Distributed Control of Dynamic Flows in Traffic Networks

Christian Rosdahl

LUND
UNIVERSITY

Department of Automatic Control

Abstract

In today's society, traffic congestion is a major problem in several aspects. Apart from the obvious problem that people are losing valuable time due to the resulting delays, it also has negative impact on as well the economy as the local and global environment. With the development of sensors and navigation support, it has now become possible and thus of interest to study optimal routing of vehicles in a traffic network, in order to reduce the congestion-related problems.

In this master's thesis, a distributed algorithm for solution of optimal dynamic traffic flow control problems is derived, implemented and tested. Traffic networks are modelled with the cell transmission model (CTM), and the solution algorithm is based on a generalization of the alternating direction method of multipliers (ADMM).

The algorithm is tested for one simple and one more complicated traffic network. The tests include both cases with time-varying external inflow of traffic as well as cases where the flow capacity of a specific road segment is varied with time, in order to simulate temporary traffic incidents.

The tests show that if the cost function is chosen as the sum of squares of the traffic volumes at the cells (road segments) of the network, the algorithm converges to the optimal solution if a specific parameter (the penalty parameter, or step length) is chosen sufficiently small.

The report starts with a description and examples from the simpler case of static traffic flow optimization. It also contains a summary of the concepts used from optimization theory. After this, the approach for dynamic traffic flow modelling and optimization is described. Finally, a description and derivation of the algorithm is provided, after which the implementation is tested for different cases involving the two different traffic networks.

Acknowledgements

I would like to thank my supervisors Giacomo Como and Gustav Nilsson, both at the Department of Automatic Control, Lund University, who have given me valuable feedback, advice and support during the work with this thesis project. I would also like to thank Enrico Lovisari who took time to look at my work and discuss and think about possible improvements. Finally, I want to thank the Department of Automatic Control in general for giving me the opportunity to work with this project.

Contents

Contents

1

Introduction

1.1 Background and Purpose

Today, over half of the world population lives in urban areas. In 2014, the urban population was 3.9 billion people, which can be compared to only 746 million in 1950. Furthermore, this number is expected to increase to over six billion by 2045 [UN, 2014]. A consequence of this is a dramatically increased loading of the transportation networks in many cities. Many people experience queues and traffic congestion daily, which costs time, causes frustration and decreases the quality of life. Moreover, this has negative environmental and climate effects and can also have economical disadvantages. Since extending the infrastructure can be both very costly and in some cases not even possible, there is a strong motivation for being able to utilize the existing resources as efficiently as possible.

By methods such as variable speed limits at roads, ramp metering – meaning that the number of vehicles that are allowed to enter a freeway at a particular time is limited – and prescribed turning ratios at junctions (e.g. through navigation support in the vehicles), the traffic flows in a given transportation network can be regulated. The question to be addressed in this project is how these control means should be chosen in order to fulfil the goal of utilizing the transportation network to its fullest potential and decreasing the delay for the vehicles. The methods that will be used to solve this optimization problem are distributed algorithms based on the dual ascent algorithm and the alternating direction method of multipliers (ADMM).

1.2 Related Work

In order to describe a dynamic traffic flow in a transportation network, the cell transmission model, originally presented in [Daganzo, 1994] and described in Section 3.2, is used. A problem with this model is that the resulting optimization problem with respect to the previously described control parameters becomes non-convex. However, in [Como et al., 2016] it is shown that the flow optimization

9

problem can be relaxed to a convex problem, and that the solution of this problem can be mapped back to control parameters for the original problem.

In [Ba et al., 2015], a similar problem is solved. This article also considers flow opimization with the cell transmission model using adaptations of the ADMM method and another method (accelerated dual descent). However, in this paper, the optimization problem is solved only for the static case, where all flows and traffic densities are constant in time.

Towards the end of this project, it was discovered that a solution of the same problem as considered here is provided in [Ba and Savla, 2016]. In this work, some interesting theoretical conclusions are drawn about the problem. However, the current project, which thus has been developed independently of [Ba and Savla, 2016], has at least two major differences. First of all, in this project, the inequality constraints are included in the Lagrangian function and as penalty terms in the augmented Lagrangian, while these instead are taken into account in the primal update step in the reference. Thus, even though both approaches use the cell transmission model and ADMM as starting points for solving the same problem, there are quite large differences between the approaches. The second important difference is that the reference does not contain any concrete implementation or any simulations, whereby it is not possible to get a sense of the speed of convergence of the algorithm. This project, on the other hand, includes a detailed description of how the method is implemented and uses MATLAB scripts to create simulation results which are presented for many different cases.

1.3 Method and Objective

The objective of this project is to derive and implement a distributed algorithm that can be used to determine how the speed limits, ramp metering and turning ratios in a given road network should be chosen in an optimal manner.

First, the special case when all the flows are constant in time, i.e., the static case, is considered. This is easy to model compared to the dynamic case, and thus this is suitable as a first step in the examination. The first goal is to implement the dual ascent algorithm and ADMM for the static case, and compare the results of these two different solutions to the problem. Another goal of this part is to gain insights about the problem and optimization methods that can be used for the more complicated dynamic optimization.

The remaining part of the project considers dynamic flow optimization where the traffic network is modelled by the cell transmission model. This model contains a large set of decision variables as well as both equality and inequality constraints, which complicates the optimization. In order to be able to solve optimization problems for larger networks with many time discretization points in the optimization horizon, it is of interest to have distributed methods for solving the problem. This means that the problem can be divided into many smaller subproblems which each

only requires information about a small subset of all variables, instead of information about all variables in the problem. This allows for scalability of the method, since even a problem involving a large network and many time points can be divided into problems which each involves few variables. These subproblems can in turn be solved in parallel on different processing units.

The main goal of the project is thus to derive a distributed method that can be used to solve the dynamic flow optimization problem. For a traffic network, the distributivity means that in order to determine traffic density and traffic flows at a specific part of the network, only densities and flows in the vicinity of the position in question have to be considered in each iteration of the algorithm. As a starting point for this algorithm, the ADMM and insights from the static flow optimization are used. The algorithm is then constructed by generalizing this to the dynamic flow optimization problem.

1.4   Outline

In **Chapter 2**, the special case of static flow optimization is considered. The chapter starts with a problem formulation of this case. After that, it continues with introducing the concepts of Lagrangian function and duality, which are used in the optimization. Following this, the dual ascent method is described and it is shown that this is distributed for the problem considered. Thereafter, the dual ascent algorithm is implemented and tested. After this, an interpretation is given of the dual variables, or Lagrange multipliers, which appears in the optimization method. The remaining part of the chapter is devoted to the ADMM, which is explained, adapted to the problem and tested.

In **Chapter 3**, the general dynamic flow optimization problem is considered. First, a short general background to traffic flow modelling is given. Thereafter, the cell transmission model is described. After this follows the problem formulation and a description of how the result from the convex optimization can be interpreted in terms of the assumed control parameters. The last part of the chapter defines the Lagrangian and dual problem for the dynamic case, which will be used in the optimization.

In **Chapter 4**, the dynamic flow optimization problem is reformulated in a way that allows for a distributed implementation of the optimization algorithm. This formulation and approach is based on the results from using the ADMM on the static flow optimization problem, and is an attempt of generalizing this. Thereafter, the necessary calculations and equations for implementing the method are carried out and described in detail.

Chapter 5 contains tests of the derived algorithm for dynamic flow optimization on two different problems and for different choices of the parameters in the algorithm. First, the problems and the methods by which they are compared are described. After this, the results from running the algorithm are presented. These

results are discussed in **Chapter 6** and conclusions from this are given in **Chapter 7**.

2

Static Network Flow Optimization

2.1 Problem Formulation

The first part of this project consists of determining how to optimally distribute a static traffic flow in a given road network. The road network can be described as a graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where $\mathcal{N} = \{n_1, n_2, \ldots, n_N\}$ is the set of **nodes**, corresponding to junctions, and $\mathcal{E} = \{e_1, e_2, \ldots e_E\}$ is the set of **links** (edges), corresponding to directed road segments between the nodes. For example, if there is a link directed from node $n_i \in \mathcal{N}$ to node $n_j \in \mathcal{N}$, then the set of links $\mathcal{E}$ contains the directed link $e = (n_i, n_j)$. The number of a node will be identified with the node, so that 'node i' refers to node n_i. Furthermore, the total number of nodes, i.e., $|\mathcal{N}|$, will be denoted N and the total number of links, i.e., $|\mathcal{E}|$, will be denoted E.

The relation between the nodes and links is given by the **node-link incidence matrix** $B \in \mathbb{R}^{N \times E}$. This matrix has one row for each node in $\mathcal{N}$, and one column for each link in $\mathcal{E}$. Column e consists of zeros except at the rows corresponding to the start and end node of link e, which contain $+1$ and -1 respectively. Expressed in equations,

$$B_{ie} = \begin{cases} +1 & \text{if} \quad e = (i, j) \\ -1 & \text{if} \quad e = (j, i) \\ 0 & \text{if} \quad e = (j, k) \end{cases}, \qquad (2.1)$$

for any node $i \in \mathcal{N}$, any link $e \in \mathcal{E}$ and nodes $j, k \neq i$.

At each node (junction), vehicles can enter the network from external roads not included in the modelled network. The external inflow and outflow at node i are given by element i in the column vectors $\lambda \in \mathbb{R}^N$ and $\mu \in \mathbb{R}^N$ respectively. Since only the net flows are of interest, it will be assumed that either $\lambda_i \geq 0$ and $\mu_i = 0$ or that $\lambda_i = 0$ and $\mu_i \geq 0$ for each i. Furthermore, since the flow is assumed to be static in this chapter, it is required that the total inflow to the network equals the

total outflow from it due to conservation of mass. This property can be stated as

$$\mathbf{1}^T \lambda = \mathbf{1}^T \mu = \tau,$$

where $\mathbf{1}$ is the all-one column vector, and τ is total flow though the network, termed the **throughput** of the network.

The flow at link $e = (i, j) \in \mathcal{E}$ is given by element e in the non-negative column vector $f \geq 0$, termed the **flow vector**. Conservation of mass, which is required in the static case, also implies that the total (external and internal) inflow to a node equals the total outflow from the node. Thus,

$$\lambda_i + \sum_{j:(j,i)\in\mathcal{E}} f_{(j,i)} = \sum_{j:(i,j)\in\mathcal{E}} f_{(i,j)} + \mu_i, \quad \forall i \in \mathcal{N}. \tag{2.2}$$

This can be equivalently, and more compactly, stated using the node-link incidence matrix (2.1):

$$Bf = \lambda - \mu. \tag{2.3}$$

For a given network topology (determining B), and given external in- and out-flows λ and μ, the task is thus to determine the flow vector $f \geq 0$ such that the mass conservation (2.3) is fulfilled. This problem has in general many solutions, and one further requirement is needed to determine which solution should be chosen. This requirement consists of minimizing a sum of cost functions $\psi_e(f_e)$, where ψ_e represents the cost for maintaining the flow f_e at link e. If the aim is to minimize the total delay for all travellers, the cost function should be chosen as a measure of the total delay that the f_e vehicles entering the link during one time unit experience. The cost functions are assumed to be defined for $f_e \geq 0$, be strictly increasing and convex, as well as fulfilling $\psi_e(0) = 0$.

The static network flow optimization problem can thus be stated as

$$\begin{aligned} \text{minimize} \quad & \psi(f) = \sum_{e\in\mathcal{E}} \psi_e(f_e), \quad f \geq 0 \\ \text{subject to} \quad & Bf = \lambda - \mu \end{aligned} \tag{2.4}$$

2.2 Lagrangian and Dual Problem

In this section, the Lagrangian function and the dual problem, which are used for solving the optimization problem (2.4), are defined and described. This theory is based on [Boyd and Vandenberghe, 2009, Ch. 5], in which it is more thoroughly described and further developed.

The **Lagrangian** of the problem (2.4) reads

$$L(f, \gamma) = \psi(f) + \gamma^T(\lambda - \mu - Bf), \quad f \geq 0, \tag{2.5}$$

where $\gamma \in \mathbb{R}^N$ are the **Lagrange multipliers** (or **dual variables**) corresponding to the equality constraints in (2.4). The domain $f \geq 0$ is inherited from the domain

of the objective function $\psi(f)$. The Lagrangian function incorporates information about both the objective function $\psi(f)$ and the constraints $Bf = \lambda - \mu$, and will be useful for solving (2.4). Moreover, the **dual function** is defined as

$$\theta(\gamma) = \inf_f L(f, \gamma). \tag{2.6}$$

For any feasible f, i.e., for any f fulfilling the constraint $Bf = \lambda - \mu$, it holds that

$$\theta(\gamma) \leq \psi(f). \tag{2.7}$$

This is referred to as **weak duality** and follows from noting that the Lagrangian, for any fixed γ, never is smaller than the dual function, according to the definition (2.6), and that the second term in the Lagrangian (2.5) is zero for feasible f. The relation (2.7) implies that the dual function $\theta(\gamma)$ for each γ constitutes a lower bound for the solution $\psi(f^*)$ of (2.4). From this it can be concluded that the solution $\psi(f^*)$ must be greater than or equal to the largest value $\theta(\gamma)$ can take, i.e., the solution of

$$\text{maximize} \quad \theta(\gamma), \tag{2.8}$$

which is called the **dual problem**. Thus, if a γ and an f are found such that f is feasible (i.e., fulfills the constraints in (2.4)), and so that $\psi(f) = \theta(\gamma) \equiv M$, it must according to (2.7) hold that M is the optimal value of both the primal problem (2.4) and the dual problem (2.8). A problem for which this is achievable is said to have **zero optimal duality gap**, and in this case one says that **strong duality** holds.

An optimization problem with a convex objective function, inequality constraints on the form $\varphi \leq 0$, where φ are convex functions, and affine equality constraints, is said to be a **convex problem**. Such problems usually have strong duality, i.e., zero optimal duality gap. It can be proved that this is the case for a particular problem by checking one of many so-called **constraint qualifications**. The problem (2.4) has a convex objective function, since it is a sum of convex functions, and affine equality constraints. Thus, it is a convex problem. In the sequel, it will be assumed that it enjoys strong duality.

2.3   The Dual Ascent Method

One optimization algorithm that can be used to solve the convex optimization problem (2.4) is the **dual ascent method**. This method takes advantage of the assumed strong duality, and uses the gradient of the dual function (2.6) to find the γ solving the dual problem (2.8). The following explanation is based on the description in [Boyd et al., 2011].

Assume that f^* and γ^* are the primal and dual optimal points, i.e., the f and γ which solve the primal problem (2.4) and the dual problem (2.8) respectively. Since strong duality is assumed to hold, this implies that

$$\psi(f^*) = \theta(\gamma^*) \equiv \inf_f L(f, \gamma^*). \tag{2.9}$$

Since the optimal point f^* is feasible with respect to the constraint in the primal problem (2.4), the second term in the Lagrangian (2.5) becomes zero for any γ, and thus

$$L(f^*, \gamma) = \psi(f^*).$$

Inserting $\gamma = \gamma^*$ in this equation and combining it with (2.9), it is concluded that

$$f^* = \operatorname*{argmin}_{f} L(f, \gamma^*).$$

In order to compute the gradient of the dual function, $\nabla\theta(\gamma)$, the dual function is formulated in terms of the Lagrangian, according to the definition (2.6). This gives

$$\theta(\gamma) = L(\tilde{f}, \gamma) = \psi(\tilde{f}) + \gamma^T (\lambda - \mu - B\tilde{f}), \quad \text{where} \quad \tilde{f} = \operatorname*{argmin}_{f} L(f, \gamma).$$

From this, the gradient is obtained as

$$\nabla\theta(\gamma) = (\lambda - \mu - B\tilde{f}), \quad \text{where} \quad \tilde{f} = \operatorname*{argmin}_{f} L(f, \gamma). \tag{2.10}$$

The dual ascent method proceeds as follows: Assume that non-optimal primal and dual points, f^k and γ^k, are given. The dual function gradient evaluated in γ^k is according to (2.10) given by

$$\nabla\theta(\gamma^k) = (\lambda - \mu - Bf^{k+1}), \quad \text{where} \quad f^{k+1} = \operatorname*{argmin}_{f \geq 0} L(f, \gamma^k).$$

The sought optimal dual point, γ^*, is the γ which maximizes $\theta(\gamma)$. To find a γ for which the dual function is larger than for γ^k, it is reasonable to move in the direction of the gradient. Thus, a point γ^{k+1} is found according to

$$\gamma^{k+1} := \gamma^k + \alpha\nabla\theta(\gamma^k) = \gamma^k + \alpha(\lambda - \mu - Bf^{k+1}),$$

where $\alpha > 0$ is the size of the step taken in the gradient direction. In summary, the dual ascent method operates by iteratively computing

$$f^{k+1} := \operatorname*{argmin}_{f} L(f, \gamma^k) \tag{2.11}$$

$$\gamma^{k+1} := \gamma^k + \alpha(\lambda - \mu - Bf^{k+1}). \tag{2.12}$$

If the step size α is chosen appropriately and some other conditions are fulfilled, the values of f^{k+1} and γ^{k+1} in the dual ascent algorithm will converge to the optimal primal and dual optimal points f^* and γ^*. In particular, γ will continue to change as long as f is not feasible, according to equation (2.12). However, the necessary conditions for convergence are in many cases not fulfilled (e.g., if the Lagrangian as a function of f does not have a minimum but can be arbitrarily small), whereby the algorithm is not applicable. This is a major drawback of the dual ascent approach.

To determine when to stop the iterative scheme, a stopping criterion is needed. Since strong duality is assumed to hold, the primal and dual objective functions, $\psi(f)$ and $\theta(\gamma)$, should be equal at optimality. An appropriate criterion is thus to require that the duality gap $\psi(f) - \theta(\gamma)$ (which always is positive for feasible f according to (2.7)) is small. The stopping criterion can thus be chosen such that the iterations should go on until

$$|\psi(f) - \theta(\gamma)| < \text{tol}$$

for some small tolerance tol > 0, in combination with requiring that the solution is feasible, i.e., fulfils

$$Bf - \lambda - \mu = 0.$$

2.4   Distributed Formulation of the Dual Ascent Method

A major advantage of the dual ascent method is that it in the present case can be formulated as a distributed algorithm. This means that the flow on any link can be determined from the Lagrange multipliers γ_i for the nodes to which it is connected only. Moreover, the Lagrange multiplier γ_i associated with any node i can be determined from the total in- and outflows to that node only.

The Lagrangian (2.5) can be decomposed according to

$$L(f, \gamma) = \sum_{e \in \mathcal{E}} \underbrace{\left(\psi_e(f_e) - \gamma^T B_e f_e\right)}_{L_e(f_e, \gamma)} + \gamma^T(\lambda - \mu),$$

where B_e denotes column e of the node-link incidence matrix B. From the definition of the node-link incidence matrix, (2.1), it can be seen that

$$\gamma^T B_e = (\gamma_i - \gamma_j), \quad \text{where} \quad e = (i, j),$$

which confirms that only the Lagrange multipliers for the nodes of the link in question are needed to compute the flow on this link. The Lagrangian thus becomes

$$L(f, \gamma) = \sum_{e \in \mathcal{E}} \underbrace{\left(\psi_e(f_e) - (\gamma_i - \gamma_j)f_e\right)}_{L_e(f_e, \gamma)} + \gamma^T(\lambda - \mu), \tag{2.13}$$

and the Lagrangian minimization (2.11) can thus be carried out by minimizing each term $L_e(f_e, \gamma)$ (corresponding to each link) separately, giving E minimization problems in one variable. The flow on link e at iteration $k+1$ is thereby given as

$$f_e^{k+1} := \underset{f_e \geq 0}{\text{argmin}}\, L_e(f_e, \gamma^k) = \underset{f_e \geq 0}{\text{argmin}}\,(\psi_e(f_e) - (\gamma_i^k - \gamma_j^k)f_e).$$

The minimization is carried out by studying the derivative of the Lagrangian term L_e. It becomes

$$\frac{\partial L_e}{\partial f_e}(f_e, \gamma^k) = \psi'_e(f_e) - (\gamma_i^k - \gamma_j^k). \tag{2.14}$$

For a given γ^k, the second term is constant, while the first term increases with f_e, since $\psi_e(f_e)$ is chosen as a convex function (implying a non-decreasing derivative). This implies that the derivative of L_e will be increasing as a function of f_e. Two cases will be distinguished, depending on the value of the derivative evaluated at $f_e = 0$. The first case is when

$$\frac{\partial L_e}{\partial f_e}(0, \gamma^k) = \psi'_e(0) - (\gamma_i^k - \gamma_j^k) \geq 0 \quad \Leftrightarrow \quad \psi'_e(0) \geq (\gamma_i^k - \gamma_j^k).$$

Since the derivative is non-negative at $f_e = 0$ and is increasing, L_e will increase for $f_e > 0$. Thus, the minimizing non-negative flow is in this case $f_e^{k+1} = 0$. The second case is when

$$\frac{\partial L_e}{\partial f_e}(0, \gamma^k) = \psi'_e(0) - (\gamma_i^k - \gamma_j^k) < 0 \quad \Leftrightarrow \quad \psi'_e(0) < (\gamma_i^k - \gamma_j^k).$$

Since the derivative is negative at $f_e = 0$ and increasing, it will take the value zero for some $f_e > 0$, which is then the minimizer of L_e. The optimal flow is then obtained by setting the derivative (2.14) equal to zero, yielding

$$f_e^{k+1} = (\psi'_e)^{-1}(\gamma_i^k - \gamma_j^k).$$

In summary, the first step in the dual ascent algorithm, (2.11), is given linkwise for each link $e \in \mathcal{E}$ as

$$f_e^{k+1} := \begin{cases} 0 & \text{if} \quad \psi'_e(0) \geq (\gamma_i^k - \gamma_j^k) \\ (\psi'_e)^{-1}(\gamma_i^k - \gamma_j^k) & \text{if} \quad \psi'_e(0) < (\gamma_i^k - \gamma_j^k) \end{cases}.$$

Also the second step of the dual ascent algorithm, (2.12), can be be computed distributively. By noting that row i of the mass conservation relation (2.3) can be expressed according to (2.2), the dual variable update for node i becomes

$$\gamma_i^{k+1} := \gamma_i^k + \alpha \left(\lambda_i + \sum_{j:(j,i)\in\mathcal{E}} f_{(j,i)} - \sum_{j:(i,j)\in\mathcal{E}} f_{(i,j)} - \mu_i \right). \tag{2.15}$$

Thus, only the in- and outflows to node i are needed in order to compute the associated Lagrange multiplier γ_i^{k+1}.

The mechanism of the dual ascent method can now be understood for a simple example. Assume that the flow vector f equals the optimal flow vector f^* except

for entry e which is smaller, i.e., $f_e < f_e^*$. Then, the mass conservation (2.2) for the start node i of link $e = (i, j)$ is violated such that (2.15) becomes

$$\gamma_i^{k+1} := \gamma_i^k + \underbrace{\alpha}_{>0} \underbrace{(f_e^* - f_e)}_{>0} > \gamma_i^k.$$

Thus, γ_i is increased. In the Lagrangian minimization step (2.11), this implies that in order for the derivative (2.14) to remain zero, if it was zero for the old dual variable values, an increase of γ_i must be compensated by an increase in $\psi_e'(f_e)$. Since $\psi_e(f_e)$ is an increasing convex function, this corresponds to an increase of f_e. Thus, the flow f_e is increased and approaches the optimal flow value f_e^* (if α and thereby the change of γ_i is small enough).

2.5 Implementation of the Dual Ascent Method

As an example, the dual ascent algorithm will be implemented for the network in Fig. 2.1. The node-link incidence matrix is in this case, according to the definition (2.1), given by

$$B = \begin{bmatrix} +1 & +1 & 0 & 0 & 0 \\ -1 & 0 & +1 & +1 & 0 \\ 0 & -1 & -1 & 0 & +1 \\ 0 & 0 & 0 & -1 & -1 \end{bmatrix}.$$

The next step is to choose the cost functions $\psi_e(f_e)$. Assuming that the goal is to minimize the total delay that all the travellers experience in the network, the cost functions should constitute a measure of this delay. Let l_e be the travel time that a vehicle experiences on link e when there is no other traffic, i.e., when $f_e = 0$. Furthermore, let $C_e > 0$ be the **link capacity**, which is the upper limit of the flow that the link can handle, i.e., $f_e < C_e$. Then, a simple model for the delay d_e that a vehicle experiences along the link, as a function of the flow on the link, is

$$d_e(f_e) = \begin{cases} \frac{l_e}{1 - f_e/C_e} & \text{if} \quad 0 \le f_e < C_e \\ +\infty & \text{if} \quad f_e \ge C_e \end{cases}.$$

This delay function is convex and has the properties $d_e(0) = l_e$ and $d_e(f_e) \to +\infty$ as $f_e \to C_e$. The cost function is then chosen as the delay that all vehicles entering the link during one time unit experience together, resulting in

$$\psi_e(f_e) = f_e d_e(f_e) = \begin{cases} \frac{f_e l_e}{1 - f_e/C_e} & \text{if} \quad 0 \le f_e < C_e \\ +\infty & \text{if} \quad f_e \ge C_e \end{cases}. \tag{2.16}$$

The cost function is convex with the properties $\psi_e(0) = 0$ and $\psi_e(f_e) \to +\infty$ as $f_e \to C_e$. Differentiating the cost function (on the interval $f_e \in [0, C_e)$) gives

$$\psi_e'(f_e) = \frac{l_e}{(1 - f_e/C_e)^2}, \quad 0 \le f_e < C_e, \tag{2.17}$$
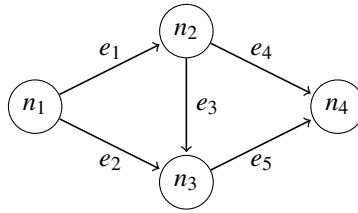
Figure 2.1   Simple road network used for example implementation of the dual ascent algorithm.
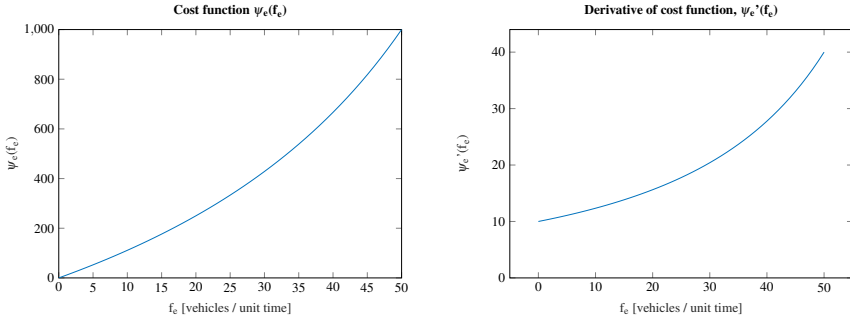
Figure 2.2   Cost function (2.16) and its derivative (2.17) for $l_e = 10$ and $C_e = 100$.

and the inverse function of the derivative is obtained as

$$(\psi_e')^{-1}(x) = C_e \left(1 - \sqrt{\frac{l_e}{x}} \right), \quad x \geq l_e.$$

The cost function and its derivative are illustrated in Fig. 2.2 for $l_e = 10$ and $C_e = 100$.

2.6   Test of Dual Ascent Algorithm

Simulation Results

In the tests, the external flows were set to a unit inflow at node n_1 in Fig. 2.1 and a unit outflow at node n_4, i.e. $\lambda = \hat{e}_1$ and $\mu = \hat{e}_4$. This flow can, as seen in the figure, be distributed between three different paths:

$$p_1 = (e_1, e_4), \quad p_2 = (e_2, e_5) \quad \text{and} \quad p_3 = (e_1, e_3, e_5).$$

The empty-link delay vector was chosen as

$$l = \begin{bmatrix} 1 & 2 & 1 & 2 & 1 \end{bmatrix}^T, \tag{2.18}$$

step size α	1	0.1	0.01	0.001	0.0001
number of iterations	no convergence	65	589	5981	59908
execution time [s]	no convergence	0.05	0.17	1.5	46

Table 2.1 Number of iterations for different step sizes α in the dual ascent algorithm with l given by (2.18), $C_e = 10 \; \forall e \in \mathcal{E}$ and tol $= 10^{-10}$.

Test	1	2	3	4	5	6
$l =$	$\begin{bmatrix} 1 \\ 2 \\ 1 \\ 2 \\ 1 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 2.4 \\ 1 \\ 2.4 \\ 1 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 2.5 \\ 1 \\ 2.5 \\ 1 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 2 \\ 1.1 \\ 2 \\ 1 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 2 \\ 1.2 \\ 2 \\ 1 \end{bmatrix}$	$\begin{bmatrix} 3 \\ 2 \\ 1 \\ 2 \\ 1 \end{bmatrix}$
$f =$	$\begin{bmatrix} 0.5975 \\ 0.4025 \\ 0.1949 \\ 0.4025 \\ 0.5975 \end{bmatrix}$	$\begin{bmatrix} 0.9466 \\ 0.0534 \\ 0.8932 \\ 0.0534 \\ 0.9466 \end{bmatrix}$	$\begin{bmatrix} 1.0000 \\ 0 \\ 1.0000 \\ 0 \\ 1.0000 \end{bmatrix}$	$\begin{bmatrix} 0.5070 \\ 0.4930 \\ 0.0141 \\ 0.4930 \\ 0.5070 \end{bmatrix}$	$\begin{bmatrix} 0.5000 \\ 0.5000 \\ 0 \\ 0.5000 \\ 0.5000 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 1.0000 \\ 0 \\ 0 \\ 1.0000 \end{bmatrix}$
iter.	589	682	1311	607	742	1194
time	0.15 s	0.17 s	0.29 s	0.15 s	0.16 s	0.27 s

Table 2.2 Results for dual ascent algorithm with step size $\alpha = 0.01$, $C_e = 10$ for all $e \in \mathcal{E}$ and tol $= 10^{-10}$.

implying that each of the paths p_1, p_2 and p_3 would give the same delay for an empty network. Furthermore, the link capacities were chosen to be equal: $C_e = 10$ for all $e \in \mathcal{E}$. The tolerance for the size of the duality gap in the stopping criterion was chosen as tol $= 10^{-10}$. Running the algorithm with this setting for different values of the step size α gave the convergence results in Tab. 2.1. In all cases except for $\alpha = 1$, where the algorithm did not converge, the resulting flow vector was

$$f = \begin{bmatrix} 0.5975 & 0.4025 & 0.1949 & 0.4025 & 0.5975 \end{bmatrix}^T,$$

which (approximately) fulfils the feasibility constraints $Bf - \lambda - \mu = 0$.

In the remaining tests, the step size $\alpha = 0.01$ was used. Keeping the link capacity vector as $C_e = 10$ for all $e \in \mathcal{E}$ and the tolerance tol $= 10^{-10}$, the results for different empty-link delay vectors are reported in Tab. 2.2.

Discussion

The tests of different step sizes α show that, for the tested set-up, the same flow vector f is obtained for all values for which the algorithm converges. Convergence is obtained for the tested step sizes which are 0.1 or smaller. The fact that the results are consistent shows that the algorithm seems to be robust with respect to changing the step size, as long as it is small enough for obtaining convergence. The resulting flow vector shows that the largest flow is on the links e_1 and e_5. This is a reasonable

result, since the flow along these links is divided along two links on part of the travel between the nodes n_1 and n_4, following from that each of these links are included in two of the paths. In contrast, the flow along link e_3 is not divided among several links during the travel (it must follow path p_3), whereby it is reasonable that this flow is small.

When increasing the empty-link delays, l_e, along links e_2 and e_4, such that the cost is increased for these links, the result in Test 2 in Tab. 2.2 is obtained. This shows that almost all flow follows path p_3, which is the only path not including the links e_2 and e_4. Increasing the empty-link delays at e_2 and e_4 further, all the flow is assigned to path p_3, as shown by Test 3. Since a large cost at some links should imply that the flow is assigned to paths not including these links, the result is in agreement with what would be expected.

When the empty-link delay instead is increased along link e_3, almost all flow becomes equally distributed between paths p_1 and p_2, which do not contain the link in question. This is shown by Test 4. If the cost is increased further (Test 5), no flow at all is assigned to the path through e_3. Also this result is reasonable, since increasing the cost at a particular link decreases the flow at that link.

Finally, the empty-link delay was increased at link e_1, which is included in paths p_1 and p_3, but not in path p_2. It is then expected that all flow should be assigned to path p_2 if the increased delay is sufficiently large, which is confirmed by Test 6.

The conclusion is that the dual ascent algorithm seems to work fine and give feasible and reasonable results for the cases tested, as long as the step size α is chosen small enough so that convergence is obtained.

2.7 Interpretation of Lagrange Multipliers

Solution methods consisting of simultaneously solving both the primal optimization problem (2.4) and its dual problem (2.8) give not only the link flow vector f, but also the vector of dual variables (Lagrange multipliers) γ. The dual variables can be interpreted as a measure of how much the cost function changes for small changes in the constraints $Bf = \lambda - \mu$, which will be shown in this section. The derivation in this section is based on [Boyd and Vandenberghe, 2009, pp. 249-253] as well as [Whittle, 2007, Ch. 1].

To show this relation, the following function is defined:

$$M(u) \equiv \min_{Bf=u} \psi(f), \quad \text{where} \quad u \equiv \lambda - \mu.$$

This function gives the solution of the primal problem (2.4) for a given right-hand side $u = \lambda - \mu$ in the constraints. Assume that the problem is solved for a given right-hand side $u = u^0$, corresponding to a Lagrangian

$$L_0(f, \gamma) = \psi(f) + \gamma^T (u^0 - Bf),$$

and with the solution $M(u^0)$. This solution will now be compared to the solution that would be obtained if u^0 would have been replaced with $\tilde{u} = u^0 + u^{\varepsilon}$ for some small u^{ε}, constituting a perturbation. In this case, the Lagrangian would become

$$\tilde{L}(f, \gamma) = \psi(f) + \gamma^T (\tilde{u} - Bf),$$

and the solution would be $M(\tilde{u})$. Using that weak duality (2.7) holds for any feasible f and any γ, it follows that

$$\begin{aligned} M(\tilde{u}) &\equiv \min_{Bf=\tilde{u}} \psi(f) \geq \tilde{\theta}(\gamma) \equiv \inf_f \tilde{L}(f, \gamma) \\ &= \inf_f \{L_0(f, \gamma) + \gamma^T u^{\varepsilon}\} = \theta_0(\gamma) + \gamma^T u^{\varepsilon}, \end{aligned} \tag{2.19}$$

where $\tilde{\theta}(\gamma)$ and $\theta_0(\gamma)$ are the dual functions corresponding to the perturbed and unperturbed problem respectively. Since this holds for any γ, the value $\gamma = \gamma^*$, being the optimal dual point for the unperturbed problem, can be chosen. Assuming that strong duality holds, it follows that $\theta_0(\gamma^*) = M(u^0)$. Thus, (2.19) gives

$$M(\tilde{u}) \geq \theta_0(\gamma^*) + (\gamma^*)^T u^{\varepsilon} = M(u^0) + (\gamma^*)^T u^{\varepsilon}$$

or, equivalently,

$$M(u^0 + u^{\varepsilon}) - M(u^0) \geq (\gamma^*)^T u^{\varepsilon}. \tag{2.20}$$

In the sequel, it will be assumed that the function $M(u)$ is differentiable. If the perturbation is chosen as $u^{\varepsilon} = \varepsilon \hat{e}_i$, where $\hat{e}_i$ is a unit basis vector, and $\varepsilon > 0$ is some small number, (2.20) then gives

$$\gamma_i^* \leq \frac{M(u^0 + \varepsilon \hat{e}_i) - M(u^0)}{\varepsilon} \rightarrow \frac{\partial M}{\partial u_i} = (\nabla M)^T \hat{e}_i \quad \text{when} \quad \varepsilon \rightarrow 0.$$

If instead choosing $u^{\varepsilon} = -\varepsilon \hat{e}_i$, the result is

$$\gamma_i^* \geq \frac{M(u^0) - M(u^0 - \varepsilon \hat{e}_i)}{\varepsilon} \rightarrow \frac{\partial M}{\partial u_i} = (\nabla M)^T \hat{e}_i \quad \text{when} \quad \varepsilon \rightarrow 0.$$

From these two results, the conclusion is that if arbitrary perturbations can be added to $u = \lambda - \mu$, then it holds that

$$\gamma_i^* = \frac{\partial M}{\partial u_i} = (\nabla M)^T \hat{e}_i.$$

In words, this means that γ_i^* can be interpreted as the rate of change of the optimal (smallest possible) cost when the external flow to node i is changed infinitesimally. If, e.g., the inflow to node i is changed by a small amount λ_i^{ε}, then the cost would increase by approximately $\lambda_i^{\varepsilon} \gamma_i^*$.

However, since the problem consists of a static flow analysis, it is required that the total external inflow equals the total external outflow, whereby the vector u cannot be chosen arbitrarily, but must fulfil $\mathbf{1}^T u = 0$. Then, the analysis and conclusions above must be modified in order to take this into account. The perturbation is instead chosen as $u^{\varepsilon} = \varepsilon \hat{v}$, with $\hat{v} = \frac{1}{\sqrt{2}}(\hat{e}_i - \hat{e}_j)$, so that $\tilde{u} = u^0 + u^{\varepsilon}$ fulfils the mass conservation $\mathbf{1}^T \tilde{u} = 0$. Then, (2.20) gives

$$\gamma_i^* - \gamma_j^* \leq \sqrt{2}\frac{M(u^0 + \varepsilon\hat{v}) - M(u^0)}{\varepsilon} \to \sqrt{2}\frac{\partial M}{\partial \hat{v}}$$
$$= (\nabla M)^T(\hat{e}_i - \hat{e}_j) \quad \text{when} \quad \varepsilon \to 0.$$

Choosing $u^{\varepsilon} = -\varepsilon\hat{v}$ gives analogously

$$\gamma_i^* - \gamma_j^* \geq \sqrt{2}\frac{M(u^0) - M(u^0 - \varepsilon\hat{v})}{\varepsilon} \to \sqrt{2}\frac{\partial M}{\partial \hat{v}}$$
$$= (\nabla M)^T(\hat{e}_i - \hat{e}_j) \quad \text{when} \quad \varepsilon \to 0.$$

The conclusion from these two inequalities is that

$$\gamma_i^* - \gamma_j^* = (\nabla M)^T(\hat{e}_i - \hat{e}_j) = \frac{\partial M}{\partial u_i} - \frac{\partial M}{\partial u_j}.$$

In the static analysis, the Lagrange multipliers can thus be interpreted as follows: If the inflow is increased by a small amount $\lambda_i^{\varepsilon} = \varepsilon$ at node i, and the outflow is increased by the same amount, $\mu_j^{\varepsilon} = \varepsilon$, at node j, then the cost will increase by approximately $\varepsilon(\gamma_i^* - \gamma_j^*)$. In other words, the difference between two optimal Lagrange multipliers, $\gamma_i^* - \gamma_j^*$, is the rate of change of the minimized cost function when increasing the flow between node i and node j.

2.8   Augmented Lagrangian

As mentioned in Section 2.3, several assumptions about the problem in question must hold in order for the dual ascent method to be applicable. An example of a problem type where the method in general is not applicable is if the objective function $\psi(f)$ is a non-zero affine function (defined for all real numbers). In this case, the first step in the algorithm, (2.11), might fail, since there in general is no finite variable value which solves the minimization problem. A modification of the dual ascent method which is applicable for a larger class of problems and in addition often has better convergence and robustness properties consists of replacing the Lagrangian with a so-called **augmented Lagrangian**, in combination with a specific choice of step size in the dual variable update (2.12).

The augmented Lagrangian for the static network flow optimization problem (2.4) is

$$L_\rho(f, \gamma) = \psi(f) + \gamma^T(u - Bf) + \frac{\rho}{2}\|u - Bf\|_2^2, \quad f \geq 0, \tag{2.21}$$

where $u \equiv \lambda - \mu$ and $\rho > 0$ is a parameter, called the **penalty parameter**, to be chosen. The augmented Lagrangian is equal to the ordinary Lagrangian (2.5) except for the last term, and reduces to the ordinary Lagrangian if setting $\rho = 0$. The augmented Lagrangian can be viewed as the ordinary Lagrangian corresponding to the problem

$$\text{minimize} \quad \psi_{\text{aug}}(f) = \sum_{e \in \mathcal{E}} \psi_e(f_e) + \frac{\rho}{2}\|\lambda - \mu - Bf\|_2^2, \quad f \geq 0$$

$$\text{subject to} \quad Bf = \lambda - \mu \tag{2.22}$$

This problem is equivalent to the original problem (2.4), since for each solution to the both the original problem and to (2.22), $Bf = \lambda - \mu$ according to the constraint, implying that the second term in the objective function $\psi_{\text{aug}}(f)$ is zero, whereby this objective function is equal to the original objective function $\psi(f)$. Thus, proceeding in accordance with the dual ascent method with the augmented Lagrangian (2.21) instead of the original Lagrangian corresponds to solving the problem (2.22), which is equivalent to the original problem (2.4) and thus has the same solution.

For an inner solution, i.e., a solution $f^* > 0$, of the problem (2.4), a necessary condition on the ordinary (non-augmented) Lagrangian $L(f, \gamma)$ is that

$$\nabla L(f, \gamma) = 0 \quad \Leftrightarrow \quad \begin{cases} \nabla_f L(f, \gamma) = \nabla \psi(f) - B^T \gamma &= 0 \\ \nabla_\gamma L(f, \gamma) = \lambda - \mu - Bf &= 0 \end{cases}. \tag{2.23}$$

This implies that the constraint $\lambda - \mu - Bf = 0$ is fulfilled, and that $\nabla \psi(f) - \nabla_f(\lambda - \mu - Bf)^T \gamma = 0$. The latter equation states that the objective gradient, $\nabla \psi(f)$, is parallel to a linear combination of the gradients of all equality constraint functions, $\lambda_i - \mu_i - B_{(:,i)}f$. If this is not the case, the objective function decreases in some feasible direction, which contradicts the fact that the point in question it is a minimum.

If the step size in the dual variable update (2.12) is chosen as $\alpha = \rho$, then the so-called **dual feasibility** $\nabla \psi(f) - B^T \gamma = 0$ in (2.23) is automatically fulfilled for an inner optimal point $f^* > 0$. From the first step in the dual ascent algorithm, (2.11), using the augmented Lagrangian, one gets that $f^{k+1} := \arg\min_f L_\rho(f, \gamma^k)$. This implies that

$$0 = \nabla_f L_\rho(f^{k+1}, \gamma^k) = \nabla \psi(f^{k+1}) - B^T \gamma^k - \rho B^T(\lambda - \mu - Bf^{k+1}). \tag{2.24}$$

Then, with $\alpha = \rho$, the dual update step (2.12) yields

$$\gamma^{k+1} := \gamma^k + \rho(\lambda - \mu - Bf^{k+1}),$$

which inserted into (2.24) gives that

$$\nabla \psi(f^{k+1}) - B^T \gamma^{k+1} = 0.$$

Using the dual ascent method with the augmented Lagrangian (2.21) and the dual variable update step size $\alpha = \rho$ is known as the **method of multipliers**.

For further explanation of the theory in this section, see [Boyd et al., 2011, Ch. 2], from which also the theory presented here was obtained.

2.9 ADMM

An advantage of the dual ascent method is that it can be formulated in a distributed manner, since the Lagrangian (2.13) can be decomposed into one term for each link, and since the dual variable update only requires information about flows on links containing the node corresponding to the variable in question. In the method of multipliers, the Lagrangian is replaced by the augmented Lagrangian, which has the extra term

$$\frac{\rho}{2}\|u - Bf\|_2^2 = \frac{\rho}{2} \sum_{i \in \mathcal{N}} \left(\sum_{j:(j,i) \in \mathcal{E}} f_{ji} - \sum_{j:(i,j) \in \mathcal{E}} f_{ij} + u_i \right)^2,$$

where $u \equiv \lambda - \mu$. This expression contains crossproducts between flows on different links, which prevents a decomposition of the augmented Lagrangian into separate terms for each link. Neither a nodewise decomposition is possible, since each flow variable f_{ij} is associated with both the node i and the node j. A possible method for accomplishing a nodewise decomposition is to use a modified version of the method of multipliers, known as the **alternating direction method of multipliers**, **ADMM**.

ADMM can be used to solve problems on the form

$$\begin{aligned} \text{minimize} \quad & \psi_1(f) + \psi_2(g) \\ \text{subject to} \quad & Ff + Gg = u, \end{aligned} \tag{2.25}$$

for some objective functions ψ_1 and ψ_2, some given matrices F and G, and some given vector u. The set of (primal) variables is here partitioned into the vectors f and g. The only difference between the method of multipliers and ADMM is that the first step of the algorithm, the Lagrangian minimization (2.11), is replaced by two steps. In the first step, the minimization is carried out with respect to only the variables f, while the remaining variables, g, are fixed. In the second step, the variables f are fixed, and the minimization is carried out with respect to the variables g. Thus, the algorithm becomes

$$f^{k+1} := \operatorname*{argmin}_f L_\rho(f, g^k, \gamma^k) \tag{2.26}$$

$$g^{k+1} := \operatorname*{argmin}_g L_\rho(f^{k+1}, g, \gamma^k) \tag{2.27}$$

$$\gamma^{k+1} := \gamma^k + \rho(u - Ff^{k+1} - Gg^{k+1}). \tag{2.28}$$

Notice the difference compared to the method of multipliers. For the case of the method of multipliers, (2.26) and (2.27) would be replaced with $(f^{k+1}, g^{k+1}) := \arg\min_{f,g} L_\rho(f, g, \gamma^k)$, i.e., simultaneous minimization with respect to all primal variables.

This section is based on the description in [Boyd et al., 2011, Ch. 3].

2.10   ADMM-formulation of the Flow Problem

In order for ADMM to solve the problem consisting of the non-decomposability of the augmented Lagrangian, the variables f and g must be chosen appropriately. A useful approach is to let f be the original link flow variables, and to let g represent the same link flows. The idea of this approach is to use f for representation of the inflows to each node, and to use g for representation of the outflows from each node. This will allow for a nodewise decomposition of the derivatives of the augmented Lagrangian used in each Lagrangian minimization step in ADMM, i.e., (2.26) and (2.27). Since f and g represent the same flows, an additional constraint, $f = g$, must be added. Furthermore, the cost functions in (2.25) are chosen as $\psi_1(f) = \psi(f)$ and $\psi_2(g) = 0$, where $\psi(f)$ is the cost function from the original problem formulation (2.4). The ADMM-formulation of the static flow optimization problem thereby becomes

$$
\begin{aligned}
&\underset{f,g}{\text{minimize}} \quad \psi(f,g) = \sum_{e \in \mathcal{E}} \psi_e(f_e), \quad f, g \geq 0, \\
&\text{subject to} \quad \sum_{j:(j,i)\in\mathcal{E}} f_{ji} - \sum_{j:(i,j)\in\mathcal{E}} g_{ij} + u_i = 0, \quad i = 1,\ldots,N, \qquad (2.29) \\
&\qquad\qquad\quad f_e - g_e = 0, \quad e = (i,j) \in \mathcal{E}, \quad e = 1,\ldots,E.
\end{aligned}
$$

This problem is equivalent to the original problem (2.4), since the only difference is that the variables f at some places in the original constraints have been replaced by g, but g is required to be equal to f according to the added constraints.

The augmented Lagrangian for the ADMM-formulation becomes

$$
L_\rho(f,g;\gamma,v) = \sum_{(i,j)\in\mathcal{E}} \psi_{ij}(f_{ij})
$$

$$
+ \sum_{i\in\mathcal{N}} \gamma_i \left(\sum_{j:(j,i)\in\mathcal{E}} f_{ji} - \sum_{j:(i,j)\in\mathcal{E}} g_{ij} + u_i \right) + \sum_{(i,j)\in\mathcal{E}} v_{ij}(f_{ij} - g_{ij})
$$

$$
+ \frac{\rho}{2} \sum_{i\in\mathcal{N}} \left(\sum_{j:(j,i)\in\mathcal{E}} f_{ji} - \sum_{j:(i,j)\in\mathcal{E}} g_{ij} + u_i \right)^2 + \frac{\rho}{2} \sum_{(i,j)\in\mathcal{E}} (f_{ij} - g_{ij})^2, \quad f, g \geq 0.
$$

$$
(2.30)
$$

Note that in this expression, each flow variable in f and g is associated to one node only. The variable f_{ij} is an inflow to node j and is thus associated with node j, while

g_{ij} is an outflow from node i and thus associated with node i. When the expression is differentiated, it will consist of terms $\psi'_{ij}(f_{ij})$, linear terms of the flow variables and constant terms. Thereby, a nodewise decomposition of the augmented Lagrangian derivatives is possible. The ADMM algorithm thus gets the appearance

$$f^{k+1} := \operatorname*{argmin}_{f} L_\rho(f, g^k; \gamma^k, v^k) \tag{2.31}$$

$$g^{k+1} := \operatorname*{argmin}_{g} L_\rho(f^{k+1}, g; \gamma^k, v^k) \tag{2.32}$$

$$\begin{bmatrix} \gamma^{k+1} \\ v^{k+1} \end{bmatrix} := \begin{bmatrix} \gamma^k \\ v^k \end{bmatrix} + \rho \begin{bmatrix} \sum_{j:(j,1)\in\mathcal{E}} f_{j1} - \sum_{j:(1,j)\in\mathcal{E}} g_{1j} + u_1 \\ \vdots \\ \sum_{j:(j,N)\in\mathcal{E}} f_{jN} - \sum_{j:(N,j)\in\mathcal{E}} g_{Nj} + u_N \\ f_1 - g_1 \\ \vdots \\ f_E - g_E. \end{bmatrix} \tag{2.33}$$

In order to update the inflow variables f_{ji} for a node i, only the dual variables v_{ji} associated with the inflow links to the node in question as well as the dual variable γ_i associated with the same node need to be known. Thus, only these variable values need to be communicated to the processing unit that computes the updated flow variables f_{ji}. Analogously, the outflow variables g_{ij} for the node i are updated using only the dual variables v_{ij} associated with the outflow links from i as well as γ_i associated with node i. The update of the dual variable γ_i uses only the flow variables f_{ji} and g_{ij} for in- and outlinks to and from node i respectively. Finally, the update of the dual variable v_{ij} uses the two flow variables f_{ij} and g_{ij} only. Thereby, the algorithm is distributed.

2.11 Lagrangian Minimization

The derivatives of the augmented Lagrangian (2.30), which are used in the Lagrangian minimization steps (2.31) and (2.32), become

$$\frac{\partial L_\rho}{\partial f_{ji}} = \psi'_{ji}(f_{ji}) + \gamma_i + v_{ji} + \rho \left(\sum_\ell f_{\ell i} - \sum_\ell g_{i\ell} + u_i \right) + \rho(f_{ji} - g_{ji}) \tag{2.34}$$

$$\frac{\partial L_\rho}{\partial g_{ij}} = -\gamma_i - v_{ij} - \rho \left(\sum_\ell f_{\ell i} - \sum_\ell g_{i\ell} + u_i \right) - \rho(f_{ij} - g_{ij}). \tag{2.35}$$

In the first Lagrangian minimization step (2.30), the derivatives (2.34) are used. The derivative with respect to f_{ji} depends on the other inflows $f_{\ell i}$ to node i, and can thereby not be decoupled completely as in the dual ascent algorithm. However, since only the inflows to node i and not any other inflows are present in this derivative,

the minimization can be considered for each node separately. Thereby, the algorithm still has distributive properties which should be advantageous for scalability.

In the test implementation of the algorithm, the first Lagrangian minimization step (2.31) is carried out by considering one node i at a time. For node i, all inflows $f_{\ell i}$ except one are assumed to be fixed at the previous value (initially zero). Minimization is then carried out with respect to the non-fixed variable, after which this is considered fixed and the next inflow variable is varied instead, and so on. This consecutive minimization procedure is carried out for all inflow variables. The entire procedure is then iterated until the sum of the changes of the Lagrangian derivatives between consecutive iterations is small. An analogous procedure holds for the second Lagrangian minimization step (2.32), but with all outflows $g_{i\ell}$ from node i considered instead.

2.12 Test of ADMM Algorithm

Simulation Results

The ADMM algorithm was tested with the problem used in 'Test 1' in the dual ascent tests in Section 2.6. To run 100 iterations took about 5 seconds. The resulting values of the cost function and feasibility residual after each iteration, for different values of the penalty parameter ρ, are illustrated in Fig. 2.3. The feasibility residual $\varepsilon_{\text{feas}}$ consists of the sum of the absolute values of the residuals for all constraints, i.e.,

$$\varepsilon_{\text{feas}} = \sum_{i=1}^{N} |B_{(i,:)}f - \lambda_i - \mu_i|.$$

A quantity that in the sequel will be referred to as the 'duality gap' is for the tests shown in Fig. 2.4. This quantity consists of $\psi(f^k) - L(f^k, \gamma^k)$, and is thus not strictly speaking equal to the real duality gap which would be $\psi(f^k) - L(f^k, \gamma^{k-1})$, since the flow variable f^k is determined using the dual variable γ^{k-1}. However, in convergence, $\gamma^k = \gamma^{k-1}$ and since f^k minimizes $L(f, \gamma^{k-1})$ it holds that $L(f^k, \gamma^k) = \theta(\gamma^k)$ and the studied quantity will thereby converge towards the real duality gap. Note that the duality gap may be negative due to infeasible values of f (for feasible values of f, it is always non-negative).

The flow values after each iteration for the different choices of ρ are plotted in Fig. 2.5.

Discussion

The simulations show that the ADMM algorithm can give a reasonably accurate solution of the problem in slightly fewer iterations compared to the dual ascent algorithm, for a good choice of the penalty parameter ρ. However, each iteration in the ADMM requires more computations and can comprise several inner iterations in the f- and g-minimization parts of the algorithm. Thereby, the running time for
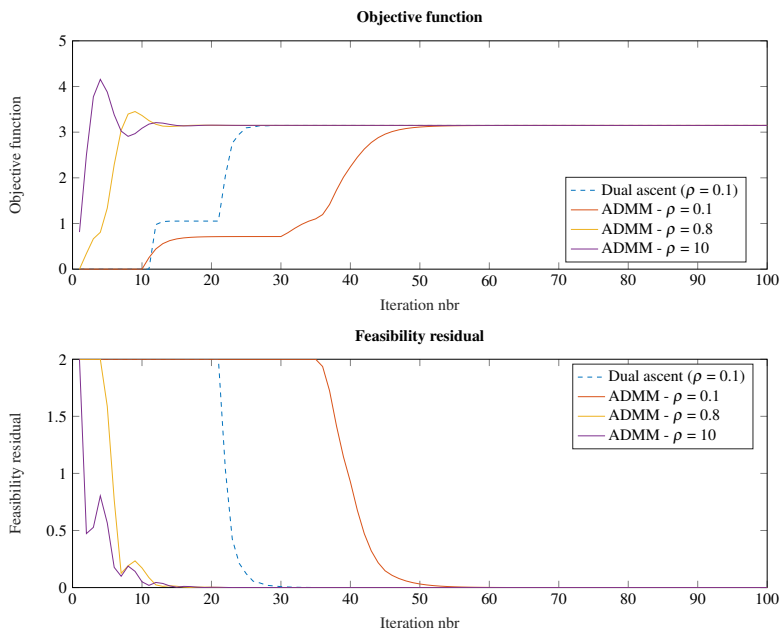
Figure 2.3 Objective function and feasibility residual from the static ADMM algorithm, with different penalty parameter values ρ.
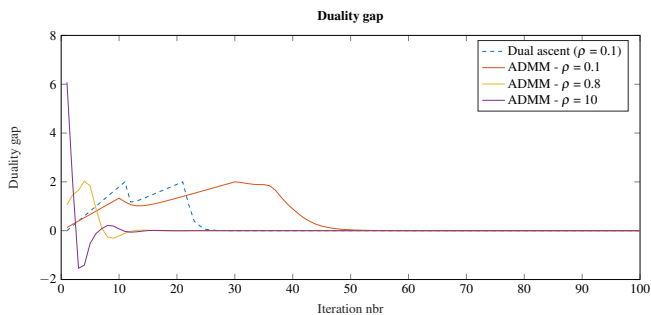
Figure 2.4 Duality gap from the static ADMM algorithm with different penalty parameter values ρ.
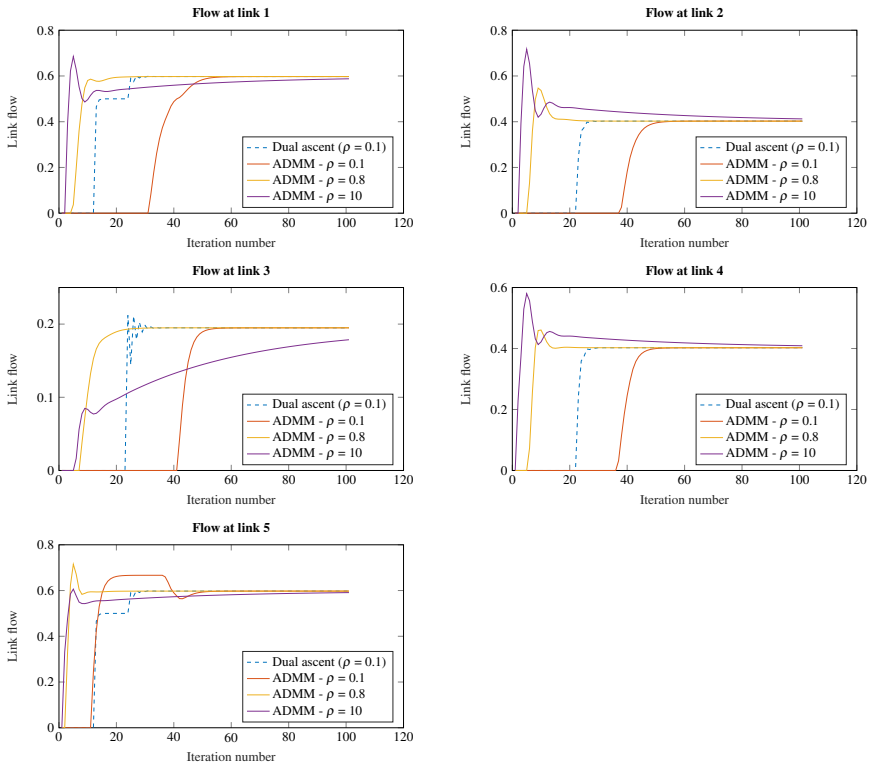
Figure 2.5   Flow variables for static ADMM algorithm with different penalty parameter values ρ.

the ADMM might anyway be greater than for the dual ascent algorithm, depending on how many computations are required in each iteration. If the implementation of each iteration step is efficient, the ADMM however seems to be an efficient method, since the cost function converges approximately to the optimal value in about half the number of iterations for $\rho = 0.8$.

Another advantage of ADMM is that it has improved robustness properties. While the dual ascent algorithm does not converge for the step size $\alpha = 0.8$, the dual ascent algorithm seems to operate fine with the same value of the penalty parameter ρ. Also larger step sizes (penalty parameters) seem to work fine with the ADMM, with the only consequence that the flow variables converge more slowly towards the minimizing values. The better robustness properties can also be seen at, e.g., the flow values at link 3. While the dual ascent algorithm results in fast oscillations before the convergence, the changes of the flow values between iterations are much smoother with ADMM.

In conclusion, the ADMM algorithm for static flow optimization seems to have

31

some advantages compared to the dual ascent algorithm, which can be useful if each iteration of the algorithm can be implemented efficiently.

3

Dynamic Network Flow Optimization

The second part of this project consists of deriving, implementing and studying an algorithm for optimization of traffic flow distribution in a given road network as a function of time. That is, the difference from the previously studied problem is that both the external in- and outflows as well as the traffic densities and flows in the network are allowed to vary with time. This implies that both the distribution of flow between different paths as well as the flow on different parts of a single road can vary. In order to describe this, a more elaborate model than earlier has to be used, which has the ability to describe both temporal and spatial variations.

This chapter contains a description of how dynamic traffic flow can be modelled, as well as a formulation of the problem type that will be studied in the rest of this report. In **Section 3.1**, a description is given of how traffic flow can be modelled in continuous time and space by hydrodynamic models. In **Section 3.2**, the cell transmission model, which is discrete in space and time but works as an approximation of the hydrodynamic model, is described. This is the model that will be used in the optimization. **Section 3.3** describes an alternative way of viewing the flow variables in the cell transmission model. Instead of considering all flow variables, the total outflow from each cell and a routing matrix which distributes this outflow among the outgoing links are considered. This concept will be useful in order to interpret the results from the optimization as ramp metering and speed regulations on the roads. **Section 3.4** describes the problem type that will be considered in the rest of this report. In this section it is also described how the result from the optimization can be interpreted in terms of turning ratios and ramp metering or speed regulations. Finally, **Section 3.5** gives the Lagrangian and the corresponding dual problem for the considered problem.

3.1 Traffic Flow Modelling

In traffic flow modelling, usually two different cases are distinguished: microscopic and macroscopic models [Seibold, 2015]. In microscopic models, the trajectory of each vehicle is described. The problem with this approach is that it, for a human driver, is impossible to describe exactly how the vehicle will move even if the states of all surrounding vehicles are known. For example, the driver's mood and expectations of how the traffic flow will evolve based on earlier experiences are factors that affect how the vehicle will move. Even if it would be possible to approximately model the driver's behaviour, small errors in this model might cause large differences in the evolution of the entire system, since the model is non-linear.

A practically more useful approach is to use macroscopic models. In hydrodynamic models, as the Lighthill-Whitham-Richards model, the traffic flow is described in the same manner as a flow of liquid in a system of pipes. Consider a road with an associated position coordinate ξ on which the traffic density (vehicles per unit length) is given by $\rho(\xi,t)$ at time t. The traffic volume $x_{ab}(t)$ at time t on the part of the road between $\xi = a$ and $\xi = b$ is given by

$$x_{ab}(t) = \int_a^b \rho(\xi,t)\,\mathrm{d}\xi.$$

If the flow (vehicles per time unit) along the road is given by $f(\xi,t)$, then the conservation of vehicles implies that the time derivative of traffic volume on the road segment $[a,b]$ equals the inflow at a minus the outflow at b, i.e.

$$\frac{\mathrm{d}}{\mathrm{d}t}x_{ab}(t) = \int_a^b \frac{\partial}{\partial t}\rho(\xi,t)\,\mathrm{d}\xi = f(a,t) - f(b,t). \tag{3.1}$$

Furthermore, the right-hand side of this expression can according to the fundamental theorem of calculus be written as

$$f(a,t) - f(b,t) = -(f(b,t) - f(a,t)) = -\int_a^b \frac{\partial}{\partial \xi}f(\xi,t)\,\mathrm{d}\xi. \tag{3.2}$$

Combining (3.1) and (3.2) and noting that this is valid for arbitrary values of a and b implies the relation

$$\frac{\partial}{\partial t}\rho(\xi,t) + \frac{\partial}{\partial \xi}f(\xi,t) = 0. \tag{3.3}$$

This is called the **continuity equation** or **flow conservation equation**, and states that no vehicles are created or disappear in any segment.

In order for the continuity equation (3.3) to be able to describe the traffic density $\rho(\xi,t)$ along the road, the flow $f(\xi,t)$ must be related to the density. Empirical studies show that when the density is under a certain limit, the vehicles' velocities are approximately independent of the density. Thus, in this case, the vehicles are assumed to have the free-flow velocity v, giving a flow $f = v\rho$. The flow can increase
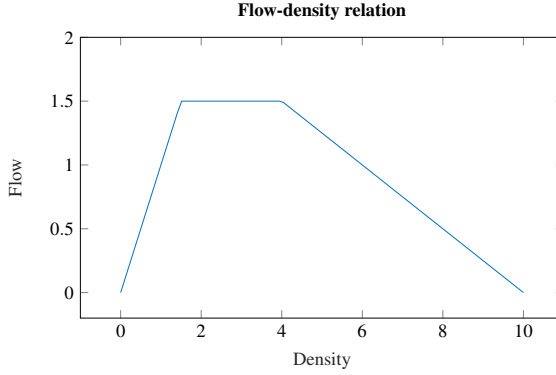
Figure 3.1 Flow f at a specific position ξ and time t at the road as a function of the density ρ according to (3.4) with $v = 1$, $w = 0.25$, $C = 1.5$, $\rho_{jam} = 10$.

until a certain limit, the **maximum flow capacity** C. When this limit is reached, the flow keeps constant for increasing density until the density reaches a certain limit. At this point, the density is large enough to start to slow down the traffic, and thereby decrease the flow. The maximum density that can occur is the **jam density** ρ_{jam}. This density corresponds to a queue where the vehicles do not move, and thus the flow is zero. A simple model for the relation between flow and density with these properties is

$$f(\xi,t) = \min\{v\rho(\xi,t),\ C,\ w(\rho_{jam} - \rho(\xi,t))\}, \tag{3.4}$$

which is illustrated in Fig. 3.1. The parameter $w > 0$ can be viewed as the speed of the congestion wave that propagates backwards when a queue starts to build up at the road. This is usually smaller than the free-flow velocity v of the vehicles. Now, by inserting (3.4) into the flow conservation equation (3.3), the density $\rho(\xi,t)$ can be computed if initial conditions and boundary conditions are given.

3.2   The Cell Transmission Model

In order to numerically compute the traffic density at each point in the road network as a function of time, a method which is discrete in space and time but is a good approximation of the hydrodynamic model should be used. Such a method is the **cell transmission model**, **CTM**, which was originally proposed in [Daganzo, 1994]. This model has in [Como et al., 2016] been used for the study of the same kind of problems as considered in this project.

   In the CTM, a spatial discretization of the road network is used. The discretization is obtained by dividing the roads into segments of a particular length, termed **cells**. An example of this is given in Fig. 3.2. The traffic volume in cell i at time
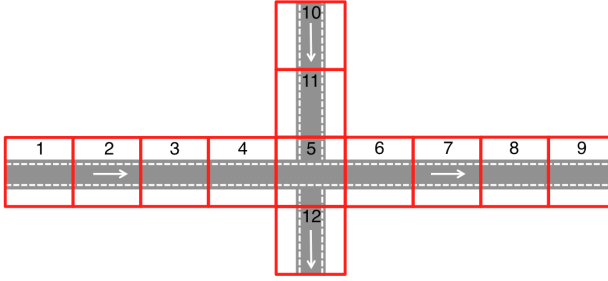
Figure 3.2 Example of spatial CTM-discretization of a road network consisting of two one-way roads. The red boxes correspond to cells. Cell 5 has two inflow links and two outflow links, while the remaining cells have at most one inflow and one outflow link.

t is denoted by $x_i(t)$. Furthermore, the external inflow, i.e., the inflow from roads outside the modelled network, to cell i at time t is denoted by $\lambda_i(t)$ and the external outflow from the cell at time t by $\mu_i(t)$. The flow between cell i and cell j at time t is denoted by $f_{ij}(t)$. All the presented quantities are assumed to be non-negative.

The discretized road network used in the CTM will be represented by a graph where the set of nodes, $\mathcal{N} = \{n_1, \dots, n_N\}$, represent the cells, and the set of links, $\mathcal{E} = \{e_1, \dots, e_E\}$, represent interconnections between cells. Note the difference from the static flow modelling where the links corresponded to roads and the nodes to junctions. Thus, the dynamics in continuous time is given by

$$\dot{x}_i(t) = \lambda_i(t) - \mu_i(t) + \sum_{j:(j,i)\in\mathcal{E}} f_{ji}(t) - \sum_{j:(i,j)\in\mathcal{E}} f_{ij}(t), \quad i = 1, \dots, N, \qquad (3.5)$$

where i refers to cell (node) n_i and N is the total number of cells. This relation is illustrated in Fig. 3.3.

In order to obtain a computer implementable algorithm, also a temporal discretization is needed. The time-derivative is, in accordance with the Euler method, approximated by a forward difference:

$$\dot{x}_i(t_k) \approx \frac{x_i(t_k + h) - x_i(t_k)}{h}, \quad \text{for} \quad t_k = (k-1)h, \quad k = 1, 2, \dots, \qquad (3.6)$$

where $h = t_{k+1} - t_k > 0$ is the chosen time step size. Denoting $x_i(t_k)$, $\lambda_i(t_k)$, $\mu_i(t_k)$ and $f_{ij}(t_k)$ by x_i^k, λ_i^k, μ_i^k and f_{ij}^k thus gives the discretized dynamics

$$x_i^{k+1} = x_i^k + h\left(\lambda_i^k - \mu_i^k + \sum_{j:(j,i)\in\mathcal{E}} f_{ji}^k - \sum_{j:(i,j)\in\mathcal{E}} f_{ij}^k\right), \quad \begin{matrix} k = 1, \dots, K \\ i = 1, \dots, N \end{matrix}, \qquad (3.7)$$

where K is the number of time steps considered, so that the time interval during which x_i is studied has length $T = Kh$. Now, if the initial traffic volumes in the
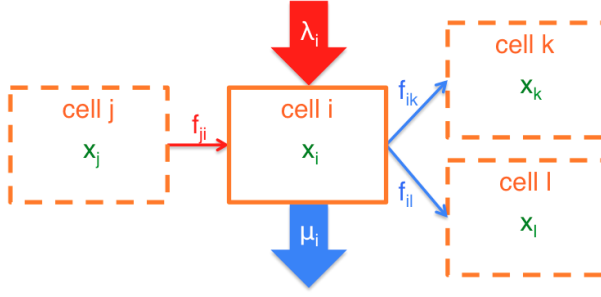
Figure 3.3   The traffic volume x_i in cell i depends on the inflows from from other cells and external inflows (red arrows) as well as outflows to other cells and external outflows (blue arrows), according to (3.5).

cells, x^1, as well as the flows λ^k, μ^k and f^k for times $k = 1, \ldots, K$ are given, the traffic volumes in the cells, x^k, can be computed for times $k = 2, \ldots, K+1$ by the model (3.7).

Except for the dynamics model (3.7), the CTM also includes constraints on the flows, which depend on the current cell traffic volumes. These correspond to the limitations of the flow that were discussed when establishing the flow-density relation (3.4) illustrated in Fig. 3.1.

The total inflow, i.e., the external and internal inflow, to cell i at time k is limited by a **supply function** $s_i(x_i^k)$, which depends on the cell traffic volume at the current time. This function is non-negative, non-increasing and concave. It describes the supply of road capacity that cell i provides when receiving flow. The total outflow from cell i at time k is limited by a **demand function**, $d_i(x_i^k)$. This function is non-decreasing and concave, and fulfils $d_i(0) = 0$. It describes the demand on road capacity that cell i requires when emitting flow. Examples of simple supply and demand functions are illustrated in Fig. 3.4. Thus, the model has the constraints

$$\lambda_i^k + \sum_{j:(j,i)\in\mathcal{E}} f_{ji}^k \leq s_i(x_i^k), \quad k = 1, \ldots, K, \quad i = 1, \ldots, N, \qquad (3.8)$$

$$\mu_i^k + \sum_{j:(i,j)\in\mathcal{E}} f_{ij}^k \leq d_i(x_i^k), \quad k = 1, \ldots, K, \quad i = 1, \ldots, N. \qquad (3.9)$$

The supply constraint implies that the allowed inflow to the cell in question decreases when the traffic volume increases. This is reasonable, since a large amount of traffic on a road segment should increase the congestion in this cell.

The demand constraint assures that the flow from a cell is zero if there is no traffic in the cell, by the property $d_i(0) = 0$ in combination with the non-negativity requirement on the flow variables. Assume that the demand function is linear, as in Fig. 3.4, and has the slope v/L, where L is the length of the cell. Then, the value of the function corresponds to the outflow that would result if the vehicles in the cell
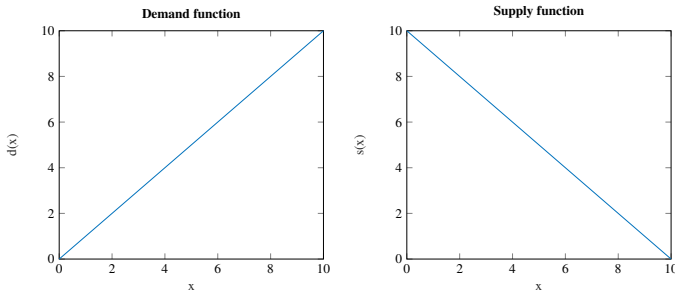
Figure 3.4 Example of demand and supply functions.

are equally distributed and drive at the speed v. Thus, choosing v as the free-flow velocity and assuming that the vehicles never drive faster than this speed naturally implies the demand constraint. A limited maximum flow capacity, corresponding to the flat part of the graph in Fig. 3.1, can be taken into account by replacing the demand function with a modified demand function $\min\{d(x_i^k), C_i^k\}$, where C_i^k is the maximum flow capacity.

The cell lengths L_i in the CTM should be chosen such that they do not exceed the length that a vehicle travels during one time interval h when driving with the free-flow speed v at the road in question, i.e., with the speed that it would have if the traffic density is low. This is necessary in order to get numerical stability of this discetized solution to the originally continuous problem, according to the CFL condition.

When the time step h, and thereby also the cell lengths L_i, go to zero, the CTM gives the same result as the hydrodynamic model, but even when the time steps are larger, the model gives similar and resonable results, as described in [Daganzo, 1994].

3.3   Routing Matrix and Control Parameters

Now that a model for the traffic network and its dynamics is established, the question to be addressed is the following: For given external inflows λ^k, how should the remaining flow variables f^k and μ^k be chosen such that the constraints (3.7), (3.8) and (3.9) are fulfilled? These flow variables are obtained if the total outflow from each cell at each point in time, as well as the distribution of this outflow among the outgoing links is specified. Thus, this is an alternative way to describe the flow distribution.

The distribution of the outflow from cell i is described by the **routing matrix** R. The matrix element $0 \leq R_{ij} \leq 1$ (row i, column j) is the fraction of the total outflow from cell i that goes to cell j at the current time. Thus, the sum of the elements at row i in R is 1 if i is not a sink, i.e., if there is no external outflow from cell
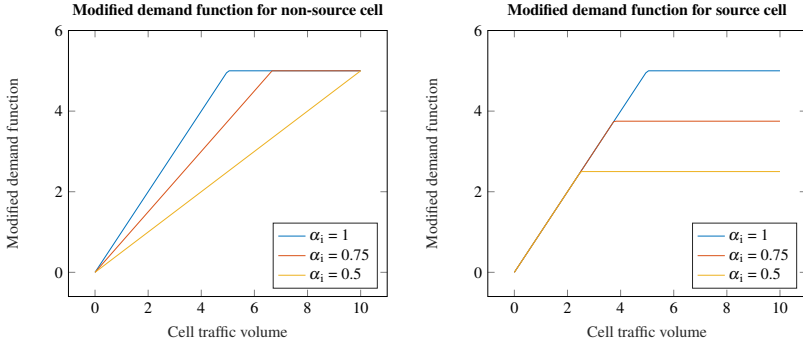
Figure 3.5   Modified demand function $\bar{d}_i(x)$ according to (3.14) for a non-source cell $i \notin \mathcal{R}$ (left) and a source cell $i \in \mathcal{R}$ (right) with the control parameter set to $\alpha_i = 1, 0.75$ and 0.5. The remaining values are chosen as $d_i(x) = x, C_i = 5$.

i. Furthermore, $R_{ij} = 0$ if there is no link between cell i and cell j. Expressed in equations, the routing matrix thereby has the properties

$$0 \leq R_{ij} \leq 1, \quad i, j = 1, \ldots, N, \tag{3.10}$$

$$R_{ij} = 0 \quad \text{if} \quad (i, j) \notin \mathcal{E}, \tag{3.11}$$

$$0 \leq \sum_{j=1}^{N} R_{ij} \leq 1, \quad i = 1, \ldots, N, \tag{3.12}$$

$$\sum_{j=1}^{N} R_{ij} = 1 \quad \forall i \text{ such that } \mu_i = 0. \tag{3.13}$$

In addition to the routing matrix, the outflow from each cell must be known in order to determine the flow variables. The total outflow from cell i is limited by the corresponding demand function $d_i(x_i)$ and by the supply functions $s_j(x_j)$ belonging to the cells connected by the outflow links from cell i according to (3.9) and (3.8). Thus, modifying the demand functions is one way of controlling the outflows from the cells. This is what will be used in the implementation of the optimal control. If the **set of sources**, i.e., the set of cells for which $\lambda_i \neq 0$, is denoted by $\mathcal{R}$, modified demand functions are defined according to

$$\bar{d}_i(x_i, \alpha_i) = \begin{cases} \min\{\alpha_i d_i(x_i), C_i\} & \text{if} \quad i \in \mathcal{E} \setminus \mathcal{R} \\ \min\{d_i(x_i), \alpha_i C_i\} & \text{if} \quad i \in \mathcal{R} \end{cases}, \tag{3.14}$$

where $\alpha_i \in [0, 1]$ are possibly time dependent control parameters, and $C_i \geq 0$ are possibly time dependent maximum flow capacities. These functions are illustrated in Fig. 3.5.

The interpretation of the modified demand functions is as follows: For the non-source cells, $i \in \mathcal{E} \setminus \mathcal{R}$, the parameter α_i changes the speed limit at free flow in the

cell. For a linear demand function $d_i(x)$, an $\alpha_i < 1$ implies a decrease of the slope of the demand function with the corresponding ratio, and the slope is in this case equal to the free-flow speed limit.

For the source cells, $i \in \mathcal{R}$, α_i implies a limitation of the maximum outflow allowed from the cell. This corresponds to implementing a ramp metering on the source cell, so that the number of vehicles that can leave this cell per unit time is limited.

For a given routing matrix R, the outflow from each cell will in the sequel be assumed to be the largest possible flow which satisfies the supply and demand constraints (3.8) and (3.9), where the demand function $d(x)$ is replaced by the modified demand function (3.14). The **total outflow** from cell i, denoted by z_i, will thus be given by

$$z_i = \begin{cases} \mu_i = \bar{d}_i(x_i, \alpha_i) & \text{for} \quad i \in \mathcal{S} \\ \beta_i \bar{d}_i(x_i, \alpha_i) & \text{for} \quad i \in \mathcal{E} \setminus \mathcal{S} \end{cases}, \tag{3.15}$$

where $\mathcal{S}$ is the **set of sink cells**, i.e., cells for which it is allowed to have $\mu_i \neq 0$, and $\beta_i \in [0, 1]$ is a parameter chosen such that, for a given routing matrix R, the outflow from cell i is as large as it can be without violating the supply constraints (3.8) of its out-neighbours. This is mathematically expressed as

$$\beta_i = \sup \left\{ \beta \in [0, 1] : \beta \cdot \max_{j:(i,j)\in\mathcal{E}} \sum_{\ell=1}^{E} R_{\ell j} \bar{d}_\ell(x_\ell, \alpha_\ell) \leq s_j(x_j) \right\}. \tag{3.16}$$

The interpretation of this expression is as follows: For cell i, one of its out-neighbour cells j is considered at a time. For each out-neighbour cell j, the total flow to this cell is given by $\sum_\ell R_{\ell j} z_\ell = \sum_\ell R_{\ell j} \beta_\ell \bar{d}_\ell(x_\ell, \alpha_\ell)$, i.e., the sum of the flows from the in-neighbour cells ℓ to j. This total inflow is not allowed to exceed the supply function $s_j(x_j)$ of cell j. Thus, the coefficient β must be chosen sufficiently small for this to be the case. On the other hand, the flow should be as large as it can be without violating the constraints. Thus, β is chosen so that the total inflow to each of the out-neighbour cells j of i is as large as it possibly can be while also satisfying the corresponding supply constraint. Once the total outflows from the cells, z, are known, the internal flows are given by

$$f_{ij} = R_{ij} z_i, \tag{3.17}$$

according to the definition of the routing matrix.

In conclusion: for given external inflows λ and initial cell traffic volumes x^1, every choice of routing matrix R (with non-negative elements) fulfilling (3.10)–(3.13), and control parameters $\alpha_i \in [0, 1]$, the cell traffic volumes x as well as the flow variables f and μ are given by the CTM according to the equations (3.7)–(3.17).

3.4 Problem Formulation

For every choice of routing matrix R and control parameters α (fulfilling the prescribed constraints), the CTM gives feasible cell traffic volumes and flows when the flows are determined according to the previous subsection. Thus, there are many possible solutions, and the problem to be studied is which one of these solutions should be chosen. In order to compare different solutions, a criterion consisting of a cost function must be chosen. The cost function ψ is assumed to be a convex function depending on x and it is also assumed to be separable with respect to the cells and to fulfil $\psi(0) = 0$. It can thus be written as $\psi(x) = \sum_{i \in \mathcal{N}} \psi_i(x_i)$. The cost considered is the value of this cost function summed over all times. The dynamic traffic assignment problem can thus be stated as

$$
\begin{aligned}
&\underset{\alpha, R}{\text{minimize}} && \sum_{k=1}^{K} \sum_{i=1}^{N} \psi_i(x_i^{k+1}) \\
&\text{subject to} && (3.7), (3.10)\text{--}(3.17) \\
& && x_i^1 = x_i^{\text{init}}, \quad i = 1, \dots, N
\end{aligned}
\tag{3.18}
$$

The problem (3.18) is non-convex, due to the flow limitation equation (3.16). Because of this, the problem is difficult both to study analytically and to find numerical solutions to. However, as proved in [Como et al., 2016], a convex relaxation of this problem can be solved, after which the optimal solution of the convex problem can be mapped to a feasible point of (3.18). The convex problem is

$$
\begin{aligned}
&\underset{x, f, \mu}{\text{minimize}} && \sum_{k=1}^{K} \sum_{i=1}^{N} \psi_i(x_i^{k+1}), \quad x_i^k, f_{ij}^k, \mu_i^k \geq 0, \quad \mu_i^k = 0 \text{ for } i \notin \mathcal{S} \\
&\text{subject to} && x_i^{k+1} = x_i^k + h \left(\lambda_i^k - \mu_i^k + \sum_j f_{ji}^k - \sum_j f_{ij}^k \right), \quad \begin{aligned} k &= 1, \dots, K \\ i &= 1, \dots, N \end{aligned} \\
& && x_i^1 = x_i^{\text{init}}, \quad i = 1, \dots, N \\
& && \lambda_i^k + \sum_{j:(j,i) \in \mathcal{E}} f_{ji}^k \leq s_i(x_i^k), \quad k = 1, \dots, K, \quad i = 1, \dots, N \\
& && \mu_i^k + \sum_{j:(i,j) \in \mathcal{E}} f_{ij}^k \leq d_i(x_i^k), \quad k = 1, \dots, K, \quad i = 1, \dots, N
\end{aligned}
\tag{3.19}
$$

A convex problem is defined as a problem for which the following properties hold: Firstly, if $(x^{(1)}, f^{(1)}, \mu^{(1)})$ and $(x^{(2)}, f^{(2)}, \mu^{(2)})$ both fulfil the constraints in (3.19), then also $(x^{(\beta)}, f^{(\beta)}, \mu^{(\beta)})$, where $x^{(\beta)} = \beta x^{(1)} + (1 - \beta)x^{(2)}$ and so on for any $\beta \in [0, 1]$, will fulfil the constraints. Secondly, it holds that

$$
\Psi(x^{(\beta)}) \leq \beta \Psi(x^{(1)}) + (1 - \beta)\Psi(x^{(2)}), \quad \forall \beta \in [0, 1],
$$

where $\Psi(x) \equiv \sum_{k=1}^{K} \sum_{i=1}^{N} \psi_i(x_i^{k+1})$. The optimization will thus consist of solving the problem (3.19).

When the solution of the convex problem (3.19) is obtained, this can be mapped to a feasible point of the original problem (3.18), according to Proposition 1 in [Como et al., 2016]. For a solution (x, f, μ), the corresponding values of the routing matrix R and control parameters α are

$$
\alpha_i^k = \begin{cases} z_i^k/d_i(x_i^k), & i \notin \mathcal{R}, \ d_i(x_i^k) > 0 \\ 1, & i \notin \mathcal{R}, \ d_i(x_i^k) = 0 \\ z_i^k/C_i^k, & i \in \mathcal{R} \end{cases} ,
$$

$$
R_{ij}^k = \begin{cases} f_{ij}^k/z_i^k, & (i,j) \in \mathcal{E}, \ z_i^k > 0 \\ 1/|\{k \in \mathcal{E} : (i,k) \in \mathcal{E}\}|, & (i,j) \in \mathcal{E}, \ z_i^k = 0 \\ 0, & (i,j) \notin \mathcal{E} \end{cases} ,
$$

where $z_i^k = \mu_i^k + \sum_j f_{ij}^k$.

3.5 Lagrangian and Dual Problem

As in the static case, a Lagrangian will be used to solve the convex optimization problem (3.19). The most significant difference from the static problem (2.4) is that the dynamic problem not only has equality constraints, but also inequality constraints. The inequality constraints are treated in the same manner as the equality constraints when establishing the ordinary Lagrangian. The Lagrangian for the problem (3.19) thus reads

$$
L(x, f, \mu; \gamma, \xi, \eta) = \sum_{k=1}^{K} \sum_{i=1}^{N} \psi_i(x_i^{k+1})
$$

$$
+ \sum_{k=1}^{K} \sum_{i=1}^{N} \gamma_i^k \left(x_i^{k+1} - x_i^k - h \left(\lambda_i^k - \mu_i^k + \sum_j f_{ji}^k - \sum_j f_{ij}^k \right) \right)
$$
(3.20)

$$
+ \sum_{k=1}^{K} \sum_{i=1}^{N} \xi_i^k \left(\lambda_i^k + \sum_j f_{ji}^k - s_i(x_i^k) \right) + \sum_{k=1}^{K} \sum_{i=1}^{N} \eta_i^k \left(\mu_i^k + \sum_j f_{ij}^k - d_i(x_i^k) \right),
$$

$$
x_i^k, f_{ij}^k, \mu_i^k \geq 0 \ \forall i, \quad x_i^1 = x_i^{\text{init}} \ \forall i, \quad \mu_i^k = 0 \text{ for } i \notin \mathcal{S}.
$$

Note that each term corresponding to an inequality constraint is obtained by writing the constraint on the form $\varphi \leq 0$ and then multiplying the expression corresponding to φ with a Lagrange multiplier ξ_i^k or η_i^k.

The dual function is defined as

$$
\theta(\gamma, \xi, \eta) = \inf_{x, f, \mu} L(x, f, \mu; \gamma, \xi, \eta).
$$
(3.21)
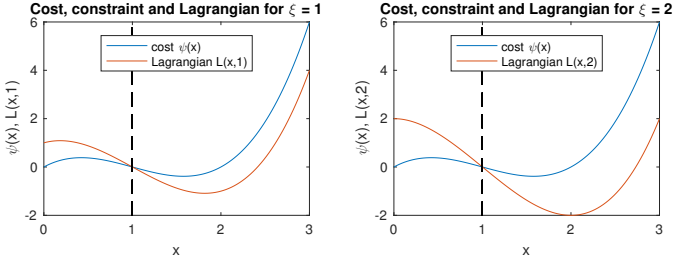
Figure 3.6 For the problem with cost function $\psi(x)$ and constraint $\varphi(x) = 1 - x \leq 0$, it holds for the Lagrangian $L(x; \xi) = \psi(x) + \xi \varphi(x)$ for any $\xi \geq 0$ that $L(x; \xi) \leq \psi(x)$ for all *feasible x*-values $x \geq 1$.

If the dual variables ξ and η are non-negative, then **weak duality** holds. This means that for all feasible variable values x, f and μ, it holds that

$$\theta(\gamma, \xi, \eta) \leq \psi(x, f, \mu) = \sum_{k=1}^{K} \sum_{i=1}^{N} \psi_i(x_i^{k+1}). \qquad (3.22)$$

This follows from that, for each set of fixed dual variables γ, $\xi \geq 0$ and $\eta \geq 0$, the Lagrangian is not less than the dual function, according to the definition (3.21). The Lagrangian, in turn, is always less than or equal to the cost function for feasible variable values. This follows from that the equality constraint terms are zero, while the inequality constraint terms consist of a non-negative dual variable (ξ_i^k or η_i^k) multiplied with the left-hand side φ of the inequality constraint, which is less than or equal to zero. An example of this is shown in Fig. 3.6. Expressed in equations:

$$\theta(\gamma, \xi, \eta) \leq L(x, f, \mu; \gamma, \xi, \eta) \leq \sum_{k=1}^{K} \sum_{i=1}^{N} \psi_i(x_i^{k+1})$$

$$\text{if} \quad x, f, \mu \text{ feasible} \quad \text{and} \quad \xi, \eta \geq 0.$$

Thus, for any choice of $\gamma \in \mathbb{R}^N$ and $\xi, \eta \geq 0$, the dual function $\theta(\gamma, \xi, \eta)$ constitutes a lower bound to the solution of the optimization problem (3.19). The implication of this is that the solution of the problem (3.19) has to be greater than or equal to the the solution of the dual problem

$$\begin{aligned} \underset{\gamma, \xi, \eta}{\text{maximize}} \quad & \theta(\gamma, \xi, \eta) \\ \text{subject to} \quad & \xi \geq 0, \quad \eta \geq 0 \end{aligned} \qquad (3.23)$$

The problem to be solved, (3.19), is a convex problem. Convex problems usually have **strong duality**, meaning that the solution of the dual problem (3.23) not only is less than or equal to the solution of the primal problem (3.19) (which always is

the case), but that the solutions coincide. This can be formally proved by checking constraint qualifications, and will in the sequel be assumed to hold.

The theoretical content in this subsection is obtained from [Boyd and Vandenberghe, 2009, Ch. 5].

4

Distributed Solution of Dynamic Flow Optimization

In this chapter, a distributed algorithm for solving the problem described in the previous chapter is derived. In **Section 4.1**, a reformulation of the dynamic flow optimization problem (3.19) is done, such that it can be solved in a distributed manner. This formulation is inspired by the ADMM algorithm used for the static flow optimization. After this, a solution algorithm is presented, which also is inspired by the static ADMM optimization. In **Section 4.2**, a detailed description with all necessary calculations is given of how the first step of the suggested algorithm can be carried out in a distributed manner. This step consists of minimizing the augmented Lagrangian with respect to the primal varialbes. By making some assumptions about the problem, explicit expressions are obtained for this solution procedure, which allows for an efficient implementation. Finally, **Section 4.3** describes the stopping criterion that is used for determining when the iterative scheme should be stopped.

4.1   Distributed Formulation of the Dynamic Problem

The idea behind the augmented Lagrangian, as described for the static case in Section 2.8, is to add a penalty term for each constraint to the objective function. This term is zero for feasible variable values and greater than zero if the constraint is violated. For the equality constraints present in the static case, the penalty term was chosen as the square of the constraint residual (multiplied with a constant). Generalizing this to the case of inequality constraints, the extra term should be zero when the constraint in question is satisfied and the square of the residual (times a constant) when it is not. Thus, for each constraint on the form $\varphi \leq 0$, a term $(\rho/2)(\max\{0,\varphi\})^2$ is added to the Lagrangian.

The square terms in the augmented Lagrangian prevent both linkwise and cellwise decomposability of this function, as in the static case. This is again solved by the same approach, consisting of adding a separate set of variables g which are required to be equal to f in optimality. The f-variables represent the inflows to each

cell, and the g-variables represent the outflows from each cell. By first minimizing the Lagrangian with respect to the f-variables and then with respect to the variables g, each cell can be considered separately in each of these two steps. Since the remaining variables, x_i^k and μ_i^k, belong only to one cell each, and the Lagrangian derivatives are affine functions of these, a cellwise decomposition can also be used in the Lagrangian minimization with respect to these.

By the same approach as in the static case, a decomposition of the Lagrangian minimization steps with respect to the cells (nodes) is thus possible to obtain. However, in the dynamic case it is also desirable to decouple the Lagrangian derivatives with respect to time, in order to get minimization problems of small dimensions which are easier so solve. The coupling of variables for different time points is a result of the mass conservation constraints in (3.19), which contain both x_i^{k+1} and x_i^k. These constraints relate the current cell traffic volume to the volume at the next time point. As for the flow decomposition, a time decomposition is obtained by introducing a new set of variables y_i^k, $k = 1, \ldots, K$, which in this case represent the cell volume one step ahead at time k. These variables are thereafter required to fulfil $y_i^k = x_i^{k+1}$ for $k = 1, \ldots, K-1$. The point of this is that the variables y_i^k $(k = 1, \ldots, K)$ can replace the variables x_i^{k+1} $(k = 1, \ldots, K)$ in the mass conservation constraints, after which these constraints only contain variables associated to time k. The equalities $y_i^k = x_i^{k+1}$ are then assured to hold by adding these as new constraints.

The distributed formulation of the problem (3.19) thus becomes

$$
\begin{aligned}
&\underset{x,y,f,g,\mu}{\text{minimize}} \quad \sum_{k=1}^{K} \sum_{i=1}^{N} \psi_i(y_i^k), \quad x_i^k, y_i^k, f_{ij}^k, g_{ij}^k, \mu_i^k \geq 0 \; \forall i, \quad \mu_i^k = 0 \text{ for } i \notin \mathcal{S} \\
&\text{subject to} \quad y_i^k = x_i^k + h\left(\lambda_i^k - \mu_i^k + \sum_j f_{ji}^k - \sum_j g_{ij}^k\right), \quad \begin{array}{l} k = 1, \ldots, K \\ i = 1, \ldots, N \end{array} \\
&\qquad\qquad f_{ij}^k = g_{ij}^k, \quad k = 1, \ldots, K, \quad (i,j) = e = 1, \ldots, E \\
&\qquad\qquad x_i^{k+1} = y_i^k, \quad k = 1, \ldots, K-1, \quad i = 1, \ldots, N \\
&\qquad\qquad x_i^1 = x_i^{\text{init}}, \quad i = 1, \ldots, N \\
&\qquad\qquad \lambda_i^k + \sum_{j:(j,i)\in\mathcal{E}} f_{ji}^k \leq s_i(x_i^k), \quad k = 1, \ldots, K, \quad i = 1, \ldots, N \\
&\qquad\qquad \mu_i^k + \sum_{j:(i,j)\in\mathcal{E}} g_{ij}^k \leq d_i(x_i^k), \quad k = 1, \ldots, K, \quad i = 1, \ldots, N
\end{aligned}
$$

$$(4.1)$$

This differs form the original problem formulation (3.19) by that f_{ij}^k have been replaced with g_{ij}^k for the outflows from cell i, and that the extra constraints $f_{ij}^k = g_{ij}^k$ have been added. Furthermore, x_i^{k+1} have been replaced by y_i^k while the constraints $x_i^{k+1} = y_i^k, k = 1, \ldots, K-1$ have been added. Since x_i^{k+1} does not appear explicitly in the equations any more, it is enough to consider x_i^k for $k = 1, \ldots, K$ and then set x_i^{k+1} to y_i^k in the final answer. The augmented Lagrangian for the distributed formulation

(4.1) becomes

$$L_\rho(x,y,f,g,\mu;\gamma,\nu,\sigma,\xi,\eta) = \sum_{k=1}^{K}\sum_{i=1}^{N}\psi_i(y_i^k)$$

$$+ \sum_{k=1}^{K}\sum_{i=1}^{N}\gamma_i^k\left(y_i^k - x_i^k - h\left(\lambda_i^k - \mu_i^k + \sum_{\ell}f_{\ell i}^k - \sum_{\ell}g_{i\ell}^k\right)\right)$$

$$+ \sum_{k=1}^{K}\sum_{e=1}^{E}\nu_e^k(f_e^k - g_e^k) + \sum_{k=1}^{K-1}\sum_{i=1}^{N}\sigma_i^k(y_i^k - x_i^{k+1})$$

$$+ \sum_{k=1}^{K}\sum_{i=1}^{N}\xi_i^k\left(\lambda_i^k + \sum_{\ell}f_{\ell i}^k - s_i(x_i^k)\right) + \sum_{k=1}^{K}\sum_{i=1}^{N}\eta_i^k\left(\mu_i^k + \sum_{\ell}g_{i\ell}^k - d_i(x_i^k)\right)$$

$$+ \frac{\rho}{2}\sum_{k=1}^{K}\sum_{i=1}^{N}\left(x_i^{k+1} - x_i^k - h\left(\lambda_i^k - \mu_i^k + \sum_{\ell}f_{\ell i}^k - \sum_{\ell}g_{i\ell}^k\right)\right)^2 \qquad (4.2)$$

$$+ \frac{\rho}{2}\sum_{k=1}^{K}\sum_{e=1}^{E}(f_e^k - g_e^k)^2 + + \frac{\rho}{2}\sum_{k=1}^{K-1}\sum_{i=1}^{N}(y_i^k - x_i^{k+1})^2$$

$$+ \frac{\rho}{2}\sum_{k=1}^{K}\sum_{i=1}^{N}\left(\max\left\{0,\lambda_i^k + \sum_{\ell}f_{\ell i}^k - s_i(x_i^k)\right\}\right)^2$$

$$+ \frac{\rho}{2}\sum_{k=1}^{K}\sum_{i=1}^{N}\left(\max\left\{0,\mu_i^k + \sum_{\ell}g_{i\ell}^k - d_i(x_i^k)\right\}\right)^2,$$

$$x_i^k, y_i^k, f_e^k, g_e^k, \mu_i^k \geq 0, \quad x_i^1 = x_i^{\text{init}}, \quad \mu_i^k = 0 \text{ for } i \notin \mathcal{S}.$$

The solution procedure is executed analogously to the ADMM algorithm used in the static case. First, the Lagrangian is minimized with respect to the inflows f, then it is minimized with respect to the outflows g, the external outflows μ, the future cell volumes y and the cell volumes x. Thus, the Lagrangian minimization is carried out in five different steps, instead of the exact minimization in the method of multipliers that would require simultaneous minimization with respect to all variables. After this, the dual variables are updated by taking a step in the direction of the dual function gradient. An important difference from the equality-constrained static problem is that there now are some dual variables which have to be non-negative, according to the dual problem formulation (3.23). If the dual function gradient component in some ξ_i^k- or η_i^k-direction is such that the updated value of this dual variable would be less than zero, the new variable value is thereby instead set to zero. The geometrical interpretation of this is that the dual function in the direction of, e.g. ξ_i^k, increases in the negative direction at $\xi_i^k = 0$. The best choice of the dual variable, with respect to maximizing the dual function, which fulfils the non-negativity constraints in (3.23), is thus $\xi_i^k = 0$. After these considerations, the distributed algorithm for solution of the dynamic optimal control problem (4.1)

becomes

$$f^{(n+1)} := \operatorname*{argmin}_{f} L_\rho(x^{(n)}, y^{(n)}, f, g^{(n)}, \mu^{(n)}; \gamma^{(n)}, \nu^{(n)}, \sigma^{(n)}, \xi^{(n)}, \eta^{(n)}) \tag{4.3}$$

$$g^{(n+1)} := \operatorname*{argmin}_{g} L_\rho(x^{(n)}, y^{(n)}, f^{(n+1)}, g, \mu^{(n)}; \gamma^{(n)}, \nu^{(n)}, \sigma^{(n)}, \xi^{(n)}, \eta^{(n)}) \tag{4.4}$$

$$\mu^{(n+1)} := \operatorname*{argmin}_{\mu} L_\rho(x^{(n)}, y^{(n)}, f^{(n+1)}, g^{(n+1)}, \mu; \gamma^{(n)}, \nu^{(n)}, \sigma^{(n)}, \xi^{(n)}, \eta^{(n)}) \tag{4.5}$$

$$y^{(n+1)} := \operatorname*{argmin}_{y} L_\rho(x^{(n)}, y, f^{(n+1)}, g^{(n+1)}, \mu^{(n+1)}; \gamma^{(n)}, \nu^{(n)}, \sigma^{(n)}, \xi^{(n)}, \eta^{(n)}) \tag{4.6}$$

$$x^{(n+1)} := \operatorname*{argmin}_{x} L_\rho(x, y^{(n+1)}, f^{(n+1)}, g^{(n+1)}, \mu^{(n+1)}; \gamma^{(n)}, \nu^{(n)}, \sigma^{(n)}, \xi^{(n)}, \eta^{(n)})$$
$$\tag{4.7}$$

$$\begin{cases} (\gamma_i^k)^{(n+1)} := (\gamma_i^k)^{(n)} + \rho\left(y_i^k - x_i^k - h\left(\lambda_i^k - \mu_i^k + \sum_\ell f_{\ell i}^k - \sum_\ell g_{i\ell}^k\right)\right) \\ (\nu_e^k)^{(n+1)} := (\nu_e^k)^{(n)} + \rho(f_e^k - g_e^k) \\ (\sigma_i^k)^{(n+1)} := (\sigma_i^k)^{(n)} + \rho(y_i^k - x_i^{k+1}) \\ (\xi_i^k)^{(n+1)} := \max\left\{0, (\xi_i^k)^{(n)} + \rho\left(\lambda_i^k + \sum_\ell f_{\ell i}^k - s_i(x_i^k)\right)\right\} \\ (\eta_i^k)^{(n+1)} := \max\left\{0, (\eta_i^k)^{(n)} + \rho\left(\mu_i^k + \sum_\ell g_{i\ell}^k - d_i(x_i^k)\right)\right\} \end{cases} \tag{4.8}$$

where $f^{(n)}$ symbolizes the values of f_e^k for $e = 1, \dots, E$ and $k = 1, \dots, K$ at algorithm iteration number n, and analogously for the other variables where (n) is used.

In the iterative algorithm, the primal variables x, y, f, g and μ are initiated such that the initial solution is feasible. This is accomplished by choosing the flow variables according to the flow rule (3.15)–(3.17) described in conjunction with the CTM, with the routing matrix R chosen such that the flow form a cell is distributed equally among its out links and with all parameters α_i chosen as $\alpha_i = 1$. All dual variables, γ, ν, σ, ξ and η are initiated as zero.

4.2 Lagrangian Minimization

Assumptions and Limitations

In order to carry out the Lagrangian minimizations (4.3)–(4.7), the derivatives of the augmented Lagrangian (4.2) with respect to all primal variables are used. If some assumptions about the problem are made, these derivatives become affine functions of the sought variables, which makes it possible to find explicit expressions for the minimizing variable values. The following assumptions are made:

- The cost functions ψ_i are chosen as either linear or quadratic functions of the cell traffic volumes, i.e., $\psi_i(x) = x$ or $\psi_i(x) = x^2$.

- The supply and demand functions, $s_i(x)$ and $d_i(x)$, are affine.

However, in the last subsubsection, a modification will be described which allows the supply and demand functions to be affine with a capacity limit, i.e., piecewise affine with two parts, where the largest part has slope zero.

Except for these assumptions, a third assumption will be made in the calculations:

- The number of inflow links as well as the number of outflow links from each cell do not exceed two.

This simplifies the explicit calculations, but the exact same procedure should be possible to extend to a larger number of in- and outflow links without any problem.

Minimization With Respect To f

Differentiating the augmented Lagrangian (4.2) with respect to f_{ji}^k gives

$$
\begin{aligned}
\frac{\partial L_\rho}{\partial f_{ji}^k} = {} & -h\gamma_i^k + v_{ji}^k + \xi_i^k \\
& - h\rho \left(y_i^k - x_i^k - h \left(\lambda_i^k - \mu_i^k + \sum_\ell f_{\ell i}^k - \sum_\ell g_{i\ell}^k \right) \right) + \rho \left(f_{ji}^k - g_{ji}^k \right) \\
& + \rho \max \left\{ 0, \lambda_i^k + \sum_\ell f_{\ell i}^k - s_i(x_i^k) \right\}.
\end{aligned}
\tag{4.9}
$$

The aim is to minimize the augmented Lagrangian with respect to all inflow variables f_{ji}^k in the domain $f_{ji}^k \geq 0$. From the derivatives (4.9) it can be concluded that each element of the gradient $\nabla_f L_\rho$ only contains inflow variables f_{ji}^k which are inflows to a particular cell i at a particular time k. Thereby, a decomposition in time and a cellwise decomposition can be accomplished for the purpose of the minimization, as desired. The subproblems to be solved are thus

- For each time point $k = 1, \ldots, K$ and cell $i = 1, \ldots, N$:

$$
\underset{f_{ji}^k \in \mathcal{F}_i^{\text{in}}}{\text{minimize }} L_\rho, \quad \text{where} \quad \mathcal{F}_i^{\text{in}} = \{f_{ji}^k : (j,i) \in \mathcal{E}\}.
\tag{4.10}
$$

Solving all these problems separately is equivalent to minimizing the augmented Lagrangian with respect to all f-variables simultaneously (i.e. (4.3)), due to the separable structure of the augmented Lagrangian.

Furthermore, since it is assumed that each cell has at most two inflow links, each subproblem (4.10) in the minimization has at most two unknown variables. Thus, explicit expressions for the minima can be derived for the case of one inflow link and for the case of two inflow links. In order to handle the last term in the derivatives

49

(4.9), each of these cases is in turn divided into two cases, depending on whether $\lambda_i^k + \sum_\ell f_{\ell i}^k - s_i(x_i)$ is less than or equal to, or greater than zero.

For the case of one inflow link, (j,i), to the considered cell i, the minimization subproblem is one-dimensional. Since the augmented Lagrangian is a convex function of the variables f_{ji}^k, the minimization on the domain $f_{ji}^k \geq 0$ can be carried out as follows:

- If $\frac{\partial L_\rho}{\partial f_{ji}^k}(0) \geq 0$, i.e., the derivative (4.9) evaluated at zero is non-negative, then the convexity implies that the derivative and thus also the function will increase for larger values of the variable f_{ji}^k. Thereby, the minimizer on the considered domain has to be $f_{ji}^k = 0$.

- If instead $\frac{\partial L_\rho}{\partial f_{ji}^k}(0) < 0$, then the convexity implies that the minimum, i.e., where the Lagrangian derivative is zero, must be attained for $f_{ji}^k > 0$.

In conclusion, the minimizing variable value can be obtained by setting the derivative (4.9) equal to zero, solving for f_{ji}^k and concluding that $\max\{0, f_{ji}^k\}$ is the minimizer on the domain $f_{ji}^k \geq 0$.

Proceeding according to this, by setting the derivative (4.9) to zero and solving for the inflow variable to cell i at time k, the minimizing flow on the domain $f_{ji}^k \geq 0$ is obtained as:

- For one inflow (f_{ji}^k) and $\lambda_i^k + \sum_\ell f_{\ell i}^k - s_i(x_i) > 0$:

$$\frac{\partial L_\rho}{\partial f_{ji}^k} = -b + a f_{ji}^k, \quad \text{where}$$

$$b = h\gamma_i^k - v_{ji}^k - \xi_i^k + h\rho\left(y_i^k - x_i^k - h\left(\lambda_i^k - \mu_i^k - \sum_\ell g_{i\ell}^k\right)\right)$$
$$+ \rho g_{ji}^k - \rho\left(\lambda_i^k - s_i(x_i^k)\right),$$
$$a = (h^2 + 2)\rho.$$

Minimizing flow: $f_{ji}^k = \max\{0, b/a\}$.

- For one inflow (f_{ji}^k) and $\lambda_i^k + \sum_\ell f_{\ell i}^k - s_i(x_i) \leq 0$:

$$\frac{\partial L_\rho}{\partial f_{ji}^k} = -b + a f_{ji}^k, \quad \text{where}$$

$$b = h\gamma_i^k - v_{ji}^k - \xi_i^k + h\rho\left(y_i^k - x_i^k - h\left(\lambda_i^k - \mu_i^k - \sum_\ell g_{i\ell}^k\right)\right) + \rho g_{ji}^k,$$

$$a = (h^2 + 1)\rho.$$

Minimizing flow: $f_{ji}^k = \max\{0, b/a\}$.

Since it is not possible to determine the sign of the condition $\lambda_i^k + \sum_\ell f_{\ell i}^k - s_i(x_i)$ a priori, it is necessary to for each of the two cases above insert the solution candidate f_{ji}^k into the assumed condition $\lambda_i^k + \sum_\ell f_{\ell i}^k - s_i(x_i) > 0$ or $\lambda_i^k + \sum_\ell f_{\ell i}^k - s_i(x_i) \le 0$ respectively, to see whether this is fulfilled. If the assumed condition is not fulfilled, the solution candidate is discarded. After this, the solution candidate corresponding to the smallest Lagrangian is chosen.

For the case of two inflow links, (j_1, i) and (j_2, i), to cell i, the minimization problem with respect to the inflows to cell i at time k is two-dimensional. The minimum on the domain $(f_{j_1 i}^k, f_{j_2 i}^k) \ge 0$ is then either an inner point in the first quadrant of $\mathbb{R}^2$ or a boundary point, which is a point at the non-negative part of one of the axes in $\mathbb{R}^2$. Thus, all possible minima can be found by identifying inner points and boundary points which satisfy the necessary conditions for being minima.

Solution candidates along the boundary are found by first assuming that one of the two flows is zero and proceeding according to the described method for the single inflow case to find the second flow variable which gives the minimum along the non-negative part of the corresponding axis. After this, the other flow variable is assumed to be zero while the first flow variable is chosen so that it corresponds to the minimum along the non-negative part of the corresponding axis. These boundary minima are saved as possible solution candidates.

For an inner point to be a minimum, the derivatives with respect to both flow variables have to be zero. Thus, points satisfying this are collected, after that it has been checked that they are inside the domain $(f_{j_1 i}^k, f_{j_2 i}^k) > 0$. By using (4.9), the solution candidates corresponding to inner points are thereby obtained as:

- For two non-zero inflows $(f_{j_1 i}^k, f_{j_2 i}^k)$ and $\lambda_i^k + \sum_\ell f_{\ell i}^k - s_i(x_i) > 0$:

$$\frac{\partial L_\rho}{\partial f_{j_1 i}^k} = -b_1 + a_{11} f_{j_1 i}^k + a_{12} f_{j_2 i}^k \quad \text{where}$$

$$b_1 = h\gamma_i^k - v_{j_1 i}^k - \xi_i^k + h\rho \left(y_i^k - x_i^k - h \left(\lambda_i^k - \mu_i^k - \sum_\ell g_{i\ell}^k \right) \right)$$

$$+ \rho g_{j_1 i}^k - \rho \left(\lambda_i^k - s_i(x_i^k) \right),$$

$$a_{11} = (h^2 + 2)\rho, \quad a_{12} = (h^2 + 1)\rho.$$

$$\frac{\partial L_\rho}{\partial f^k_{j_2 i}} = -b_2 + a_{21} f^k_{j_2 i} + a_{22} f^k_{j_2 i} \quad \text{where}$$

$$b_2 = h\gamma^k_i - v^k_{j_2 i} - \xi^k_i + h\rho \left(y^k_i - x^k_i - h \left(\lambda^k_i - \mu^k_i - \sum_\ell g^k_{i\ell} \right) \right)$$

$$+ \rho g^k_{j_2 i} - \rho \left(\lambda^k_i - s_i(x^k_i) \right),$$

$$a_{21} = (h^2 + 1)\rho, \quad a_{22} = (h^2 + 2)\rho.$$

Solution candidate:

$$\begin{bmatrix} f^k_{j_1 i} \\ f^k_{j_2 i} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}^{-1} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \frac{1}{a_{11} a_{22} - a_{21} a_{12}} \begin{bmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}.$$

- For two non-zero inflows ($f^k_{j_1 i}$, $f^k_{j_2 i}$) and $\lambda^k_i + \sum_\ell f^k_{\ell i} - s_i(x_i) \leq 0$:
  Same as in the previous case, but with

$$b_1 = h\gamma^k_i - v^k_{j_1 i} - \xi^k_i + h\rho \left(y^k_i - x^k_i - h \left(\lambda^k_i - \mu^k_i - \sum_\ell g^k_{i\ell} \right) \right) + \rho g^k_{j_1 i},$$

$$a_{11} = (h^2 + 1)\rho, \quad a_{12} = h^2 \rho,$$

$$b_2 = h\gamma^k_i - v^k_{j_2 i} - \xi^k_i + h\rho \left(y^k_i - x^k_i - h \left(\lambda^k_i - \mu^k_i - \sum_\ell g^k_{i\ell} \right) \right) + \rho g^k_{j_2 i},$$

$$a_{21} = h^2 \rho, \quad a_{22} = (h^2 + 1)\rho.$$

After checking that each of the solution candidates satisfies the assumed condition and is inside the domain, these (for the boundary and inner points) are collected and the solution candidate corresponding to the smallest augmented Lagrangian is chosen. In order to find this best candidate, only the part of the augmented Lagrangian depending on the inflow variables f^k_{ji} to cell i has to be considered. Thus, the following function (which is the relevant part of (4.2)) is evaluated in order to compare the solution candidates:

$$L^f_i = -\gamma^k_i h \sum_j f^k_{ji} + \sum_j v^k_{ji} f^k_{ji} + \xi^k_i \sum_j f^k_{ji}$$

$$+ \frac{\rho}{2} \left(y^k_i - x^k_i - h \left(\lambda^k_i - \mu^k_i + \sum_j f^k_{ji} - \sum_j g^k_{ij} \right) \right)^2 \tag{4.11}$$

$$+ \frac{\rho}{2} \sum_j (f^k_{ji} - g^k_{ji})^2 + \frac{\rho}{2} \left(\max \left\{ 0, \lambda^k_i + \sum_j f^k_{ji} - s_i(x^k_i) \right\} \right)^2.$$

The solution is the candidate ($f^k_{j_1 i}$, $f^k_{j_2 i}$) which minimizes (4.11).

Minimization With Respect To g

The g-update (4.4) in the algorithm is carried out completely analogously to the f-update. The derivative of the augmented Lagrangian (4.2) with respect to g_{ij}^k is

$$
\frac{\partial L_\rho}{\partial g_{ij}^k} = h\gamma_i^k - v_{ij}^k + \eta_i^k
$$

$$
+ h\rho \left(y_i^k - x_i^k - h \left(\lambda_i^k - \mu_i^k + \sum_\ell f_{\ell i}^k - \sum_\ell g_{i\ell}^k \right) \right) - \rho (f_{ij}^k - g_{ij}^k) \qquad (4.12)
$$

$$
+ \rho \max \left\{ 0, \mu_i^k + \sum_\ell g_{i\ell}^k - d_i(x_i^k) \right\}.
$$

As for the inflows in the f-update, each augmented Lagrangian derivative with respect to g_{ij}^k only contains outflow variables corresponding to outflows from a particular cell i at a particular time k. Thus, a timewise and cellwise decomposition is possible also in this case, resulting in minimization problems of at most dimension two, since each cell i is assumed to have at most two outflows. The subproblems to be solved in this case are thus

- For each time point $k = 1, \ldots, K$ and cell $i = 1, \ldots, N$:

$$
\underset{g_{ij}^k \in \mathcal{F}_i^{\text{out}}}{\text{minimize}} \; L_\rho, \quad \text{where} \quad \mathcal{F}_i^{\text{out}} = \{g_{ij}^k : (i,j) \in \mathcal{E}\}. \qquad (4.13)
$$

Solving all these problems separately, which each has at most dimension two, is equivalent to minimizing the augmented Lagrangian with respect to all g-variables simultaneously (i.e. (4.4)), due to the separable structure of the Lagrangian.

For the case when there is only one outflow link, (i,j), from the considered cell i, the minimizing outflow is according to (4.12) given by:

- For one outflow (g_{ij}^k) and $\mu_i^k + g_{ij}^k - d_i(x_i^k) > 0$:

$$
\frac{\partial L_\rho}{\partial g_{ij}^k} = -b + a g_{ij}^k, \quad \text{where}
$$

$$
b = -h\gamma_i^k + v_{ij}^k - \eta_i^k - h\rho \left(y_i^k - x_i^k - h \left(\lambda_i^k - \mu_i^k + \sum_\ell f_{\ell i}^k \right) \right)
$$

$$
+ \rho f_{ij}^k - \rho(\mu_i^k - d_i(x_i^k)),
$$

$$
a = (h^2 + 2)\rho.
$$

Minimizing flow: $g_{ij}^k = \max\{0, b/a\}$.

- For one outflow (g_{ij}^k) and $\mu_i^k + g_{ij}^k - d_i(x_i^k) \leq 0$:

$$\frac{\partial L_\rho}{\partial g_{ij}^k} = -b + a g_{ij}^k, \quad \text{where}$$

$$b = -h\gamma_i^k + v_{ij}^k - \eta_i^k - h\rho\left(y_i^k - x_i^k - h\left(\lambda_i^k - \mu_i^k + \sum_\ell f_{\ell i}^k\right)\right) + \rho f_{ij}^k,$$

$$a = (h^2 + 1)\rho.$$

Minimizing flow: $g_{ij}^k = \max\{0, b/a\}$.

As for the f-update, it is checked for each of these potential solutions that it satisfies the assumed constraint. After this, the remaining solution candidate that gives the smallest value of the augmented Lagrangian is chosen.

For the case when the considered cell i has two outflow links, (i, j_1) and (i, j_2), the solution $(g_{ij_1}^k, g_{ij_2}^k)$ is either inside the first quadrant of $\mathbb{R}^2$ or on its boundary, consisting of the non-negative part of the axes. First, potential minima on the boundary are found by assuming that one of the two outflows at a time is zero, and computing the other minimizing outflow according to the one-outflow case just described. Then, possible minima inside the domain are found by setting the augmented Lagrangian derivatives (4.12) corresponding to the outflow variables $g_{ij_1}^k$ and $g_{ij_2}^k$ to zero and solving for the flows. After this, it is checked that the solution is inside the domain $(g_{ij_1}^k, g_{ij_2}^k) > 0$. The solution candidates corresponding to inner points are thereby given as:

- For two non-zero outflows $(g_{ij_1}^k, g_{ij_2}^k)$ and $\mu_i^k + \sum_\ell g_{i\ell}^k - d_i(x_i^k) > 0$:

$$\frac{\partial L_\rho}{\partial g_{ij_1}^k} = -b_1 + a_{11} g_{ij_1}^k + a_{12} g_{ij_2}^k, \quad \text{where}$$

$$b_1 = -h\gamma_i^k + v_{ij_1}^k - \eta_i^k - h\rho\left(y_i^k - x_i^k - h\left(\lambda_i^k - \mu_i^k + \sum_\ell f_{\ell i}^k\right)\right)$$
$$+ \rho f_{ij_1}^k - \rho(\mu_i^k - d_i(x_i^k)),$$
$$a_{11} = (h^2 + 2)\rho, \quad a_{12} = (h^2 + 1)\rho.$$

$$\frac{\partial L_\rho}{\partial g_{ij_2}^k} = -b_2 + a_{21} g_{ij_1}^k + a_{22} g_{ij_2}^k, \quad \text{where}$$

$$b_2 = -h\gamma_i^k + v_{ij_2}^k - \eta_i^k - h\rho\left(y_i^k - x_i^k - h\left(\lambda_i^k - \mu_i^k + \sum_\ell f_{\ell i}^k\right)\right)$$
$$+ \rho f_{ij_2}^k - \rho(\mu_i^k - d_i(x_i^k)),$$
$$a_{21} = (h^2 + 1)\rho, \quad a_{22} = (h^2 + 2)\rho.$$

Solution candidate:

$$\begin{bmatrix} g_{ij_1}^k \\ g_{ij_2}^k \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}^{-1} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \frac{1}{a_{11}a_{22} - a_{21}a_{12}} \begin{bmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}.$$

- For two non-zero outflows $(g_{ij_1}^k, g_{ij_2}^k)$ and $\mu_i^k + \sum_\ell g_{i\ell}^k - d_i(x_i^k) \leq 0$:
  Same as in the previous case, but with

$$b_1 = -h\gamma_i^k + v_{ij_1}^k - \eta_i^k - h\rho \left(y_i^k - x_i^k - h \left(\lambda_i^k - \mu_i^k + \sum_\ell f_{\ell i}^k \right) \right) + \rho f_{ij_1}^k,$$

$$a_{11} = (h^2 + 1)\rho, \quad a_{12} = h^2\rho,$$

$$b_2 = -h\gamma_i^k + v_{ij_2}^k - \eta_i^k - h\rho \left(y_i^k - x_i^k - h \left(\lambda_i^k - \mu_i^k + \sum_\ell f_{\ell i}^k \right) \right) + \rho f_{ij_2}^k,$$

$$a_{21} = h^2\rho, \quad a_{22} = (h^2 + 1)\rho.$$

After checking that these solution candidates satisfy the assumed constraints and are inside the domain $(g_{ij_1}^k, g_{ij_2}^k) > 0$, the candidates are collected together with the possible minima on the boundary. Finally, the one of all these candidates that gives the smallest value of the augmented Lagrangian is chosen as the solution. This can be determined by inserting the candidates to the part of the augmented Lagrangian that depends on the considered outflows from cell i at time k:

$$\begin{aligned} L_i^g = {}& \gamma_i^k h \sum_j g_{ij}^k - \sum_j v_{ij}^k g_{ij}^k + \eta_i^k \sum_j g_{ij}^k \\ & + \frac{\rho}{2} \left(y_i^k - x_i^k - h \left(\lambda_i^k - \mu_i^k + \sum_j f_{ji}^k - \sum_j g_{ij}^k \right) \right)^2 \\ & + \frac{\rho}{2} \sum_j (f_{ij}^2 - g_{ij}^2)^2 + \frac{\rho}{2} \left(\max \left\{ 0, \mu_i^k + \sum_j g_{ij}^k - d_i(x_i^k) \right\} \right)^2. \end{aligned} \tag{4.14}$$

Thus, the solution candidate $(g_{ij_1}^k, g_{ij_2}^k)$ that minimizes (4.14) is the sought solution.

Minimization With Respect To μ

For the μ-update in the algorithm, (4.5), two main cases are distinguished. The first case is if the cell i is not a sink, i.e., $i \notin \mathcal{S}$. Then, the corresponding variables μ_i^k ($k = 1, \ldots, K$) are set to $\mu_i^k = 0$, as defined in the domain of the objective function in (3.19), as well as in the domain of the augmented Lagrangian (4.2). The second case is then the cell i is a sink, i.e., $i \in \mathcal{S}$. For this case, the augmented Lagrangian

(4.2) is differentiated with respect to μ_i^k:

$$\frac{\partial L_\rho}{\partial \mu_i^k} = h\gamma_i^k + \eta_i^k + h\rho \left(y_i^k - x_i^k - h \left(\lambda_i^k - \mu_i^k + \sum_\ell f_{\ell i}^k - \sum_\ell g_{i\ell}^k \right) \right)$$
$$+ \rho \max \left\{ 0, \mu_i^k + \sum_\ell g_{i\ell}^k - d_i(x_i^k) \right\}.$$

Each of these derivatives does not contain any other external outflow variables μ_i^k than the one which the differentiation is with respect to, i.e., the one corresponding to a specific cell i and time k. Thus, the μ-update step (4.5) can be carried out by solving the following one-dimensional minimization problems:

- For each time point $k = 1, \ldots, K$ and cell $i = 1, \ldots, N \notin \mathcal{S}$:

$$\underset{\mu_i^k}{\text{minimize }} L_\rho. \tag{4.15}$$

The minimizing μ_i^k-values for the sink cells $i \in \mathcal{S}$ are thus given by:

- For $\mu_i^k + g_{ij}^k - d_i(x_i^k) > 0$:

$$\frac{\partial L_\rho}{\partial \mu_i^k} = -b + a\mu_i^k, \quad \text{where}$$

$$b = -h\gamma_i^k - \eta_i^k - h\rho \left(y_i^k - x_i^k - h \left(\lambda_i^k + \sum_\ell f_{\ell i}^k - \sum_\ell g_{i\ell}^k \right) \right)$$
$$- \rho \left(\sum_\ell g_{i\ell}^k - d_i(x_i^k) \right),$$
$$a = (h^2 + 1)\rho.$$

  Minimizing flow: $\mu_i^k = \max\{0, b/a\}$.

- For $\mu_i^k + g_{ij}^k - d_i(x_i^k) \leq 0$:

$$\frac{\partial L_\rho}{\partial \mu_i^k} = -b + a\mu_i^k, \quad \text{where}$$

$$b = -h\gamma_i^k - \eta_i^k - h\rho \left(y_i^k - x_i^k - h \left(\lambda_i^k + \sum_\ell f_{\ell i}^k - \sum_\ell g_{i\ell}^k \right) \right),$$
$$a = h^2\rho.$$

  Minimizing flow: $\mu_i^k = \max\{0, b/a\}$.

After that it has been checked that these solution candidates satisfy the assumed constraints, the candidate that minimizes the μ_i^k-dependent part of the augmented Lagrangian is chosen as the solution. Thus, the solution is the candidate that minimizes

$$
L_i^\mu = \gamma_i^k h \mu_i^k + \eta_i^k \mu_i^k + \frac{\rho}{2}\left(y_i^k - x_i^k - h\left(\lambda_i^k - \mu_i^k + \sum_\ell f_{\ell i}^k - \sum_\ell g_{i\ell}^k\right)\right)^2
$$
$$
+ \frac{\rho}{2}\left(\max\left\{0, \mu_i^k + \sum_\ell g_{i\ell}^k - d_i(x_i^k)\right\}\right)^2.
$$
(4.16)

Minimization With Respect To y

For the y-update in the algorithm, (4.6), the augmented Lagrangian (4.2) is differentiated with respect to y_i^k:

$$
\frac{\partial L_\rho}{\partial y_i^k} =
\begin{cases}
\psi_i'(y_i^k) + \gamma_i^k + \sigma_i^k \\
\quad + \rho\left(-x_i^k - h\left(-x_i^k - h(\lambda_i^k - \mu_i^k + \sum_\ell f_{\ell i}^k - \sum_\ell g_{i\ell}^k)\right)\right) \\
\quad + \rho(y_i^k - x_i^{k+1}), \quad \text{if } k < K \\[1em]
\psi_i'(y_i^k) + \gamma_i^k \\
\quad + \rho\left(-x_i^k - h\left(-x_i^k - h(\lambda_i^k - \mu_i^k + \sum_\ell f_{\ell i}^k - \sum_\ell g_{i\ell}^k)\right)\right), \\
\quad \text{if } k = K.
\end{cases}
$$
(4.17)

Since the cost functions ψ_i are assumed to be either linear or quadratic, the derivatives (4.17) are affine functions of y_i^k. Furthermore, each derivative contains no other y-variables than the y_i^k that the differentiation is carried out with respect to. Thereby, one again obtains a set of one-dimensional minimization subproblems:

- For each time point $k = 1, \ldots, K$ and cell $i = 1, \ldots, N$:

$$
\underset{y_i^k}{\text{minimize }} L_\rho.
$$
(4.18)

The y-update step (4.6) is solved exactly by solving these subproblems separately.

For each cell i and time point k the subproblem (4.18) is solved by setting the corresponding derivative to zero, solving for y_i^k and taking $\max\{0, y_i^k\}$ as the optimal non-negative point. For the different cost functions and time points, the solutions are thereby given as:

- If $\psi_i(y) = y$:

- For $k < K$:

$$\frac{\partial L_\rho}{\partial y_i^k} = -b + a y_i^k, \quad \text{where}$$

$$b = -1 - \gamma_i^k - \sigma_i^k$$

$$- \rho \left(-x_i^k - h \left(\lambda_i^k - \mu_i^k + \sum_\ell f_{\ell i}^k - \sum_\ell g_{i\ell}^k \right) \right) + \rho x_i^{k+1},$$

$$a = 2\rho.$$

Minimizing variable: $y_i^k = \max\{0, b/a\}$.

- For $k = K$:

$$\frac{\partial L_\rho}{\partial y_i^k} = -b + a y_i^k, \quad \text{where}$$

$$b = -1 - \gamma_i^k$$

$$- \rho \left(-x_i^k - h \left(\lambda_i^k - \mu_i^k + \sum_\ell f_{\ell i}^k - \sum_\ell g_{i\ell}^k \right) \right),$$

$$a = \rho.$$

Minimizing variable: $y_i^k = \max\{0, b/a\}$.

- If $\psi_i(y) = y^2$:

  - For $k < K$:

$$\frac{\partial L_\rho}{\partial y_i^k} = -b + a y_i^k, \quad \text{where}$$

$$b = -\gamma_i^k - \sigma_i^k$$

$$- \rho \left(-x_i^k - h \left(\lambda_i^k - \mu_i^k + \sum_\ell f_{\ell i}^k - \sum_\ell g_{i\ell}^k \right) \right) + \rho x_i^{k+1},$$

$$a = 2(1 + \rho).$$

Minimizing variable: $y_i^k = \max\{0, b/a\}$.

  - For $k = K$:

$$\frac{\partial L_\rho}{\partial y_i^k} = -b + a y_i^k, \quad \text{where}$$

$$b = -\gamma_i^k - \rho \left(-x_i^k - h \left(\lambda_i^k - \mu_i^k + \sum_\ell f_{\ell i}^k - \sum_\ell g_{i\ell}^k \right) \right),$$

$$a = (2 + \rho).$$

Minimizing variable: $y_i^k = \max\{0, b/a\}$.

Minimization With Respect To x

For the last Lagrangian minimization step (4.7), the augmented Lagrangian (4.2) is differentiated with respect to x_i^k for $k = 2, \ldots, K$ (since x_i^1 are given):

$$
\begin{aligned}
\frac{\partial L_\rho}{\partial x_i^k} = & -\gamma_i^k - \sigma_i^{k-1} - \xi_i^k s_i'(x_i^k) - \eta_i^k d_i'(x_i^k) \\
& - \rho\left(y_i^k - x_i^k - h\left(\lambda_i^k - \mu_i^k + \sum_\ell f_{\ell i}^k - \sum_\ell g_{i\ell}^k\right)\right) - \rho(y_i^{k-1} - x_i^k) \\
& - \rho \max\left\{0, \lambda_i^k + \sum_\ell f_{\ell i}^k - s_i(x_i^k)\right\} s_i'(x_i^k) \\
& - \rho \max\left\{0, \mu_i^k + \sum_\ell g_{i\ell}^k - d_i(x_i^k)\right\} d_i'(x_i^k), \quad \text{for } k = 2, \ldots, K.
\end{aligned}
\tag{4.19}
$$

Each derivative does not depend on any other x-variables than the x_i^k which the differentiation is carried out with respect to. Thereby, the minimization of the augmented Lagrangian with respect to x, i.e. (4.7), can be carried out in a distributed manner by separately solving the one-dimensional subproblems

- For each time point $k = 1, \ldots, K$ and cell $i = 1, \ldots, N$:

$$
\underset{x_i^k}{\text{minimize}} \; L_\rho.
\tag{4.20}
$$

Furthermore, the supply and demand functions are assumed to be affine. Thus, these functions and their derivatives can be written as

$$
\begin{aligned}
s_i(x) = k_i^s x + m_i^s \quad &\Rightarrow \quad s_i'(x) = k_i^s, \\
d_i(x) = k_i^d x + m_i^d \quad &\Rightarrow \quad d_i'(x) = k_i^d.
\end{aligned}
$$

Inserting this into (4.19) gives

$$
\begin{aligned}
\frac{\partial L_\rho}{\partial x_i^k} = & -\gamma_i^k - \sigma_i^{k-1} - \xi_i^k k_i^s - \eta_i^k k_i^d \\
& - \rho\left(y_i^k - x_i^k - h\left(\lambda_i^k - \mu_i^k + \sum_\ell f_{\ell i}^k - \sum_\ell g_{i\ell}^k\right)\right) - \rho(y_i^{k-1} - x_i^k) \\
& - \rho \max\left\{0, \lambda_i^k + \sum_\ell f_{\ell i}^k - (k_i^s x + m_i^s)\right\} k_i^s \\
& - \rho \max\left\{0, \mu_i^k + \sum_\ell g_{i\ell}^k - (k_i^d x + m_i^d)\right\} k_i^d, \quad \text{for } k = 2, \ldots, K.
\end{aligned}
\tag{4.21}
$$

This derivative consists of piecewise affine functions of x_i^k. The points where the derivative is zero can thereby be found by considering each of the affine parts of the function separately. The following cases are thereby considered:

- For $\lambda_i^k + \sum_\ell f_{\ell i}^k - s_i(x_i^k) > 0$ and $\mu_i^k + \sum_\ell g_{i\ell}^k - d_i(x_i^k) > 0$:

$$\frac{\partial L_\rho}{\partial x_i^k} = -b + a, \quad \text{where}$$

$$b = \gamma_i^k + \sigma_i^{k-1} + \xi_i^k k_i^s + \eta_i^k k_i^d$$

$$+ \rho \left(y_i^k - h \left(\lambda_i^k - \mu_i^k + \sum_\ell f_{\ell i}^k - \sum_\ell g_{i\ell}^k \right) \right) + \rho y_i^{k-1}$$

$$+ \rho k_i^s \left(\lambda_i^k + \sum_\ell f_{\ell i}^k - m_i^s \right) + \rho k_i^d \left(\mu_i^k + \sum_\ell g_{i\ell}^k - m_i^d \right),$$

$$a = \left(2 + (k_i^s)^2 + (k_i^d)^2 \right) \rho.$$

Minimizing variable: $x_i^k = b/a$.

- For $\lambda_i^k + \sum_\ell f_{\ell i}^k - s_i(x_i^k) > 0$ and $\mu_i^k + \sum_\ell g_{i\ell}^k - d_i(x_i^k) \le 0$:
  Same as the previous case, but remove the term $\rho k_i^d \left(\mu_i^k + \sum_\ell g_{i\ell}^k - m_i^d \right)$ from b and the term $(k_i^d)^2 \rho$ from a.

- For $\lambda_i^k + \sum_\ell f_{\ell i}^k - s_i(x_i^k) \le 0$ and $\mu_i^k + \sum_\ell g_{i\ell}^k - d_i(x_i^k) > 0$:
  Same as the first case, but remove the term $\rho k_i^s \left(\lambda_i^k + \sum_\ell f_{\ell i}^k - m_i^s \right)$ from b and the term $(k_i^s)^2 \rho$ from a.

- For $\lambda_i^k + \sum_\ell f_{\ell i}^k - s_i(x_i^k) \le 0$ and $\mu_i^k + \sum_\ell g_{i\ell}^k - d_i(x_i^k) \le 0$:
  Same as the first case, but remove the terms $\rho k_i^d \left(\mu_i^k + \sum_\ell g_{i\ell}^k - m_i^d \right)$ and $\rho k_i^s \left(\lambda_i^k + \sum_\ell f_{\ell i}^k - m_i^s \right)$ from b as well as the terms $(k_i^d)^2 \rho$ and $(k_i^s)^2 \rho$ from a.

The solutions from these different cases that satisfies the assumed conditions are collected as solution candidates. After this, the solution candidate that minimizes the augmented Lagrangian is chosen as the solution. This is equivalent to that the solution candidate minimizes the x_i^k-dependent part of the augmented Lagrangian,

which is

$$
\begin{aligned}
L_i^x = & -\gamma_i^k x_i^k - \sigma_i^{k-1} x_i^k - \xi_i^k s_i(x_i^k) - \eta_i^k d_i(x_i^k) \\
& + \frac{\rho}{2} \left(y_i^k - x_i^k - h\left(\lambda_i^k - \mu_i^k + \sum_\ell f_{\ell i}^k - \sum_\ell g_{i\ell}^k \right) \right)^2 + \frac{\rho}{2} \left(y_i^{k-1} - x_i^k \right)^2 \\
& + \frac{\rho}{2} \left(\max\left\{ 0, \lambda_i^k + \sum_\ell f_{\ell i}^k - s_i(x_i^k) \right\} \right)^2 \\
& + \frac{\rho}{2} \left(\max\left\{ 0, \mu_i^k + \sum_\ell g_{i\ell}^k - d_i(x_i^k) \right\} \right)^2 .
\end{aligned}
\tag{4.22}
$$

Modification of x-update for Capacity Limited Flows

The algorithm can be generalized to handle supply and demand functions which are not just affine, but which also takes into account a limited flow capacity for the cell. The functions are of the form

$$
s_i(x) = \min\{k_i^s x + m_i^s, C_i\}, \tag{4.23}
$$
$$
d_i(x) = \min\{k_i^d x + m_i^d, C_i\}, \tag{4.24}
$$

where C_i is the maximum flow capacity for cell i. Thus, the supply and demand functions are not simply affine, but piecewise affine. The minimization of the augmented Lagrangian can again be carried out by considering the different cases corresponding to the two different affine parts of the function.

All the different cases that should be considered in the augmented Lagrangian minimization with respect to x_i^k do now depend on whether four different conditions are fulfilled or not. These conditions are

- c_1: $\lambda_i^k + \sum_\ell f_{\ell i}^k - s_i(x_i^k) > 0$,

- c_2: $\mu_i^k + \sum_\ell g_{i\ell}^k - d_i(x_i^k) > 0$,

- c_3: $s_i(x_i^k) = C_i$,

- c_4: $d_i(x_i^k) = C_i$.

If none of the conditions are satisfied, then the augmented Lagrangian derivative

(4.19) becomes

$$\frac{\partial L_\rho}{\partial x_i^k} = -b + a, \quad \text{where}$$

$$b = \gamma_i^k + \sigma_i^{k-1} + \xi_i^k k_i^s + \eta_i^k k_i^d$$

$$+ \rho \left(y_i^k - h \left(\lambda_i^k - \mu_i^k + \sum_\ell f_{\ell i}^k - \sum_\ell g_{i\ell}^k \right) \right) + \rho y_i^{k-1},$$

$$a = \left(2 + (k_i^s)^2 + (k_i^d)^2 \right) \rho.$$

(4.25)

Setting this to zero gives the solution candidate $x_i^k = b/a$ if this is positive and the conditions c_1–c_4 are not satisfied, as assumed.

The remaining solution candidates are found by considering all different combinations of the conditions c_1–c_4 being satisfied and not being satisfied. For the cases when some of the conditions are satisfied, (4.25) is modified as follows:

- If c_1 is satisfied, then

    - the term $\rho k_i^s \left(\lambda_i^k + \sum_\ell f_{\ell i}^k - m_i^s \right)$ is added in b,
    - the term $(k_i^s)^2 \rho$ is added in a.

- If c_2 is satisfied, then

    - the term $\rho k_i^d \left(\mu_i^k + \sum_\ell g_{i\ell}^k - m_i^d \right)$ is added in b,
    - the term $(k_i^d)^2 \rho$ is added in a.

- If c_3 is satisfied, then

    - k_i^s is replaced with 0,
    - m_i^s is replaced with C_i.

- If c_4 is satisfied, then

    - k_i^d is replaced with 0,
    - m_i^d is replaced with C_i.

After the modifications, the solution candidate for each specific case is again obtained as $x_i^k = b/a$, if this is a non-negative value fulfilling the assumed conditions. The solution is then chosen as the solution candidate which gives the smallest value of the x_i^k-dependent part of the augmented Lagrangian, (4.22), for the new supply and demand functions.

4.3   Stopping Criterion

To determine when the iterative scheme should be stopped, a stopping criterion is needed. For the static algorithm, the criterion was, as described in Section 2.3, chosen as requiring that the duality gap is small in combination with that the constraints are approximately satisfied. The same approach is used for the dynamic optimization algorithm, resulting in the following criterion:

$$
\left| \sum_{k=1}^{K} \sum_{i=1}^{N} \psi_i(y_i^k) - \theta(\gamma, \xi, \eta) \right| < \text{gapTol}, \quad \text{and}
$$

$$
\begin{aligned}
&\sum_{k=1}^{K} \sum_{i=1}^{N} \left| y_i^k - x_i^k - h \left(\lambda_i^k - \mu_i^k + \sum_\ell f_{\ell i}^k - \sum_\ell g_{i\ell}^k \right) \right| \\
&+ \sum_{k=1}^{K} \sum_{e=1}^{E} |f_e^k - g_e^k| + \sum_{k=1}^{K-1} \sum_{i=1}^{N} |y_i^k - x_i^{k+1}| \\
&+ \sum_{k=1}^{K} \sum_{i=1}^{N} \max \left\{ 0, \ \lambda_i^k + \sum_\ell f_{\ell i}^k - s_i(x_i^k) \right\} \\
&+ \sum_{k=1}^{K} \sum_{i=1}^{N} \max \left\{ 0, \ \mu_i^k + \sum_\ell g_{i\ell}^k - d_i(x_i^k) \right\} < \text{feasTol},
\end{aligned}
\tag{4.26}
$$

where $\theta(\gamma, \xi, \eta)$ is the dual function, gapTol is the **duality gap tolerance** and feasTol is the **feasibility residual tolerance**. The tolerances should be chosen as small positive numbers.

5

Tests and Results

In this chapter, the algorithm is tested for two different problems. The first test problem is rather simple and is described in **Section 5.1**. The second test problem is a bit more complicated and is described in **Section 5.2**. The algorithm is tested for these problems with different choices of the penalty parameter ρ. In order to compare the results, the quantities described in section **Section 5.3** are used. The results of the tests are presented in **Section 5.4** and **Section 5.5** for test problem 1 and test problem 2 respectively.

5.1 Test Problem 1

In order to test the derived distributed algorithm (4.3)–(4.8) on a simple case, the road network in Fig. 5.1 is considered. The arrows indicate the allowed driving direction on the roads. This network is divided into four cells according to the figure, which results in a graph representation according to Fig. 5.2. This graph is described by the node-link incidence matrix

$$B = \begin{bmatrix} +1 & +1 & 0 & 0 \\ -1 & 0 & +1 & 0 \\ 0 & -1 & 0 & +1 \\ 0 & 0 & -1 & -1 \end{bmatrix}$$

according to the definition (2.1). It is assumed that vehicles enter the network from the external world at cell 1, which is the only source. The external inflow is thus given by λ_1. Furthermore, it is assumed that vehicles can leave the network only at cell 4, which thus is the only sink node. The external outflow is thereby given by μ_4.

The supply and demand functions for the problem are chosen as

$$s_1(x) = 2\sum_k \sum_i \lambda_i^k, \tag{5.1}$$

$$s_i(x) = 10 - x, \quad i = 2,\ldots,N, \tag{5.2}$$

$$d_i(x) = x, \quad i = 1,\ldots,N. \tag{5.3}$$
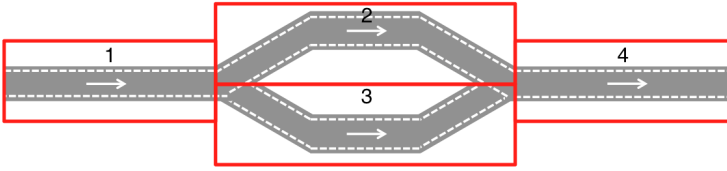
Figure 5.1   Simple road network with a 4-cell discretization that is used for testing the algorithm.
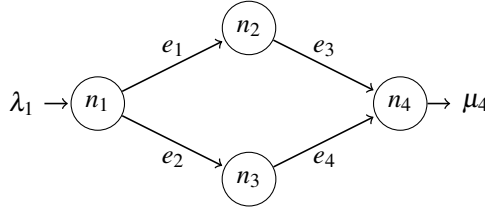
Figure 5.2   Model for the simple road network in Fig. 5.1 that is used for testing the algorithm.

The function $s_1(x)$ is chosen as a constant that is guaranteed to always be larger than the flow (since it is two times the total flow that goes through the network during the entire simulation). This implies that the supply constraint always is fulfilled for cell 1, and is equivalent to not having any supply constraint for this cell. The purpose of this is that the problem might lack a feasible solution if the source cell 1 is supply-constrained and the inflow λ_1 is large. By not having a supply function for this cell, this is prevented. This can be interpreted as that cell 1 is an on-ramp where arbitrarily large traffic queues can appear while waiting for the opportunity to enter the rest of the network.

The tests will be carried out with the time step interval $h = 1$ and during 10 time units, i.e., $K = 10$. Furthermore, it is assumed that the initial traffic volume is $x_i^1 = 0$ for all cells i.

The algorithm will be tested on this problem with some different configurations as follows:

- The **cost function** will be chosen as a linear function $\psi_i(x) = x$ or a quadratic function $\psi_i(x) = x^2$.

- The **inflow** will be chosen as either $\lambda_1^1 = 1$ and $\lambda_1^k = 0$ for $k = 2, \ldots, K$ or as $\lambda_1^k = 1$ for $k = 1, \ldots, K$.

- The optimization will be carried out for different values of the **penalty parameter** ρ that is used in the augmented Lagrangian and as step length in the dual variable update.

65

Finally, it is examined how the algorithm handles a temporary traffic incident. It is assumed that an incident occurs in the time interval $t \in [3,5)$, which corresponds to the intervals including and after the discrete time points $k = 4$ and $k = 5$, since $t_k = (k-1)h$ for $k = 1, \ldots, K$ according to (3.6). The incident causes a blockage in cell 3 such that there cannot be any inflow to this cell during the considered time interval. This is implemented by changing the supply functions $s_3^4(x)$ and $s_3^5(x)$ in (5.2) to $s_3^4(x) = 0$ and $s_3^5(x) = 0$.

5.2 Test Problem 2

In order to test the distributed algorithm (4.3)–(4.8) on a slightly more complicated problem, an example from [Como et al., 2016] is used. The used setup is different from the one used in the reference by a small detail, namely that the capacity C_4^k only is changed for the supply function in the reference, while it in this implementation is set to be the same for the supply and demand function. This gives slightly different results when the capacity is changed, but the overall behaviour is similar and the optimal values are close to each other.

In the test problem, the single-source single-sink network illustrated in Fig. 5.3 is studied. In this network, external inflows are prescribed at cell 1, and external outflows are allowed only at cell 10. Demand constraints are imposed on all cells, and supply constrains are imposed on all cells except for cell 1, in order to allow for arbitrary external inflows λ_1 as in test problem 1. This first cell can thus be interpreted as the queue on the on-ramp of the road, which can be arbitrarily long. The supply and demand functions are given by

$$s_i(x_i, k) = \min\left\{ \frac{w_i(x_i^{\text{jam}} - x_i)}{L_i}, C_i^k \right\}, \quad i = 2, \ldots, N, \tag{5.4}$$

$$d_i(x_i, k) = \min\left\{ \frac{v_i x_i}{L_i}, C_i^k \right\}, \quad i = 1, \ldots, N, \tag{5.5}$$

where v_i is the free-flow speed, w_i the speed of the congestion wave, C_i^k the capacity (at time step k), L_i the cell length and x_i^{jam} the jam traffic volume for cell i. The values of the parameters are given in Tab. 5.1. The supply and demand functions can be written on the forms (4.23) and (4.24) respectively with

$$k_i^s = -w_i/L_i, \quad m_i^s = w_i x_i^{\text{jam}}/L_i,$$
$$k_i^d = v_i/L_i, \quad m_i^d = 0.$$

The lack of supply function for the first cell is implemented by setting $k_i^s = 0$, $m_i^s = 2\sum_k \sum_i \lambda_i$ and $C_i^k = 2\sum_k \sum_i \lambda_i$ in (4.23), so that the supply constraint always is satisfied, which is equivalent to that there is no supply constraint for the cell in question.
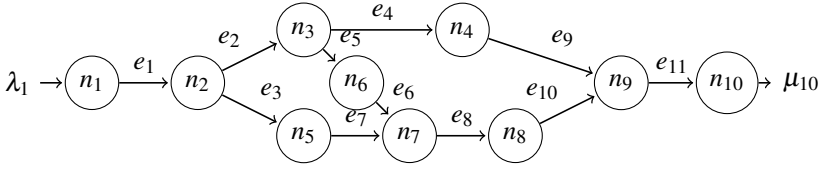
Figure 5.3   Network used for testing the dynamic optimization algorithm.

In the studied case, the maximum flow capacity is reduced for cell 4 at some time points, in order to simulate a time dependent bottleneck. Furthermore, the sampling time used is $h = 10$ seconds. The external inflow is given as $\lambda_1^1 = 0.8$, $\lambda_1^2 = 1.6$, $\lambda_1^3 = 0.8$ and $\lambda_1^k = 0$ for $k \geq 4$. The cost function is chosen either as the sum of the total traffic volume in all cells for all time points: $\sum_{k=1}^{K} \sum_{i=1}^{N} x_i^{k+1}$, i.e., $\psi_i(x) = x$, or as the the sum of the squares of the traffic volumes in all cells at all time points, i.e., $\psi_i(x) = x^2$.

Parameter	Value
Free-flow speed v_i	50 feet/s
Wave speed w_i	50 feet/s
Length of cell L_i	500 feet
Number of lanes ℓ_i	2 for $i = 1, 2, 9, 10$
	1 otherwise
Capacity C_i^k	$0.6\ell_i$ vehicles/s for $i \neq 4$
	$C_4^5 = C_4^6 = 0$ vehicles/s
	$C_4^7 = C_4^8 = 0.3$ vehicles/s
	$C_4^k = 0.6$ vehicles/s for $k \neq 5, 6, 7, 8$
Jam volume x^{jam}	$10\ell_i$ vehicles

Table 5.1   Cell parameters.

5.3   Test Quantities

In order to determine the accuracy of the results, the convex dynamic optimal flow problem (3.19) was solved with the MATLAB-based software CVX [CVX Research Inc., 2017], and the results from the algorithm were compared with this solution. CVX solves the problem in a centralized, or non-distributed, manner. This means that all decision variables are considered simultaneously, which makes the problem difficult and very time-consuming to solve if the problem is of large scale. However, the solution method works well for solution of small problems and can thus be used as a reference solution in the cases tested here.

The accuracy of the cost is measured by the **relative cost error**

$$\varepsilon_\Psi = \frac{|\Psi - \Psi^*|}{\Psi^*},$$

where $\Psi = \sum_{k=1}^{K} \sum_{i=1}^{N} \psi_i(y_i^k)$ is the optimal value of the cost function according to the algorithm and Ψ^* is the optimal cost according to CVX. The accuracy of the cell traffic volumes is measured by the **average cell volume error**

$$\bar{\varepsilon}_y = \frac{1}{KN} \sum_{k=1}^{K} \sum_{i=1}^{N} |y_i^k - (x_i^{k+1})^*|$$

and the **maximum cell volume error**

$$\varepsilon_y^{\max} = \max_{i,k} |y_i^k - (x_i^{k+1})^*|,$$

where y_i^k is the cell traffic volume in cell i at time $k+1$ according to the derived algorithm, and $(x_i^{k+1})^*$ is the same cell volume according to CVX.

For the case with the linear cost function, the solution is not unique. Thus, the errors in cell volumes have not been included here, since the volumes could be different from the reference calculations but still be a feasible solution with the same optimal value of the cost function. However, since the quadratic cost gives a strictly convex objective function in the cell volume variables, the optimal cell volume variables are unique and can be compared with the reference. Thus, this comparison is made for the quadratic cost case only.

5.4   Results from Test Problem 1

In order to examine how well the algorithm works for different values of the penalty parameter ρ, it was run for test problem 1 with different values of this parameter. First, the algorithm was tested with a unit pulse inflow: $\lambda_1^1 = 1$, $\lambda_1^k = 0$ for all $k \geq 2$, and a linear cost function, $\psi_i(x) = x$. The stopping criterion (4.26) was chosen as requiring that both the feasibility residual and the duality gap are less than 10^{-3}. The results from this are presented in Tab. 5.2. Furthermore, the cost function, feasibility residual and duality gap are plotted as a function of number of iterations in Fig. 5.4. The resulting cell traffic volumes are plotted in Fig. 5.5 for the case $\rho = 1$ only. However, the cell volumes for the other tested ρ-values are so close to this case that the different cases cannot be distinguished by looking at the figure in the given scale.

Repeating the same procedure but with the only difference that the cost is chosen as a quadratic function: $\psi_i(x) = x^2$, gave the result presented in Tab. 5.3. The cost function, feasibility residual and duality gap changed during the execution of the

penalty parameter ρ	0.001	0.01	0.1
number of iterations	32630	3608	759
computation time [s]	237	25	6
relative cost err. ε_ψ	56×10^{-6}	1.9×10^{-6}	142×10^{-6}
penalty parameter ρ	**1**	**10**	**100**
number of iterations	519	187	90
computation time [s]	4	1	1
relative cost err. ε_ψ	165×10^{-6}	235×10^{-6}	70×10^{-6}

Table 5.2   Results for test problem 1 with $\lambda_1^1 = 1$, $\lambda_1^k = 0$, $k \geq 2$, different penalty parameters ρ, **linear cost function** $\psi_i(x) = x$, feasibility residual tolerance 10^{-3} and duality gap tolerance 10^{-3}.

algorithm as illustrated in Fig. 5.6. The resulting cell traffic volumes are shown for some of the ρ-values in Fig. 5.7.

After this, the same test was repeated for both the linear cost $\psi_i(x) = x$ and the quadratic cost $\psi_i(x) = x^2$, but with the inflow chosen to be constant at one, i.e., $\lambda_1^k = 1$ for all k. The result from this with the linear cost is given in Tab. 5.4. In these tests, an upper limit of the number of iterations was chosen as 500,000. Thus, in the cases where the number of iterations required to satisfy the stopping criterion is larger than this, the results are presented for 500,000 iterations instead. The cost function, feasibility residual and duality gap are illustrated in Fig. 5.8 and the cell volumes for some ρ-values in Fig. 5.9. The result from the case with the quadratic cost is given in Tab. 5.5. The cost function, feasibility residual and duality gap are shown in Fig. 5.10 and the cell volumes for some ρ-values in Fig. 5.11.

Finally, the algorithm was tested on the modified problem, where the supply function for cell 3 was changed for time points 4 and 5, such that $s_3^4(x) = s_3^5(x) = 0$. This corresponds to simulating a blockage such that there cannot be any inflow to cell 3 in the time interval $t \in [3, 5)$. In this test, the quadratic cost function $\psi_i(x) = x^2$ was used. The result from this is reported in Tab. 5.6. The cost function, feasibility residual and duality gap as a function of the number of iterations are shown in Fig. 5.12 and the cell volumes in Fig. 5.13.
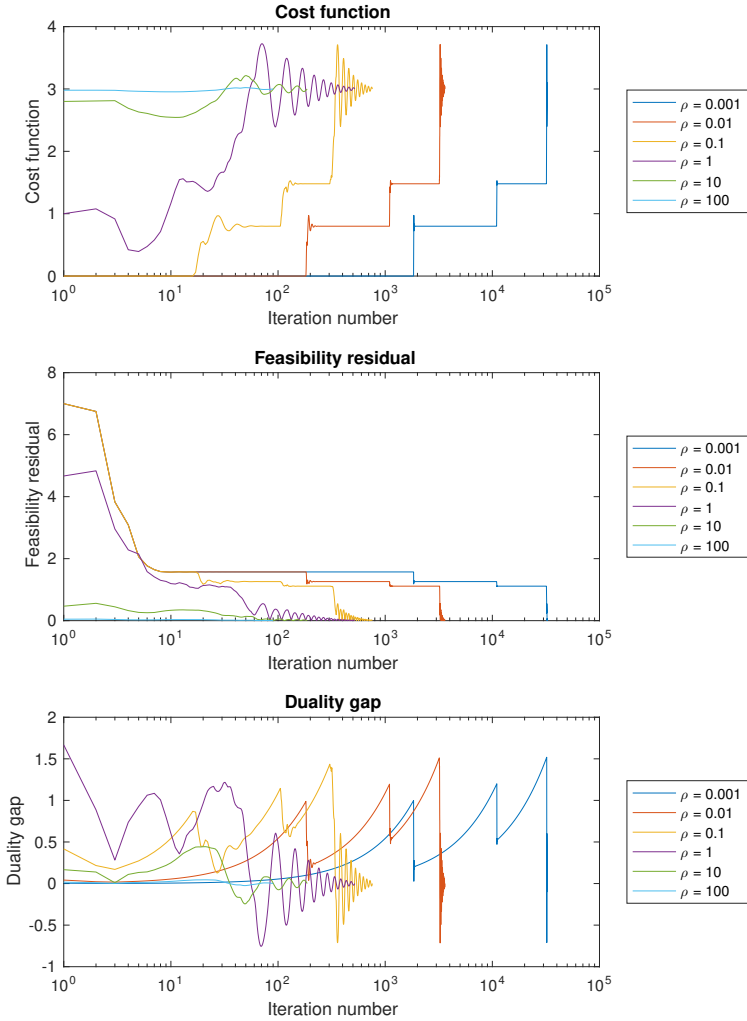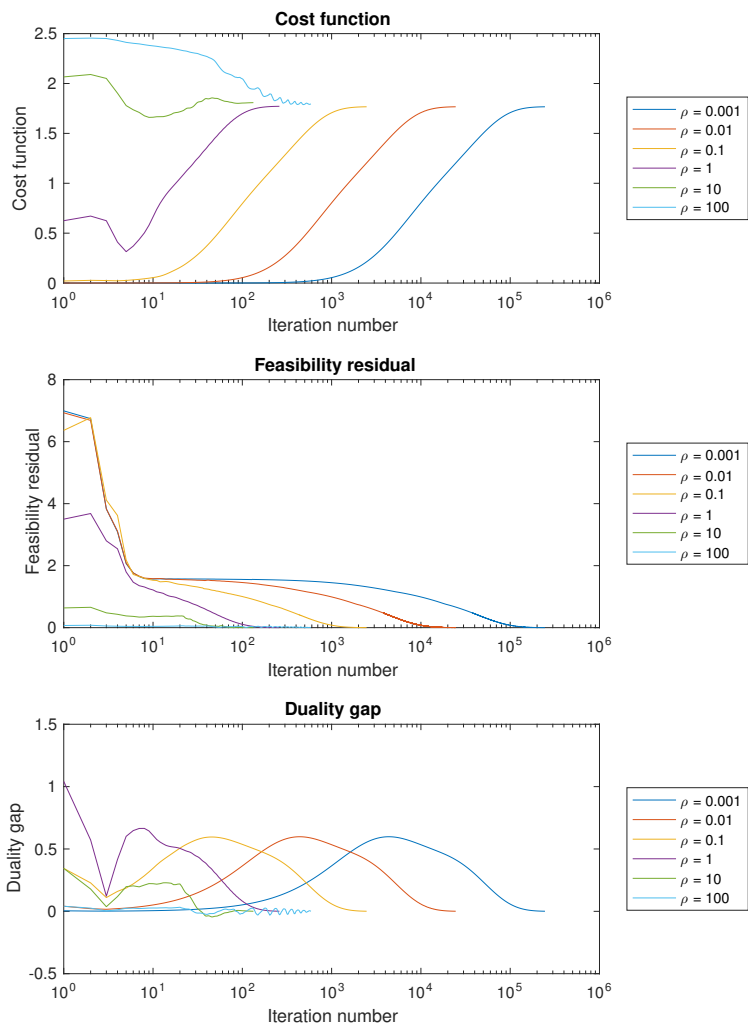
Figure 5.4 Cost function, feasibility residual and duality gap for test problem 1 with $\lambda_1^1 = 1$, $\lambda_1^k = 0$, $k \geq 2$, different penalty parameters ρ, **linear cost function** $\psi_i(x) = x$, feasibility residual tolerance 10^{-3} and duality gap tolerance 10^{-3}.
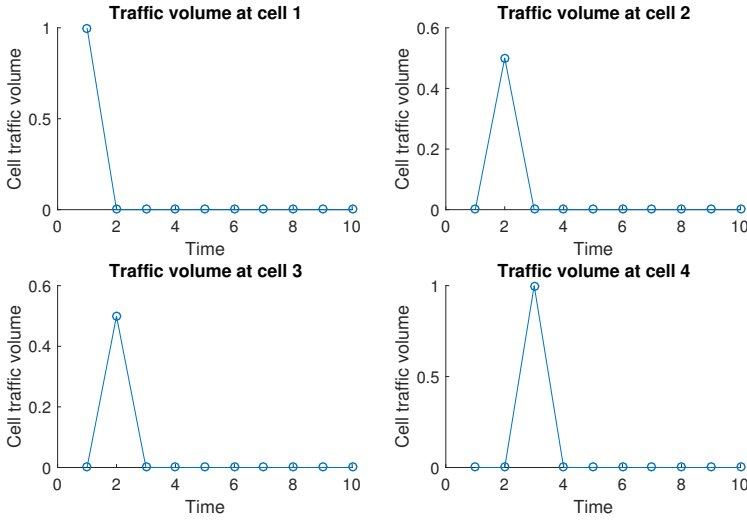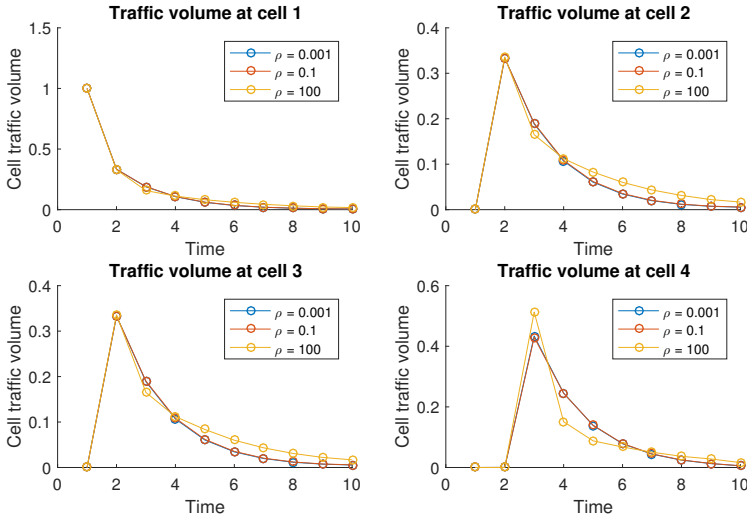
Figure 5.5   Cell traffic volumes for test problem 1 with $\lambda_1^1 = 1$, $\lambda_1^k = 0$, $k \geq 2$, penalty parameter $\rho = 1$, **linear cost function** $\psi_i(x) = x$, feasibility residual tolerance 10^{-3} and duality gap tolerance 10^{-3}.

penalty parameter ρ	0.001	0.01	0.1
number of iterations	245037	24518	2465
computation time [s]	2394	207	20
relative cost err. ε_ψ	424×10^{-6}	442×10^{-6}	409×10^{-6}
mean cell vol. err. $\bar{\varepsilon}_y$	51×10^{-6}	87×10^{-6}	816×10^{-6}
max cell vol. err. $\varepsilon_y^{\max}$	313×10^{-6}	682×10^{-6}	4.7×10^{-3}
penalty parameter ρ	1	10	100
number of iterations	260	133	586
computation time [s]	2	1	5
relative cost err. ε_ψ	2.3×10^{-3}	23×10^{-3}	16×10^{-3}
mean cell vol. err. $\bar{\varepsilon}_y$	8.0×10^{-3}	22×10^{-3}	19×10^{-3}
max cell vol. err. $\varepsilon_y^{\max}$	41×10^{-3}	109×10^{-3}	95×10^{-3}

Table 5.3   Results for test problem 1 with $\lambda_1^1 = 1$, $\lambda_1^k = 0$, $k \geq 2$, different penalty parameters ρ, **quadratic cost function** $\psi_i(x) = x^2$, feasibility residual tolerance 10^{-3} and duality gap tolerance 10^{-3}.
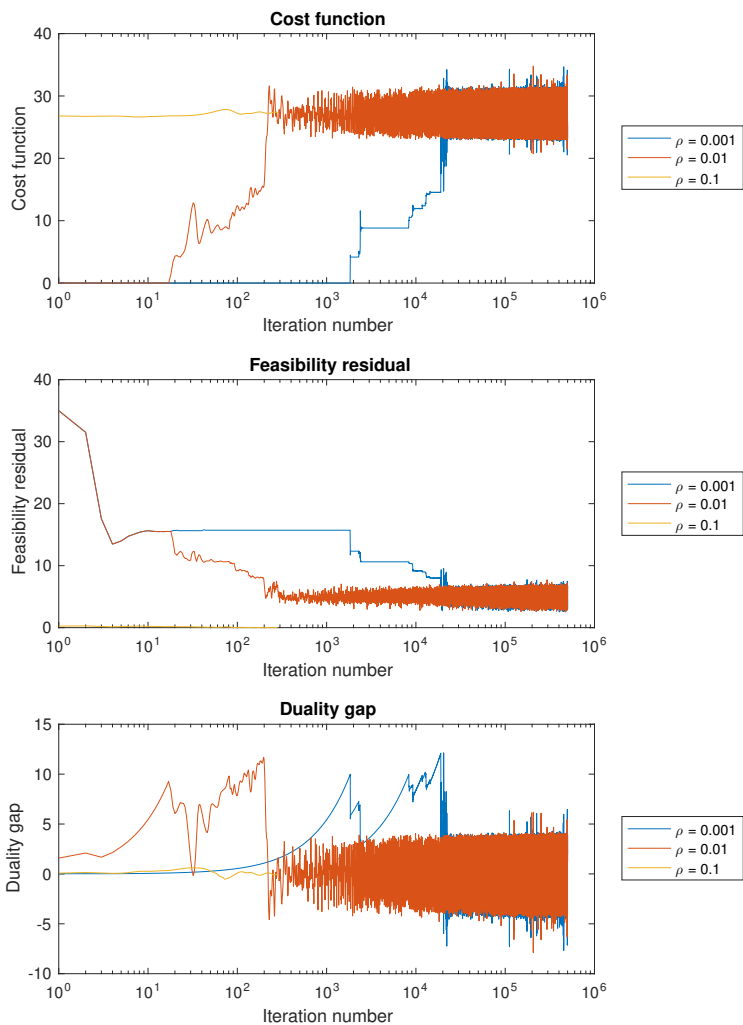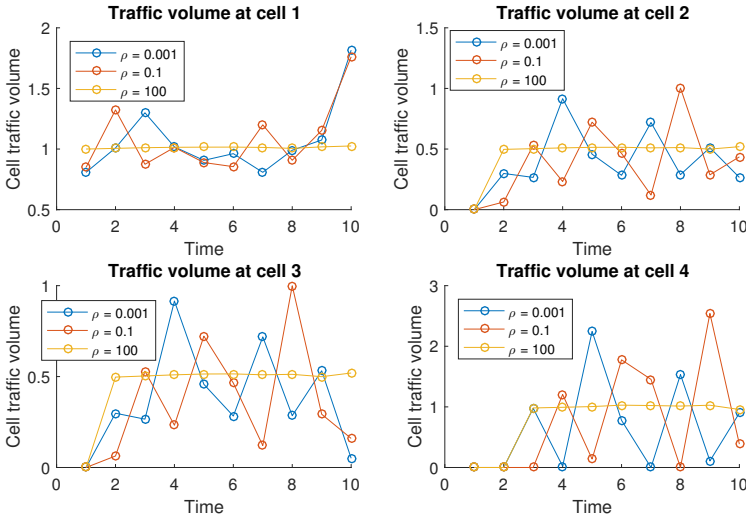
Figure 5.6 Cost function, feasibility residual and duality gap for test problem 1 with $\lambda_1^1 = 1$, $\lambda_1^k = 0$, $k \geq 2$, different penalty parameters ρ, **quadratic cost function** $\psi_i(x) = x^2$, feasibility residual tolerance 10^{-3} and duality gap tolerance 10^{-3}.

Figure 5.7   Cell traffic volume for test problem 1 with $\lambda_1^1 = 1$, $\lambda_1^k = 0$, $k \geq 2$, different penalty parameters ρ, **quadratic cost function** $\psi_i(x) = x^2$, feasibility residual tolerance 10^{-3} and duality gap tolerance 10^{-3}.

penalty parameter ρ	0.001	0.01	0.1
number of iterations	> 500000	> 500000	> 500000
computation time [s]	6475	6363	6342
relative cost err. ε_ψ	72×10^{-3}	2.9×10^{-3}	46×10^{-3}
penalty parameter ρ	1	10	100
number of iterations	> 500000	820	283
computation time [s]	6320	6	2
relative cost err. ε_ψ	87×10^{-3}	117×10^{-3}	12×10^{-3}

Table 5.4   Results for test problem 1 with $\lambda_1^k = 1$ for all k, different penalty parameters ρ, **linear cost function** $\psi_i(x) = x$, feasibility residual tolerance 10^{-3} and duality gap tolerance 10^{-3}.
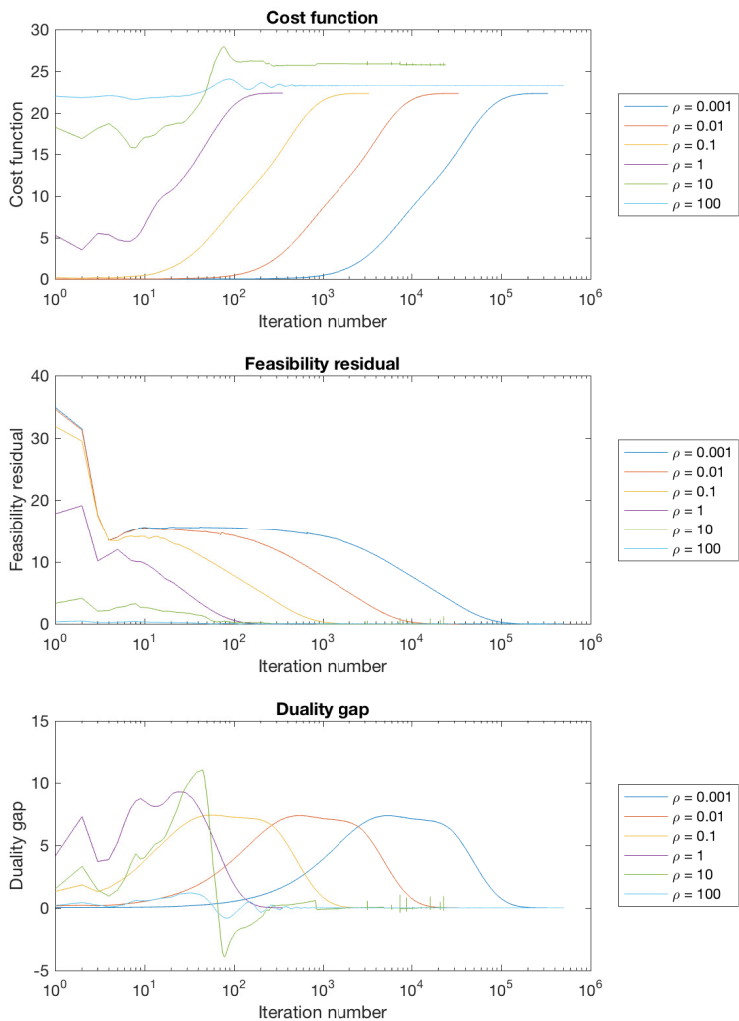
Figure 5.8 Cost function, feasibility residual and duality gap for test problem 1 with $\lambda_1^k = 1 \; \forall k$, different penalty parameters ρ, **linear cost function** $\psi_i(x) = x$, feasibility residual tolerance 10^{-3} and duality gap tolerance 10^{-3}.
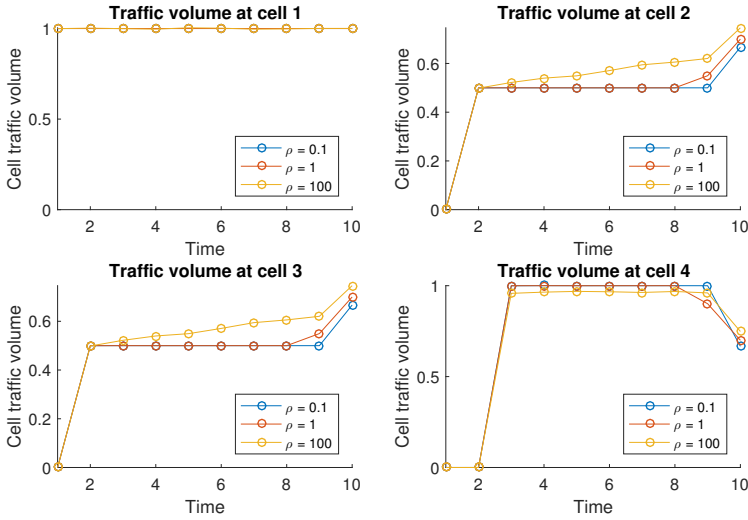
Figure 5.9   Cell traffic volumes for test problem 1 with $\lambda_1^k = 1 \ \forall k$, different penalty parameters ρ, **linear cost function** $\psi_i(x) = x$, feasibility residual tolerance 10^{-3} and duality gap tolerance 10^{-3}.

penalty parameter ρ	0.001	0.01	0.1
number of iterations	334541	33467	3356
computation time [s]	4149	330	26
relative cost err. ε_ψ	45×10^{-6}	45×10^{-6}	44×10^{-6}
mean cell vol. err. $\bar{\varepsilon}_y$	57×10^{-6}	56×10^{-6}	51×10^{-6}
max cell vol. err. $\varepsilon_y^{\max}$	461×10^{-6}	458×10^{-6}	431×10^{-6}
penalty parameter ρ	1	10	100
number of iterations	352	24015	> 500000
computation time [s]	3	211	6754
relative cost err. ε_ψ	2.3×10^{-3}	155×10^{-3}	127×10^{-3}
mean cell vol. err. $\bar{\varepsilon}_y$	7.5×10^{-3}	105×10^{-3}	84×10^{-3}
max cell vol. err. $\varepsilon_y^{\max}$	99×10^{-3}	414×10^{-3}	246×10^{-3}

Table 5.5   Results for test problem 1 with $\lambda_1^k = 1$ for all k, different penalty parameters ρ, **quadratic cost function** $\psi_i(x) = x^2$, feasibility residual tolerance 10^{-3} and duality gap tolerance 10^{-3}.
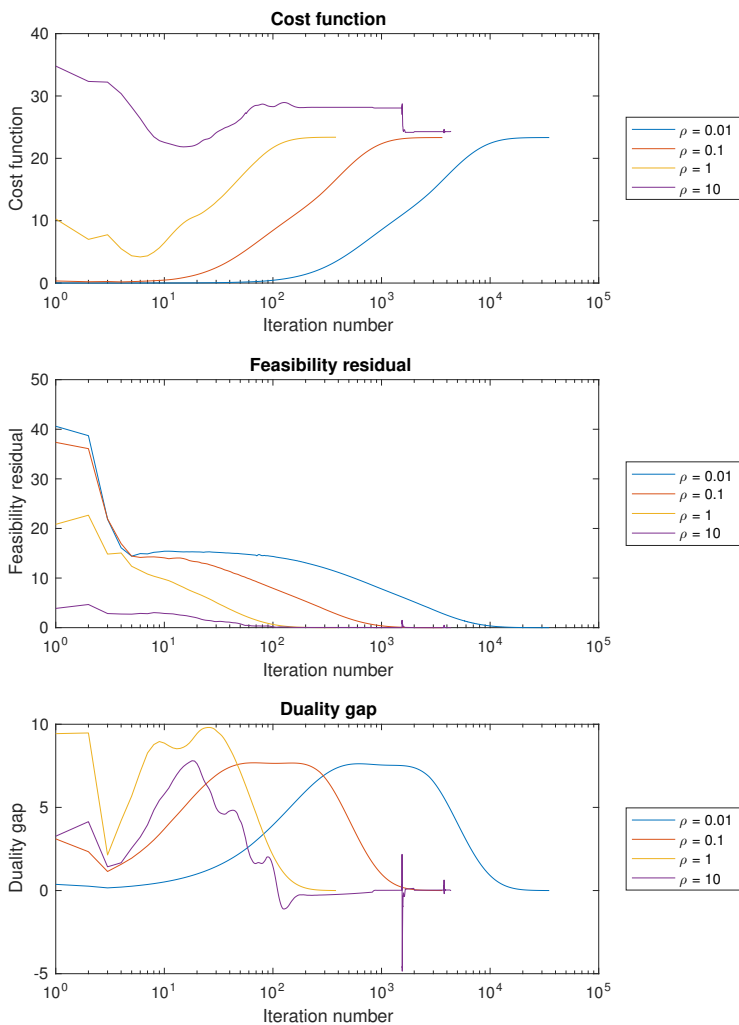
Figure 5.10 Cost function, feasibility residual and duality gap for test problem 1 with $\lambda_1^k = 1$ for all k, different penalty parameters ρ, **quadratic cost function** $\psi_i(x) = x^2$, feasibility residual tolerance 10^{-3} and duality gap tolerance 10^{-3}.
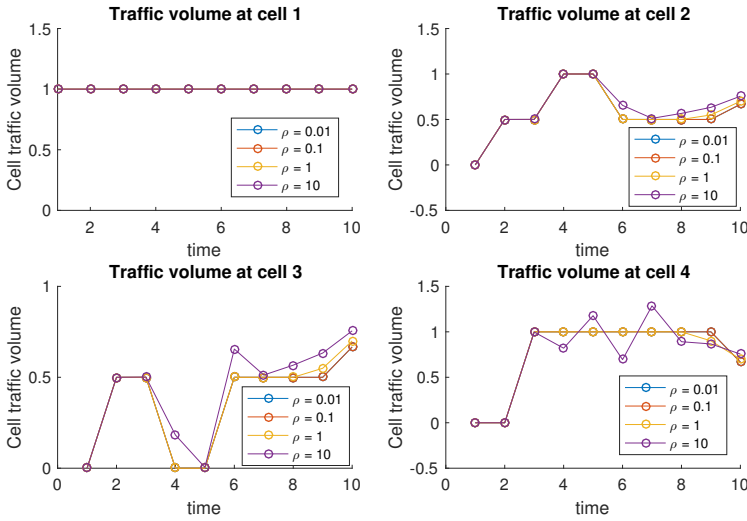
Figure 5.11   Cell traffic volumes for test problem 1 with $\lambda_1^k = 1$ for all k, different penalty parameters ρ, **quadratic cost function** $\psi_i(x) = x^2$, feasibility residual tolerance 10^{-3} and duality gap tolerance 10^{-3}.

penalty parameter ρ	0.01	0.1	1	10
number of iterations	35017	3632	381	4373
computation time [s]	262	27	4	32
relative cost err. ε_ψ	33×10^{-6}	42×10^{-6}	2.2×10^{-3}	42×10^{-3}
mean cell vol. err. $\bar{\varepsilon}_y$	59×10^{-6}	41×10^{-6}	7.5×10^{-3}	60×10^{-3}
max cell vol. err. $\varepsilon_y^{\max}$	387×10^{-6}	337×10^{-6}	99×10^{-3}	306×10^{-3}

Table 5.6   Results for test problem 1 **modified** such that $s_3^4(x) = s_3^5(x) = 0$ with $\lambda_1^k = 1$ for all k, different penalty parameters ρ, **quadratic cost function** $\psi_i(x) = x^2$, feasibility residual tolerance 10^{-3} and duality gap tolerance 10^{-3}.

Figure 5.12 Cost function, feasibility residual and duality gap for test problem 1 **modified** such that $s_3^4(x) = s_3^5(x) = 0$ with $\lambda_1^k = 1$ for all k, different penalty parameters ρ, **quadratic cost function** $\psi_i(x) = x^2$, feasibility residual tolerance 10^{-3} and duality gap tolerance 10^{-3}.

Figure 5.13   Cell traffic volumes for test problem 1 **modified** such that $s_3^4(x) = s_3^5(x) = 0$ with $\lambda_1^k = 1$ for all k, different penalty parameters ρ, **quadratic cost function** $\psi_i(x) = x^2$, feasibility residual tolerance 10^{-3} and duality gap tolerance 10^{-3}.

5.5   Results from Test Problem 2

The result for test problem 1 with a constant inflow λ_1 and a linear cost function $\psi_i(x) = x$ (Fig. 5.8) shows that the optimization variables do not converge to one feasible optimal solution even for small values of the penalty parameter. However, the same test with a quadratic cost function $\psi_i(x) = x^2$ converges for small ρ-values (Fig. 5.10). Because of this, test problem 2 was tested with the quadratic cost function $\psi_i(x) = x^2$.

The stopping criterion (4.26) used for test problem 2 was the same as for test problem 1, i.e., requiring that both the feasibility residual and the duality gap are less than 10^{-3}. In order to handle cases with very slow convergence, an upper limit of number of iterations was used. For the penalty parameter value $\rho = 0.1$, the limit was 1,000,000 iterations, and for the parameter values $\rho = 1$, 10, 100 and 1000, the limit was chosen as 100,000 iterations. Thus, for the cases where the number of iterations for satisfying the stopping criterion is larger than the limit, the results are presented for the limiting number of iterations. The results from test problem 2 are given in Tab. 5.7. The cost function, feasibility residual and duality gap as a function of number of iterations for the different cases except $\rho = 1000$ are shown in Fig. 5.14. For the case with $\rho = 1000$, the result is not converging and behaves similarly to the red curves in Fig. 5.8. Because of that the oscillating curves in this case make it more difficult to see the other results, this case it not included in the
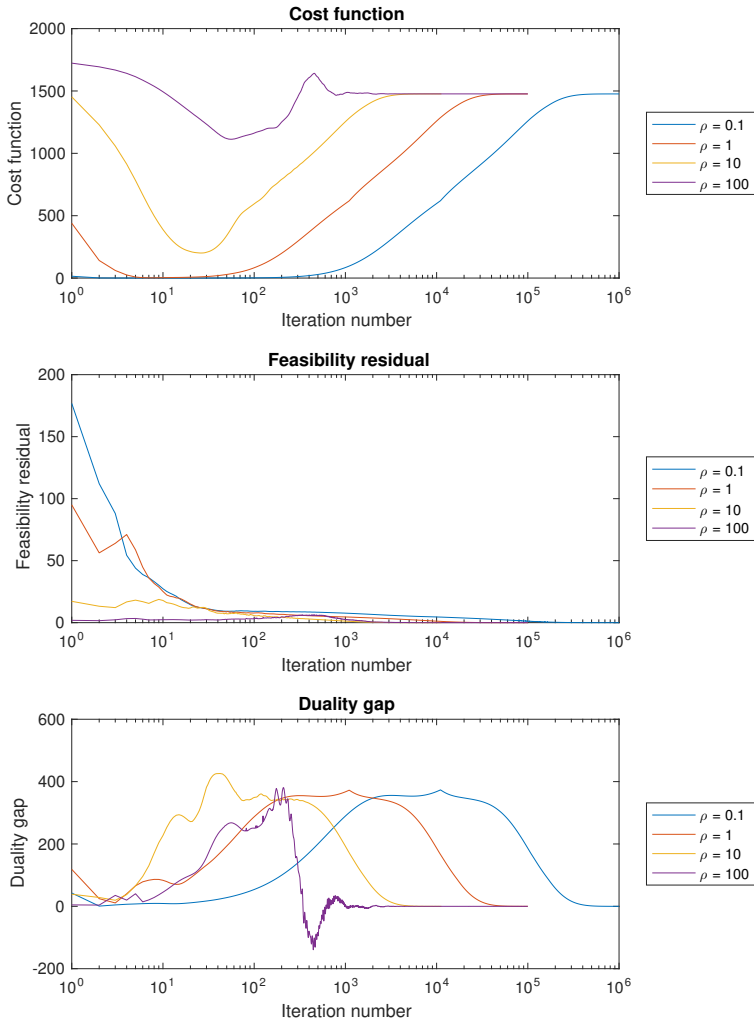
penalty parameter ρ	0.1	1	10
number of iterations	> 1000000	> 100000	11285
computation time [s]	84835	6962	765
relative cost err. ε_ψ	3.5×10^{-6}	34×10^{-6}	13×10^{-6}
mean cell vol. err. $\bar{\varepsilon}_y$	69×10^{-6}	674×10^{-6}	6.7×10^{-3}
max cell vol. err. $\varepsilon_y^{\max}$	497×10^{-6}	4.8×10^{-3}	47×10^{-3}
feasibility res.	2.9×10^{-6}	29×10^{-6}	5.6×10^{-6}
duality gap	5.1×10^{-3}	5.2×10^{-3}	1.0×10^{-3}
penalty parameter ρ	**100**	**1000**	
number of iterations	> 100000	> 100000	
computation time [s]	6837	6799	
relative cost err. ε_ψ	168×10^{-6}	145×10^{-3}	
mean cell vol. err. $\bar{\varepsilon}_y$	12×10^{-3}	682×10^{-3}	
max cell vol. err. $\varepsilon_y^{\max}$	170×10^{-3}	2.4	
feasibility res.	10×10^{-3}	4.5	
duality gap	16×10^{-3}	-325	

Table 5.7 Results for test problem 2 with different penalty parameters ρ, **quadratic cost function** $\psi_i(x) = x^2$, feasibility residual tolerance 10^{-3} and duality gap tolerance 10^{-3}.
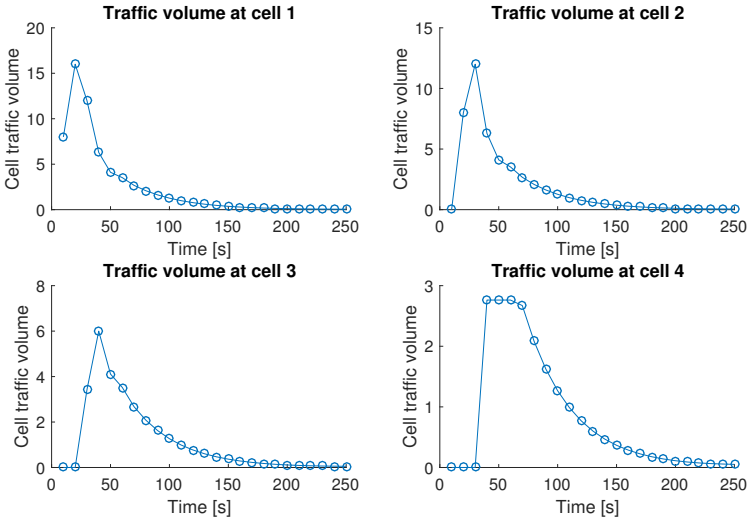
figure. Some of the cell traffic volumes obtained for $\rho = 1$ are presented in Fig. 5.15. For the other converging cases, i.e., all cases tested except $\rho = 1000$, the resulting cell volumes are similar enough to the case of $\rho = 1$ to be almost indistinguishable, which is why only the latter case is plotted.

Finally, the interpretation of the result in terms of turning ratios and control parameters α, i.e., ramp metering or speed regulation, was computed according to the end of Section 3.4. According to the equations in this section, the calculations are different depending on whether the demand function or the total outflow is zero or greater than zero. Since these values sometimes can be slightly larger than zero even if they should be zero, due to numerical inaccuracy, a tolerance must be chosen such that the value in question is considered as zero if it is smaller than this tolerance. In the present case, the limit 10^{-3} was used for this purpose, which can be compared to the average flow on the links during the simulation and the maximum flow at any link which are 0.07 and 1.2 respectively. The resulting turning ratios for the outlinks from diverge cells are illustrated in Fig. 5.16 and the resulting control parameters α_i for all cells are presented in Fig. 5.17.
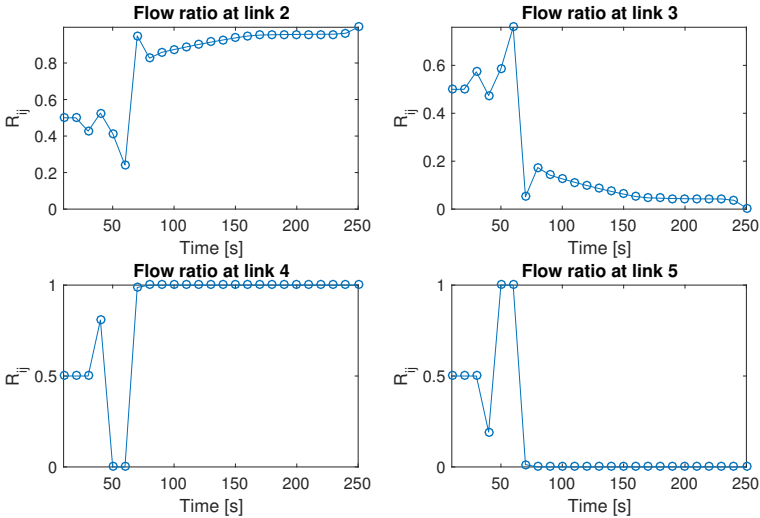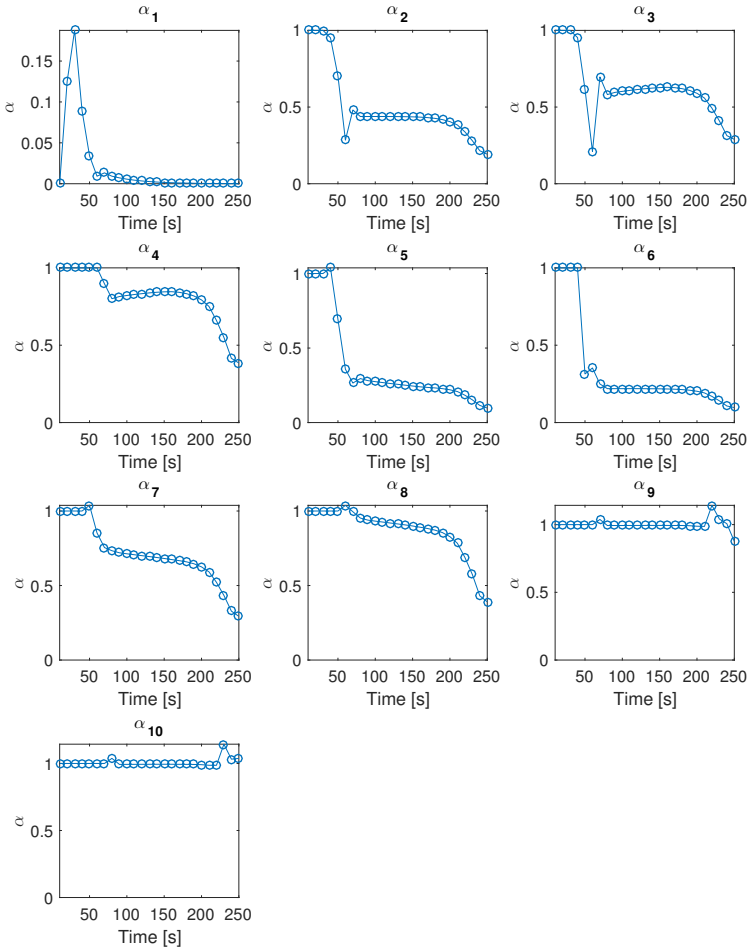
Figure 5.14 Cost function, feasibility residual and duality gap for test problem 2 with different penalty parameters ρ, **quadratic cost function** $\psi_i(x) = x^2$, feasibility residual tolerance 10^{-3} and duality gap tolerance 10^{-3}.

Figure 5.15   Cell traffic volumes in some of the cells for test problem 2 with penalty parameter $\rho = 1$, **quadratic cost function $\psi_i(x) = x^2$**, feasibility residual tolerance 10^{-3} and duality gap tolerance 10^{-3}.

Figure 5.16   Turning ratios for test problem 2 with penalty parameter $\rho = 1$, **quadratic cost function $\psi_i(x) = x^2$**, feasibility residual tolerance 10^{-3} and duality gap tolerance 10^{-3}.

Figure 5.17   Control parameters α_i for test problem 2 with penalty parameter $\rho = 1$, **quadratic cost function** $\psi_i(x) = x^2$, feasibility residual tolerance 10^{-3} and duality gap tolerance 10^{-3}.

6

Discussion

This chapter contains a discussion about both the simulation results from the test problems and about the method used in general. **Section 6.1** and **Section 6.2** discuss the results obtained in Chapter 5. In **Section 6.3**, the dynamic traffic flow optimization algorithm is discussed in general.

6.1 Test Problem 1

Unit Pulse Inflow

The results from test problem 1 with unit pulse inflow ($\lambda_1^1 = 1$, $\lambda_1^k = 0$ for all $k \geq 2$) and linear cost function, as reported in Tab. 5.2, show that the obtained cost value is very close to the optimal cost obtained by CVX for all tested penalty parameter values ρ. The largest relative cost error is 0.02 %, which is obtained for $\rho = 10$. In Fig. 5.4 it can be seen that the cost function goes to the same solution, and that the duality gap and feasibility residual go to zero in all cases. For smaller ρ-values, more iterations are required, but besides this, the behaviour is very similar for all small ρ-values. This is in accordance with the expectations, since ρ is the size of the step taken in the domain of the dual function in the attempt to maximize this function. Thus, if ρ is reduced to a tenth of its previous value, then ten times as many steps should be needed in order to reach the maximum of the dual function.

The cases that differ most from the rest are the ones corresponding to the largest penalty parameter values, 10 and 100. For $\rho = 100$, the cost function ends up very close to the optimal value already in the first iteration. One fact that might contribute to this result is that in the special case of test problem 1, the primal variables are initiated such that they in fact are equal to the optimal values, while the dual variables are initiated as zero (according to the end of Section 4.1). Thus, the starting point in the primal variable domain is the optimal point, which probably simplifies the process of finding the optimal point in the dual variable domain, and makes it possible to do so even for a large step size. Another reason that this is possible is probably that the structure of test problem 1 is very simple and symmetric. It is reasonable that the simple structure of the problem implies that the dual function

gets a simple and more regular structure as well (compared to a more complicated problem), which makes it more probable to get a converging result even for a large step size.

The obtained cell traffic volumes in Fig. 5.5 are as expected. Since the supply functions are large enough to allow for the entire unit flow to enter any cell, the entire flow unit will leave cell 1 already between time points 1 and 2, and enter the connected cells 2 and 3. At the next time step, all flow in these cells will enter cell 4. In the last time step, the entire flow leaves the network from cell 4, which is the only sink. The flow dynamics is thus such that all flow moves one step closer to the sink at each time step, which is the fastest possible way that all flow can leave the network, and thus an optimal solution.

When the linear cost function was replaced with a quadratic cost, the results in Tab. 5.3 were obtained. Here, it is clear that larger penalty parameter values give less good results. For example, $\rho = 10$ results in a cost value with a 2 % deviation from the real optimum, while the maximum cell volume error is as much as 11 %. On the other hand, the corresponding errors for $\rho = 0.001$ and $\rho = 0.01$ are less than 0.1 %. It can be seen in Fig. 5.6 that the cost function, feasibility residual and duality gap behave almost identically for the three smallest penalty parameter values, but with a scaling of the number of iterations such that they increase with the same proportion as the penalty parameter decreases. This consistency indicates that the algorithm works well for the tested case as long as the penalty parameter is small enough. In Fig. 5.7, it can be seen that also the obtained cell traffic volumes are consistent for the small penalty parameter values 0.001 and 0.1.

The less good performance for large penalty parameters is reasonable, since a large step in the dual variable domain might imply that the dual function has started to decrease at the end of the step even though it was increasing at the beginning of the step. That is, it is possible to move too far and end up beyond the maximal function value in the step direction.

Constant Inflow

For test problem 1 with a constant inflow ($\lambda_1^k = 1$ for all k) and a linear cost function, the results in Tab. 5.4 were obtained. It can be seen that the method converges, in the sense that the stopping criterion is satisfied, only for the large penalty parameters $\rho = 10$ and $\rho = 100$. However, the results for these large step sizes cannot be trusted, since, e.g., the case with $\rho = 10$ gives an error in the cost function of 12 %.

The behaviour of the cost, feasibility residual and duality gap can be seen in Fig. 5.8 for some ρ-values. It is clear that all these quantities start to experience increasing oscillations after a certain number of iterations, even for small penalty parameters. A probable explanation for this is the following: For the problem considered, there are infinitely many optimal solutions, since the outflow from cell 1 can be arbitrarily distributed between cells 2 and 3, as long as the supply constraints are satisfied, and still give the same cost function value. Thus, in the different steps

of one algorithm iteration, which consist of updating the different variables, the different variables can be chosen such that they belong to different optimal solutions. Thereby, the algorithm jumps between points that are closer to different optimal solutions for different variables, which can give raise to an oscillating behaviour.

Inspecting the obtained cell traffic volumes for some penalty parameter values in Fig. 5.9, it can be seen that the solutions for $\rho = 0.001$ and $\rho = 0.1$ are not optimal, since the volumes at cell one and cell 4 then should be 1 for most time points. However, for the case of $\rho = 100$, a solution that is close to an optimal solution is found. As discussed for the case of the unit pulse inflow, this probably depends on that the particular problem has a quite simple structure and is initiated with optimal primal variables. If the algorithm would have proceeded as desired, the same result should be obtained for smaller penalty parameter values, as this would correspond to dividing the step in the dual variable domain into several smaller steps. Thus, it is reasonable to assume that the result in this case is an effect of the simplicity of the particular problem, and that the result for a more complicated example would be bad also for large penalty parameters. The conclusion from this test is that the derived algorithm cannot be trusted for linear cost functions, since the result is not consistent for small penalty parameter values. This is probably due to that the these cost functions are not strictly convex, which implies that there is not a unique optimum. Thereby, the focus will in the remaining tests be on the case of the quadratic cost function.

The same problem as just considered but with a quadratic cost gave the results in Tab. 5.5. It can be seen that the result for the penalty parameters $\rho = 0.001, 0.01$ and 0.1 give both cost function errors of less than 0.01 % and maximum cell volume errors of less than 0.1 %. Thus, the method seems to work fine for this case, as long as the penalty parameter is chosen small enough. Also the case of $\rho = 1$ gives a result that is quite accurate in terms of the discussed error quantities, but the tests with larger penalty parameters result in large errors.

Inspecting the cost, feasibility residual and duality gap in Fig. 5.10 it shows that the behaviours of these quantities are very similar for the ρ-values lower than or equal to 0.1 except for that they are differently scaled. The results for these cases also result in almost exactly the same cell traffic volumes, coinciding with the blue curves in Fig. 5.11. In this figure, it can also be seen the case of $\rho = 1$ gives a result which is quite close to the results for smaller penalty parameters, but different enough to be distinguishable.

The conclusion it that the algorithm seems to work fine for solving test problem 1 with constant inflow and quadratic cost, as long as the penalty parameter is around 0.1 or smaller.

Modified Test Problem 1

When test problem 1 was modified to simulate an incident that blocks inflow to cell 3 in the time interval $t \in [3,5)$, by changing two supply function values to $s_3^4(x) = 0$

and $s_3^5(x) = 0$, the results for some different penalty parameters were as shown in Tab. 5.6. It is seen that for both $\rho = 0.01$ and $\rho = 0.1$, the relative cost errors are again less than 0.01 % and the maximum cell volume errors are less than 0.1 %. The corresponding cost functions, feasibility residuals and duality gaps behave again consistently, according to Fig. 5.12.

The resulting cell traffic volumes in Fig. 5.13 for the two smallest ρ-values are reasonable and as expected. Since the inflow to cell 3 must be zero in the time interval $t \in [3,5)$, the cell volume cannot increase between time points 3 and 4 as well as 4 and 5. As seen in the result, the algorithm manages to take this into account. The outflow from cell 1 is during this interval instead redirected to cell 2, which gives a feasible and optimal solution.

The results from the modified problem 1 show that the algorithm seems to work fine even for this slightly more complicated case with a time-varying supply function, when a quadratic cost function is used, as long as the penalty parameter is 0.1 or smaller.

6.2 Test Problem 2

When applying the algorithm to the more complicated test problem 2, the results were as shown in Tab. 5.7. When running 1,000,000 iterations with penalty parameter $\rho = 0.1$, the relative cost error was very small (0.0004 %) and the maximum cell volume error was 0.05 %. For 100,000 iterations with $\rho = 1$, both these quantities were about ten times as big. The case $\rho = 10$ with a bit more than 10,000 iterations gave again a ten times larger maximum cell volume error, i.e., 5 %, but a very small cost error. Thus, all these three parameter values, and especially the first two, give approximately the same solution. The similar behaviours for the cost function, feasibility residual and duality gap can be seen in Fig. 5.14.

The resulting cell traffic volumes for some of the cells are shown in Fig. 5.15. This result is reasonable since a large external inflow to cell 1 is present at the first 30 seconds, which leads to a large volume in this cell during this time interval. The inflow is 0.8 vehicles/s the first ten seconds, then 1.6 vehicles/s for ten seconds and again 0.8 vehicles per second during ten seconds. This explains the peak for the traffic volume after 20 seconds. After the first two time steps of 10 seconds (recall that the time discretization interval is $h = 10$ s), the traffic reaches cell 2 (see Fig. 5.3), after 30 seconds it has reached cell 3, and so on.

The traffic volume at cell 4 is in agreement with the time-dependent capacity C_4^k in Tab. 5.1. According to this, and the supply and demand functions (5.4) and (5.5), there cannot be any in- or outflow to or from cell 4 during the time interval $t \in [40,60)$, and the same flows are limited more than at the beginning for times $t \in [60,80)$. Thus, the fact that the cell volume is constant in the time interval $t \in [40,60)$ shows that the algorithm handles this requirement.

The resulting turning ratios shown in Fig. 5.16 are reasonable. Initially, when

the cell traffic volume in cell 2 is large, the outflow is divided approximately equally between link 2 and link 3 (see Fig. 5.3), which is reasonable since the flow is too large to send along the shortest path along links 2, 4 and 9 only, and thus should be divided so that flow is sent also along the longer path along links 3, 7, 8 and 10. When the flow is blocked at cell 4, the flow must go either along links 2, 5 and 6 or along links 3 and 7. Since the latter path is shorter, most of the flow is directed along this path during the time interval $t \in [40, 60)$. When cell 4 is unblocked again, the flow in the first part of the network has decreased (since external flow only is supplied during the first 30 seconds). Thus, one of the paths between cells 1 and 10 is sufficient for sending almost all of the remaining flow. Thus, the shortest path, along links 2, 4 and 9, is chosen for this. This is why almost all flow from cell 2 is directed along link 2 and almost none of it along link 3 after around 100 seconds.

The flow ratios for link 4 and 5 are also reasonable. First, when the flow is large, the outflow from cell 3 is equally distributed among the two links, in order to utilize the entire capacity of the network. When cell 4 is blocked, all flow is naturally directed along link 5, and after this, when the flow is small, all flow is directed along link 4 since this corresponds to the shortest path.

The control parameters in Fig. 5.17 also seem to be quite reasonable for the first half of the simulation time interval. The first one, α_1, is a measure of how large flow is allowed to leave cell 1. This follows from the optimal traffic volume at cell 1, shown in Fig. 5.15. The remaining control parameters correspond to speed limitations. The value $\alpha_i = 1$ means that the speed is at the maximum allowed speed for cell i, while smaller values mean that the speed is limited with the corresponding proportion (since the demand functions are linear). These control parameters are at 1 initially, since no traffic flow has reached the cells yet, which implies that the speed can be at its maximum.

As an example of an interpretation of the control parameters, it can be seen that α_6 for most times is much smaller than α_4. This is reasonable, since the outflow from cell 3 preferable should go to cell 4 instead of cell 6, because of the fact that first alternative implies a shorter itinerary to the sink cell (after the times when cell 4 is blocked). Thus, the traffic flow travelling through cells 6, 7 and 8 should be slowed down compared to the traffic travelling through cell 4, since both these flows compete about the space in cell 9, while the first alternative corresponds to a shorter and more optimal path. This implies that the vehicles at the shorter path are prioritized and allowed to drive faster compared to the vehicles along the longer path, so that most of the flow and thus most of the vehicles can travel along the shortest path.

Furthermore, the control parameters for cells 9 and 10, i.e., α_9 and α_{10}, are always 1, which is reasonable since there is no reason to slow down the traffic in these cells. The reason that the traffic sometimes is slowed down is in order to allow vehicles along another path ending up at the same end cell to be prioritized. However, since there only is one inlink to cell 10, there is no reason to limit the speed in this case. The reason that the parameters sometimes are slightly larger than

1 is probably due to numerical inaccuracy. In this case, they should reasonably be assumed to be 1.

From these observations, the values of the control parameters seem to be reasonable in most aspects. However, at the latter half of the simulation time interval, the control parameters are for most cells decreasing. This means that the speed is decreased much even though it is known that the flow is small for these times. This is probably a consequence of the combination of a quadratic cost function and a quite short time horizon (number of time points K) used in the optimization. This combination implies that it might be more optimal to retain small traffic volumes in some cells during the simulated time horizon rather than to let all flow pass through the network, which would result in larger cell volumes and a cost that increases quadratically. This effect is clearly seen in the plot of the cell volumes in Fig. 5.15. The conclusion of this is that the control parameters for the latter half of the time interval probably are not very practically useful. In order to improve this result, a larger time horizon should be chosen, possibly in combination with a better cost function (e.g., one that is more similar to the linear function). A way to handle the current result if it would be used in practice could be to simply ignore the control parameters and instead use the maximum speed when the traffic volume in the cell in question is very small.

6.3   General Discussion

The derived and tested algorithm (4.3)–(4.8) is based on a generalization of ADMM, which is described in [Boyd et al., 2011, Ch. 3], and differs from this mainly in two ways.

Firstly, the primal update step in ADMM consists of two steps only, while it here consists of four independent steps (five steps, but where the updates of μ and g can be considered as the same step, since both of these are outflows and they are independent of each other since just one of them is non-zero for each cell and thus for each subproblem). This implies that the theoretical background such as convergence proofs and similar conclusions that hold for ADMM not necessarily applies to the derived algorithm. However, the idea behind the approach is the same as for ADMM, i.e., proceeding according to the method of multipliers but with the difference that the Lagrangian minimization step is carried out as several steps instead of minimizing the augmented Lagrangian with respect to all primal variables simultaneously. This yields an approximation of the method of multipliers that presumably should give similar results if the changes of the variables in each iteration, i.e., the dual step length ρ and thus also the remaining variables, are small.

Secondly, the adapted algorithm contains not only equality constraints but also inequality constraints, which complicates the procedure by requiring some of the dual variables to be non-zero. However, when this is taken into account, the result seems as a quite natural extension of ADMM.

A question that has been ubiquitous during the tests of the algorithm is how the penalty parameter ρ should be chosen. A large value of ρ implies that deviations from the constraints are penalized much in the cost function of the augmented Lagrangian (thus the name 'penalty parameter'), which should be good in order to find a feasible solution faster. However, it also implies that the step length in the dual variable domain during the dual update is large. As have been seen in the results, this can cause oscillations and hinder the algorithm from converging to the correct solution. For all tests with quadratic cost functions, the optimal solution has been found given that ρ is small enough. Thus, the conclusion is that it is better to choose ρ smaller than to choose it large. The price to be paid for this is that the algorithm requires more iterations and thus more computing time to find the solution.

7

Conclusions and Future Work

7.1 Conclusions

In this work, a distributed algorithm, (4.3)–(4.8), for solving the optimal convex dynamic traffic flow control problem (3.19) has been derived. This problem, in turn, is a relaxation of the dynamic traffic flow control problem (3.18), and the optimal solutions for the convex problem supplied by the algorithm can be mapped to feasible points for this problem.

The derived algorithm does not in general work well with cost functions that are linear in the cell traffic volumes, i.e., $\psi_i(x) = x$. However, for quadratic cost functions, $\psi_i(x) = x^2$, the algorithm converges to the optimal solution for the tested cases, given that the penalty parameter (which equals the dual ascent step length) is chosen small enough. This is probably due to that the quadratic cost is strictly convex in the cell traffic volumes, while the linear cost is not. A drawback with the algorithm is that many iterations are required in order to obtain accurate results, which is a known drawback also with the ADMM that was the starting point, according to [Boyd et al., 2011, Ch. 3]. The number of iterations required increases with decreasing penalty parameter values, which necessitates a trade-off between convergence and computational speed in the choice of this parameter.

The tests show that the optimization problem with a quadratic cost function can be solved for quite complicated problems. Even cases when the problem contains time-varying supply and demand functions can be handled. Thus, problems where the circumstances of the road network are time-varying can be solved.

7.2 Future Work

The most important step in further developing the results from this work, in order to obtain a practically useful method, is to modify the method such that the number of iterations is reduced. The algorithm converges consistently to the optimal solution

for the cases with quadratic cost function and small penalty parameter. However, very many iterations are needed in order to reach the optimum, which is associated with a large computational time. One approach that could be tested in order to speed up the algorithm is to vary the step size, such that one starts with large steps and decreases the step size when the optimum is approached. Another idea could be to try to incorporate second order information in the algorithm. In the derived algorithm, the dual update uses only the gradient, and is thus a first order method. If it would be possible to in some way use information about the exact or approximate second derivatives, this could be useful for achieving faster convergence.

Another aspect that should be studied further is the theoretical aspects of the algorithm. For example, it would be desirable to prove that the algorithm always converges if some conditions are satisfied. The tests in this work indicate that convergence seems to be obtained for quadratic cost functions and small penalty parameters, but this has not yet been proved.

Bibliography

Ba, Q., K. Savla, and G. Como (2015). "Distributed optimal equilibrium selection for traffic flow over networks". In: *IEEE Conference on Decision and Control*, pp. 6942–6947.

Ba, Q. and K. Savla (2016). "On distributed computation of optimal control of traffic flow over networks". *Fifty-fourth Annual Allerton Conference, Allerton House, UIUC, Illinois, USA*.

Boyd, S. and L. Vandenberghe (2009). *Convex Optimization*. Cambridge University Press.

Boyd, S., N. Parikh, E. Chu, B. Peleato, and J. Eckstein (2011). "Distributed optimization and statistical learning via the alternating direction method of multipliers". *Found. Trends Mach. Learn.* **3**:1, pp. 1–122.

Como, G., E. Lovisari, and K. Savla (2016). "Convexity and robustness of dynamic traffic assignment and freeway network control". *Transportation Research Part B: Methodological* **91**, pp. 446–465.

CVX Research Inc. (2017). *CVX: matlab software for disciplined convex programming, version 2.1*. `http://cvxr.com/cvx`.

Daganzo, C. F. (1994). "The cell transmission model: A dynamic representation of highway traffic consistent with the hydrodynamic theory". *Transportation Research B: Methodological* **28B**:4, pp. 269–287.

Seibold, B. (2015). *A mathematical introduction to traffic flow theory*. `http://helper.ipam.ucla.edu/publications/tratut/tratut_12985.pdf`.

UN (2014). *World's population increasingly urban with more than half living in urban areas*. `http://www.un.org/en/development/desa/news/population/world-urbanization-prospects-2014.html`.

Whittle, P. (2007). *Networks: Optimisation and Evolution*. Cambridge University Press.

Title and subtitle

Distributed Control of Dynamic Flows in Traffic Networks

Abstract

 In today's society, traffic congestion is a major problem in several aspects. Apart from the obvious problem that people are losing valuable time due to the resulting delays, it also has negative impact on as well the economy as the local and global environment. With the development of sensors and navigation support, it has now become possible and thus of interest to study optimal routing of vehicles in a traffic network, in order to reduce the congestion-related problems.

 In this master's thesis, a distributed algorithm for solution of optimal dynamic traffic flow control problems is derived, implemented and tested. Traffic networks are modelled with the cell transmission model (CTM), and the solution algorithm is based on a generalization of the alternating direction method of multipliers (ADMM).

 The algorithm is tested for one simple and one more complicated traffic network. The tests include both cases with time-varying external inflow of traffic as well as cases where the flow capacity of a specific road segment is varied with time, in order to simulate temporary traffic incidents.

 The tests show that if the cost function is chosen as the sum of squares of the traffic volumes at the cells (road segments) of the network, the algorithm converges to the optimal solution if a specific parameter (the penalty parameter, or step length) is chosen sufficiently small.

 The report starts with a description and examples from the simpler case of static traffic flow optimization. It also contains a summary of the concepts used from optimization theory. After this, the approach for dynamic traffic flow modelling and optimization is described. Finally, a description and derivation of the algorithm is provided, after which the implementation is tested for different cases involving the two different traffic networks.