Version R2010a. Mathworks, inc., 3 apple hill drive, natick, ma. Information available at http://www.mathworks.com; visited on June 2012. 118

[88] P.I. Richards. Shock waves on the highway. *Operations research*, pages 42–51, 1956. 3, 23, 24

[89] V. Riley and S.I. Gass. *Linear programming and associated techniques*. Number 5. Published for Operations Research Office, the Johns Hopkins University, by the Johns Hopkins Press, 1958. 69, 105

[90] A. Schrijver. *Theory of linear and integer programming*. John Wiley & Sons Inc, 1998. 2, 58, 60, 69

[91] SCIP. Tu darmstadt, discrete optimization university of erlangen-nürnberg, chair of edom siemens ag, corporate technology. Information available at http://scip.zib.de/; visited on June 2012. 2, 67

[92] E.F. Toro. *Riemann solvers and numerical methods for fluid dynamics: a practical introduction*. Springer Verlag, 2009. 27

[93] F. Tröltzsch. *Optimal control of partial differential equations: theory, methods, and applications*, volume 112. Amer Mathematical Society, 2010. 2, 58

[94] L.A. Wolsey. Integer programming. *IIE Transactions*, 32:273–285, 2000. 2, 58, 105

[95] E. Zauderer. *Partial differential equations of applied mathematics*, volume 71. Wiley-Interscience, 2011. 12

[96] L. Zhao, X. Peng, L. Li, and Z. Li. A fast signal timing algorithm for individual oversaturated intersections. *Intelligent Transportation Systems, IEEE Transactions on*, (99):1–4, 2011. 93

[97] H. Zidani, R. Monneau, and C. Imbert. A hamilton-jacobi approach to junction problems and application to traffic flows. 2011. 3, 40

[98] U. Ziegler and S. Göttlich. Design network problem and heuristics. *Progress in Industrial Mathematics at ECMI 2008*, pages 515–520, 2010. 2, 3, 18, 43, 57, 77, 82

LEBENSLAUF

Persönliche Daten

Name:	Ziegler
Vorname:	Ute
Geburtstag:	03.02.1983
Geburtsort:	Bensheim
Staatsangehörigkeit:	deutsch

Qualifikationen

2002	Abitur
2003 – 2008	Studium der Technomathematik *Anwendungsfach Physik*
Abschluss:	Diplom
ab 2008 – 2012	Promotion in Mathematik