

FLOWS OVER TIME
WITH
FLOW-DEPENDENT TRANSIT TIMES

vorgelegt von
Dipl.-Math. Katharina Langkau
aus Bonn

Von der Fakultät II – Mathematik und Naturwissenschaften
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften
– Dr. rer. nat. –

genehmigte Dissertation

Berichter: Prof. Dr. Rolf H. Möhring
Prof. Dr. Günter Rote

Tag der wissenschaftlichen Aussprache: 10. September 2003

Berlin 2003
D 83

ZUSAMMENFASSUNG

Transportprobleme werden in der Kombinatorischen Optimierung üblicherweise mit Hilfe von Netzwerkflüssen modelliert. Das klassische Flussmodell ist jedoch statisch und reflektiert daher nur unzureichend die zeitliche Komponente eines Transportproblems.

Ein spezielles Transportproblem entsteht bei der Straßenverkehrlenkung. Neue Technologien, wie z.B. Navigationssysteme, haben ein verstärktes Interesse an einer akkuraten Modellierung und Optimierung von Verkehrsflüssen geweckt. In der vorliegenden Arbeit werden Flussmodelle betrachtet, welche die typischen Eigenschaften von Straßenverkehr widerspiegeln. Hierbei wird auf folgende grundlegende Eigenschaften abgezielt: Im Straßenverkehr verändert sich die Netzbelastung über die Zeit hinweg. Während zur Hauptverkehrszeit eine hohe Netzbelastung herrscht, kann bereits wenige Zeit später eine entspannte Verkehrssituation vorliegen. Ein weiteres Phänomen ist die Abhängigkeit der Fahrzeiten vom Verkehrsfluss. Ein hohes Verkehrsaufkommen zieht lange Fahrzeiten und Verzögerungen nach sich.

Schwerpunkt der vorliegenden Arbeit ist ein Flussmodell, das den beiden genannten Eigenschaften genügt und damit ein, wenn auch vereinfachtes, Abbild des realen Straßenverkehrs darstellt. Die zugrundeliegende Modellannahme ist die Abhängigkeit der Fahrzeit einer Kante von der vorherrschenden Zuflussrate auf der Kante. Innerhalb dieses Modells werden klassische Fragestellungen der Netzwerkflusstheorie behandelt.

Beim „Quickest Flow Problem“ ist ein Netzwerkfluss gesucht, der in minimaler Zeit eine gegebene Flussmenge von einem Startknoten (Quelle) zu einem Zielknoten (Senke) transportiert. Dieses Problem ist bereits \mathcal{NP} -schwer, und folglich existiert vermutlich kein polynomialer Lösungsalgorithmus. Jedoch lassen sich in polynomialer Zeit approximative Lösungen, d.h. zulässige Lösungen beweisbarer Güte, berechnen. Ein wesentlicher Bestandteil der vorliegenden Arbeit ist die Herleitung solcher Approximationsalgorithmen.

In einem ersten Schritt wird eine geeignete, polynomial lösbare Relaxierung des Modells vorgestellt. Diese beruht auf einer Expansion des ursprünglichen Netzwerks zu einem Netzwerk, in dem die Fahrzeiten nicht mehr explizit flussabhängig sind sondern konstant. Im wesentlichen tritt jede ursprüngliche Kante im expandierten Netzwerk vielfach auf. Jede Kopie repräsentiert eine andere konstante Fahrzeit auf der Kante. Durch geschickte Wahl der Kapa-

zitäten sind die auftretenden Fahrzeiten nur noch indirekt flussabhängig.

Basierend auf dieser Relaxierung werden in einem zweiten Schritt Approximationsalgorithmen entwickelt. Insbesondere wird auch das Mehrgüterflussproblem behandelt. Hierbei sind mehrere Quelle-Senke Paare gegeben und gesucht ist ein Fluss, der den Bedarf jedes einzelnen Quelle-Senke Paares deckt. Für dieses Problem wird ein voll polynomiales Approximationsschema entwickelt, d.h. ein Lösungsverfahren, das zulässige Lösungen berechnet, die eine optimale Lösung beliebig genau annähern. Da das „Quickest Flow Problem“ \mathcal{NP} -schwer ist, ist dieses Ergebnis aus Komplexitätstheoretischer Sicht vermutlich nicht verbesserbar.

ACKNOWLEDGEMENTS

About three years ago I applied for a PhD position in the European Graduate Program “Combinatorics, Geometry, and Computations”. Now, my thesis has taken shape and I would like to thank several people for their support.

In the first place, I am grateful to Rolf Möhring for supervising my thesis. He helped me to find the right topic and fully supported my research. At the same time he allowed independent work and development. I benefited a lot from the possibility of visiting international workshops and conferences and I am thankful for Rolf Möhring’s trust.

It was a real pleasure to work together with Alex Hall, Ekkehard Köhler, and Martin Skutella. In particular, I wish to thank Ekki and Martin who gave me directions in asking the right questions and finding the right answers.

The European Graduate Program “Combinatorics, Geometry, and Computations” offered me numerous opportunities to widen my horizon also in other fields of Mathematics. The weekly lectures, schools, courses, and workshops all over Europe were interesting, instructive, and, moreover, lots of fun. The financial support was granted by the German National Science Foundation (DFG) (grant GRK 588/2).

My special thanks go to András Frank for giving me the opportunity to join his research group at the Eötvös Loránd University, Budapest. He aroused my interest in various graph theoretical problems and always had time for inspiring discussions.

I wish to thank Martin Skutella, Marc Pfetsch, Ekkehard Köhler, Nicole Megow, Ines Spenke, Volker Kaibel, Georg Baier, Sebastian Stiller, and Heiko Schilling for their careful proof-reading of different parts of the manuscript and for their helpful suggestions for improvement. I thank Nadine Baumann, Lydia Franck, and Ewgenij Gawrilow for their support in producing computational results and Marc Pfetsch for providing his dissertation files. Moreover, I thank Günter Rote for his willingness to take the second assessment of this thesis.

Finally, my thanks go to the members of the research groups of Rolf Möhring and Günter Ziegler. Without the daily coffee breaks, work would not have been half as nice.

Berlin, July 2003

Katharina Langkau

CONTENTS

Introduction	1
1 Network Flow Models	7
1.1 Introduction	7
1.2 Static Network Flows	8
1.3 Flows Over Time with Fixed Transit Times	9
1.3.1 Continuous Flows Over Time	10
1.3.2 Discrete Flows Over Time	13
1.3.3 Time-Expanded Graphs	13
1.3.4 Continuous versus Discrete Model	15
1.3.5 Known Results	16
1.4 Flow-Dependent Transit Times	23
1.4.1 Time-Dependent Flows	25
1.4.2 Inflow-Dependent Transit Times	26
1.4.3 Load-Dependent Transit Times	27
1.4.4 Temporally Repeated Flows	28
1.4.5 Dynamic Traffic Assignment	30
2 Quickest s-t-Flows	35
2.1 Introduction	35
2.2 A Relaxation	36
2.2.1 The Bow Graph	36
2.2.2 Relaxation Property of the Bow Graph	37
2.3 Constant Factor Approximations for Quickest Flows	42
2.3.1 Piecewise Constant Transit Times	43
2.3.2 General Transit Times	47
2.3.3 An Improved Result for Concave Transit Times	52
2.3.4 The Fan Graph	58
2.3.5 Convex Transit Times	60
2.3.6 Computing Temporally Repeated Flows	64
2.3.7 Limits of the Static Approach	67
2.4 Complexity	70
2.5 Computational Results	74

3	Quickest Multi-Commodity Flows	87
3.1	Introduction	87
3.2	A Stronger Relaxation	88
3.2.1	The New Bow Graph	89
3.2.2	Relaxation Property of the New Model	90
3.3	A Constant Factor Approximation for Quickest Flows	94
3.3.1	Quickest Weakly Inflow-Preserving Flows	94
3.3.2	The Algorithm	98
3.4	FPTAS	101
3.4.1	Condensed Time-Expanded Graphs	101
3.4.2	The Algorithm	103
3.5	Complexity	113
4	The First In, First Out Property	115
4.1	Introduction	115
4.2	First In, First Out Violations in the Fan Graph	117
4.3	Complexity	118
4.4	Geometrically Increasing Capacities	126
	Bibliography	131
	Symbol Index	137
	Index	139

INTRODUCTION

Today, mobility is an indispensable part of everyday life and people attach great importance to private transport. Everybody is familiar with the unpleasant side effects: congestion, traffic jams, and rush hour characterize urban traffic. This has been a major incentive for road traffic industry to develop new traffic management systems that improve the overall traffic situation. The resulting technological and financial efforts are considerable as it is illustrated by the following example.

In the city of Berlin a traffic management center was established whose aim is to record and evaluate the traffic situation in Berlin. The data gathered shall be used to generate comprehensive traffic information and to improve the traffic situation. The creation and operation of the center will cost €16 million. In return “a total of 50 WebCams and over 200 infrared sensors will be installed in locations such as Potsdamer Platz and the TV Tower Interchange (Dreieck-Funkturm). These will feed into the central Traffic Management Centre computer centre which controls 22 outdoor electronic display units and a network of existing data centres.”¹

As this last sentence indicates, a tremendous amount of data must be processed when operating such systems. In order to turn the collected data into valuable information for traffic forecast, traffic management, or even for route guidance, mathematical models and efficient algorithms are required. Yet, the design and the optimization of realistic traffic models constitute an extremely difficult task. A significant body of literature is devoted to this topic, which is widely referred to as *traffic assignment*.

A standard approach is to interpret road traffic as a network flow, where the streets and street crossings form the arcs and nodes, respectively, of the underlying network. A good model should include the typical characteristics of traffic flow: First of all, road traffic is dynamic, i.e., the amount of traffic on a road changes throughout the day due to fluctuating drivers' requests. An empty road might turn into a highly congested road during rush hour times. In contrast to static traffic assignment, dynamic traffic assignment concentrates on time-varying traffic models, which reflect the dynamic nature of traffic.

The second characteristic of traffic flow is the dependency of travel times

¹<http://www.roadtraffic-technology.com/projects/vmz/>

on the current traffic situation. In general, driving through a congested street takes much longer than driving through an empty street. We use the term *flow-dependent transit times* to express that transit times depend on the flow in one way or another. Bookbinder and Sethi [5] remark in their survey article on the dynamic transportation problem: "Perhaps the most important extension is the case in which the time required for a shipment to reach from source i to sink j is a function of the amount shipped." Describing the exact dependency of transit times on the flow values is a highly nontrivial and open problem. Of course, a fully realistic model must take several parameters into account including density, speed, and flow rate evolving along the road.

The literature on dynamic traffic assignment has not succeeded yet in finding a mathematical model that includes the essential characteristics of traffic flow and that is, above all, tractable. There are hardly any algorithmic techniques known which are capable of providing reasonable solutions even for networks of rather modest size. For problem instances of realistic size, already the solution of mathematical programs relying on simplifying assumptions is in general still beyond the scope of modern computers. As Kaufman, Nonis, and Smith [39] conclude: "Unlike the elegant and complete treatment of the classic static case, the dynamic traffic assignment problem is still largely unexplored, at least from a formal point of view, where even its problem formulation is not clearly understood."

Flow-dependent transit times are relevant in many more applications including evacuation planning, production systems, and communication networks (e. g., the Internet). It seems, however, that road traffic assignment has spawned the most sustainable interest in this topic. Not surprisingly, Hamacher and Tjandra [29] consider dynamic traffic models in the context of evacuation planning.

Our work is primarily motivated by applications in road traffic control. Therefore, we usually interpret network flows as road traffic and we measure the quality of a model with respect to its practicality for traffic control purposes. The focus of our research is on a specific network flow model in which flow can change over time and in which transit times are flow-dependent.

The underlying assumption of our approach is that at any moment in time the transit time needed to traverse an arc solely depends on the current rate of inflow into that arc. We will therefore refer to these flows as *flows over time with inflow-dependent transit times*, emphasizing that transit times are considered as functions of the rate of inflow. In the context of road traffic, this assumption means that the time needed to drive through a street is determined when entering the road and it only depends on the number of cars entering the road at that moment in time.

Much of our research is inspired by the seminal work of Ford and Fulk-

erson [18, 19] on network flows over time (also known as dynamic network flows). In this model transit times are fixed. Given a network with capacities and transit times on the arcs, they study the problem of sending a maximal amount of flow from a source node s to a sink node t within a pre-specified time horizon T . Ford and Fulkerson show that this problem can be solved by one minimum cost static flow computation, where transit times of arcs are interpreted as cost coefficients. This result of Ford and Fulkerson has initiated intensive research in this area so that many more classical network flow problems have been considered and solved in the dynamic setting.

Flows over time with inflow-dependent transit times are a direct generalization of flows over time (with fixed transit times); only that the transit time of an arc is not fixed, but depends on the inflow rate of that arc. Unfortunately, flows over time with inflow-dependent transit times are much more difficult to handle: In the quickest s - t -flow problem, we ask for a flow over time that sends a given demand from a source node s to a sink node t as quickly as possible. While for fixed transit times the problem can be solved efficiently (in polynomial time), the problem becomes strongly \mathcal{NP} -hard for inflow-dependent transit times. In particular, we cannot hope for a polynomial time algorithm that computes an optimal solution.

However, flows over time with inflow-dependent transit times still constitute a model that is tractable in the following sense: a solution that approximates the optimal solution within arbitrary precision can be computed efficiently, i.e., in polynomial time.

In this thesis, we develop completely novel techniques to compute such approximate solutions. The techniques rely on a relaxed model of inflow-dependent transit times. The relaxation is defined on an expanded graph with fixed transit times on the arcs. Here, the expansion of each arc in the original graph implicitly models the dependency of transit times on the inflow rate. Since in the expanded graph transit times are fixed, we can apply all methods that are available for flows over time (with fixed transit times). In fact, many of the latter methods again rely on static network flow computations. Therefore, all algorithms presented in this thesis eventually rely on static network flow formulations or on generalizations thereof, which can be solved by linear or convex programming techniques. In particular, our algorithms are simple to implement and are efficient also from a practical viewpoint. Moreover, our techniques can be applied even to the case of multiple source nodes and sink nodes. To our knowledge, we present the first nontrivial model of flow-dependent transit times for which provably good solutions can be computed in polynomial time even in the multi-commodity case.

Surely, the model of inflow-dependent transit times is a simplified traffic

assignment model and only a rough approximation of real-life traffic. For instance, in this model only the rate of inflow into an arc is explicitly bounded; the flow rates evolving along an arc can be arbitrarily large. This is of course unrealistic for road traffic behavior. Moreover, flows over time with inflow-dependent transit times do in general not obey the *first in, first out property* on an arc. This property requires that flow units are entering and leaving an arc in the same order. In the context of road traffic this means that no overtaking is allowed on a street. A violation of the first in, first out property can occur if the inflow rate of an arc drops down quickly. This causes a strong decrease in the transit time of that arc such that some flow units might pass other flow units which are ahead on the arc traveling at a slower speed. However, this phenomenon does not occur in solutions that are generated by our algorithm for the quickest s - t -flow problem. Unfortunately, unrealistic behavior can occur in the multi-commodity flow solutions. Nevertheless, we hope that the model and methods we present, together with the insights we gained when studying this model, give new impulses to the area of dynamic traffic assignment.

Outline of the thesis

This thesis is divided into four chapters.

Chapter 1. In the first chapter we introduce the basic network flow models that are relevant in this thesis including classical (static) network flows, flows over time, and flows over time with inflow-dependent transit times. Besides, we elaborate on some useful techniques developed for flows over time and needed in subsequent chapters. Moreover, we give an overview of the literature on flows over time and on dynamic traffic assignment, respectively.

Chapter 2. In the second chapter we present constant factor approximation algorithms for the quickest s - t -flow problem in the setting of inflow-dependent transit times. First, we introduce a relaxed model of inflow-dependent transit times and show how the original model of inflow-dependent transit times can be embedded into the relaxed model. Based on this relaxation, we develop a $(2 + \varepsilon)$ -approximation algorithm for the quickest flow problem for the case of general arc transit time functions. For the case that all arc transit time functions are concave, we can show that the algorithm achieves an improved performance ratio of $3/2 + \varepsilon$. We present two other approximation algorithms that both achieve performance ratio $2 + \varepsilon$. One of the two algorithms is due to Köhler and Skutella [44] and was originally designed for a model in which the transit time of an arc depends on the load, i.e., the traffic volume, on that arc. The two algorithms both rely on a static convex cost flow formulation

in the original graph. In particular, both algorithms assume that the given transit times are convex.

Additionally, we prove that the quickest s - t -flow problem is strongly \mathcal{NP} -hard in the setting of inflow-dependent transit times. The proof uses a reduction from the \mathcal{NP} -complete problem 3-PARTITION. We close the chapter with a practical evaluation of two of the algorithms mentioned above. All computational experiments are performed on real-life instances given by parts of the Berlin road network. Our computational results confirm the practical usefulness of our algorithms.

Chapter 3. Here we address the quickest flow problem in the case of multiple source nodes and sink nodes. In addition, we consider the problem where arc costs are added. The techniques proposed in Chapter 2 are not powerful enough to handle the more general setting. Therefore, we define a stronger relaxation which reflects flows over time with inflow-dependent transit times more accurately than the model given in Chapter 2. We introduce a technique that converts a flow solution in the relaxed instance into a flow solution to the original problem without losing too much in the objective function value. In particular, we present a $(2 + \varepsilon)$ -approximation algorithm and a fully polynomial approximation scheme for the quickest multi-commodity flow problem with costs. This approach is inspired by the work of Fleischer and Skutella [13, 14, 15] on multi-commodity flows over time with fixed transit times.

Chapter 4. In this chapter we raise a topic that is of great relevance in dynamic traffic assignment: the first in, first out property. We analyze inflow-dependent transit times with respect to this property. More precisely, we consider a slightly simpler model of inflow-dependent transit times that is defined on a generalized time-expanded graph called *fan graph*. This model has been considered, e.g., by Carey and Subrahmanian [9] and Kaufman, Nonis, and Smith [39]. We show that computing a static multi-commodity flow in the fan graph that satisfies the first in, first out property is strongly \mathcal{NP} -hard. The proof uses a reduction from the NP-complete problem 3-SATISFIABILITY. For the single source, single sink case, we present some simple approximation results.

The thesis is intended to be largely self-contained. Nevertheless, we assume that the reader is familiar with the basic concepts of combinatorial optimization including static network flows and linear programming. For an introduction to these topics, we refer to the textbooks of Papadimitriou and Steiglitz [55], Nemhauser and Wolsey [52], Schrijver [62], Ahuja, Magnanti, and Orlin [1], Korte and Vygen [45], Grötschel, Lovász, and Schrijver [26],

and Schrijver [61]. An overview of the literature on network flows over time can be found, for instance, in the survey articles of Aronson [3] and Powell, Jaillet, and Odoni [57], in the PhD thesis of Hoppe [33], and in the article of Fleischer and Skutella [15]. For a comprehensive treatment of complexity theory we refer to the book of Garey and Johnson [23]. An introduction to approximation algorithms in combinatorial optimization is given in the book of Hochbaum [32] and in the book of Vazirani [64].

CHAPTER 1

NETWORK FLOW MODELS

1.1 INTRODUCTION

One look at the table of contents of any textbook on combinatorial optimization reveals to the reader that network flow theory is a fundamental building block of this research area. A second look into the textbook gives an explanation. A rich and concise toolbox of powerful algorithmic techniques has been developed for solving a variety of network flow problems.

In this thesis we study network flows. However, our focus is not on the classical network flow models that are exhaustively covered by most textbooks. We study flow models which capture the essential properties of flows arising in real-life applications such as road traffic control.

The aim of this chapter is to introduce those flow models that are relevant in this thesis. We start with a summary of basic definitions used for classical static flows in Section 1.2. Then, in Section 1.3, we concentrate on time-varying flows. In many optimization problems originating from real-life applications, the factor time is a key ingredient to the problem formulation. In classical network flow theory, however, this factor is not sufficiently reflected. For that reason, we consider network flows over time, which provide an adequate framework for modeling time-dependent and network-structured problems. The model was introduced by Ford and Fulkerson [18, 19] in the late 1950s and since then this topic has become an area of active research.

Motivated by applications in road traffic control, we study time-varying flows with an additional property. Road users are facing this phenomenon in everyday road traffic; the amount of time needed to traverse a street increases as the arc becomes more congested. In Section 1.4, we present network flow models which reflect this dependency. The aim of this thesis is to analyze these models and, hopefully, to advance basic research in areas of applied network flow theory such as road traffic control.

We are considering network flow problems in a directed graph $G = (V, A)$. Each arc $a \in A$ has a positive *capacity* u_a and a nonnegative, nondecreasing, left-continuous *transit time function* $\tau_a : [0, u_a] \rightarrow \mathbb{R}^+$. Moreover, a set of *commodities* $K = \{1, \dots, k\}$ is given; associated with each commodity $i \in K$

is a set of *terminals* $S_i = S_i^+ \cup S_i^- \subset V$. Every *source node* $v \in S_i^+$ has a supply $d_{v,i} \geq 0$ and every *sink node* $v \in S_i^-$ has a demand $d_{v,i} \leq 0$ such that $\sum_{v \in S_i} d_{v,i} = 0$. Every node which is neither a source node nor a sink node is called *intermediate node*. We often consider two special cases: If each commodity $i \in K$ has only one source $s_i \in V$ and one sink $t_i \in V$, we set $d_i := d_{s_i,i}$. If only one commodity is given with a set of terminals $S = S^+ \cup S^-$, we denote the demand (respectively, supply) of a node $v \in S$ by d_v .

Sometimes we consider flows with costs. Then, each arc $a \in A$ has associated *cost coefficients* $c_{a,i} \geq 0, i \in K$, where $c_{a,i}$ is interpreted as the cost (per flow unit) for sending flow of commodity i through the arc. For an arc $a = (v, w) \in A$, let $\text{head}(a) := w$ be the *head node* and let $\text{tail}(a) := v$ be the *tail node* of arc a . For a node $v \in V$, let $\delta^+(v)$ and $\delta^-(v)$ denote the set of arcs leaving and entering v , respectively. We define the *transit time of a path* P in G to be $\tau_P(x) := \sum_{a \in P} \tau_a(x_a)$. If all transit time functions are constant, we denote the transit time of path P simply by τ_P .

1.2 STATIC NETWORK FLOWS

In this section we provide basic definitions and simple facts on classical (static) network flows. For a comprehensive overview on (static) network flows see, e.g., [1, 11, 45]. A *static multi-commodity transshipment* x in G assigns to every arc a and every commodity i a nonnegative flow value $x_{a,i}$ such that *flow conservation* holds:

$$\sum_{a \in \delta^+(v)} x_{a,i} - \sum_{a \in \delta^-(v)} x_{a,i} = 0, \quad \text{for all } i \in K \text{ and } v \in V \setminus S_i. \quad (1.1)$$

The total amount of flow on arc a is denoted by $x_a := \sum_{i \in K} x_{a,i}$. The static flow x is called *feasible* if it obeys the capacity constraints $x_a \leq u_a$, for all arcs $a \in A$. The flow x satisfies supplies and demands if

$$\sum_{a \in \delta^+(v)} x_{a,i} - \sum_{a \in \delta^-(v)} x_{a,i} = d_{v,i}, \quad \text{for all } i \in K \text{ and } v \in S_i. \quad (1.2)$$

The *cost* of a static flow x is defined as

$$c(x) := \sum_{a \in A} \sum_{i \in K} c_{a,i} x_{a,i}.$$

If each commodity $i \in K$ has only a single source s_i and a single sink t_i , then we call x a *multi-commodity flow*. If only one commodity is given, possibly having several sources and sinks, we call x a *transshipment*. Notice that, for

every $i \in K$, the flow given by $x_{a,i}$, $a \in A$, defines a transshipment in G which we denote by x_i .

If only one commodity with a single source s and a single sink t is given, we use the term *s-t-flow*. We define the *value* of an *s-t-flow* x as

$$|x| := \sum_{a \in \delta^+(s)} x_a - \sum_{a \in \delta^-(s)} x_a.$$

A transshipment x in G satisfying all demands d_v , $v \in S$, corresponds to an *s-t-flow* \tilde{x} in a slightly modified graph; introduce a super source s which is connected to every source node $v \in S^+$ by an arc (s, v) of capacity d_v and introduce a super sink t to which every sink node $v \in S^-$ is connected by an arc (v, t) of capacity $-d_v$. Then, the transshipment x in G defines an *s-t-flow* \tilde{x} of value $\sum_{v \in S^+} d_v$ in the modified graph by setting $\tilde{x}_a := x_a$, for $a \in A$, and by setting $\tilde{x}_{(s,v)} := d_v$, for $v \in S^+$, and $\tilde{x}_{(v,t)} := -d_v$, for $v \in S^-$.

Due to this equivalence, literature on static network flows usually only discusses multi-commodity flows and not multi-commodity transshipments. Later, we will see that in a time-varying setting a distinction between both problem settings is reasonable.

It is well-known that, for every *s-t-flow* x , there exists a family of *s-t-paths* \mathcal{P} and a family of cycles \mathcal{C} in G together with nonnegative flow values $(x_P)_{P \in \mathcal{P} \cup \mathcal{C}}$, such that $|\mathcal{P} \cup \mathcal{C}| \leq |A|$ and

$$x_a = \sum_{P \in \mathcal{P} \cup \mathcal{C}: a \in P} x_P,$$

holds, for all $a \in A$. Then, the value of x can be expressed as $\sum_{P \in \mathcal{P}} x_P$. If $\mathcal{C} = \emptyset$, we call $(x_P)_{P \in \mathcal{P}}$ a *path decomposition* of x . Path decompositions can easily be generalized to transshipments; let x be a transshipment in G and let \tilde{x} be the corresponding *s-t-flow* in the modified graph as is explained above. A path decomposition $(\tilde{x}_P)_{P \in \tilde{\mathcal{P}}}$ of \tilde{x} naturally defines a path decomposition $(x_P)_{P \in \mathcal{P} \cup \mathcal{C}}$ of x in G where \mathcal{P} is a set of paths in G and every path $P \in \mathcal{P}$ connects a source node in S^+ to a sink node in S^- . We further generalize the notion of path decompositions to multi-commodity transshipments; a path decomposition of a multi-commodity transshipment x is defined by a set of paths $\mathcal{P} := \cup_{i \in K} \mathcal{P}_i$ in G together with flow values $(x_P)_{P \in \mathcal{P}}$, such that $(x_P)_{P \in \mathcal{P}_i}$ is a path decomposition of x_i .

1.3 FLOWS OVER TIME WITH FIXED TRANSIT TIMES

Ford and Fulkerson [18, 19] introduce flows over time to add a time dimension to the traditional network flow model. The following two aspects of flows over

time distinguish them from the traditional model. Firstly, the flow value on an arc may change over time. This feature is important in applications, where the supplies and demands are not given as fixed measures; instead, they change over time subject to seasonal influences. Naturally, the flow value on each arc should adjust to these changes. Secondly, there is a transit time on every arc which specifies the amount of time flow units need to traverse the arc. As mentioned before, in typical applications not only the flow rate but also the transit times are varying over time. However, in this section we assume that all transit time functions $(\tau_a)_{a \in A}$ are constant. To simplify notation, we let τ_a denote the transit time value on arc a .

Flows over time have been previously referred to as dynamic network flows. Fleischer [16] points out that the term “dynamic” is more consistently used for a problem with input that changes over time. In the context of dynamic flows, the input data is available at the start. It is the flow solution that changes over time. In accordance with [16] and the recent literature on this topic, we use the term “flow over time”. In the model presented by Ford and Fulkerson, time progresses in discrete steps. Research on dynamic flow problems has also pursued another approach where time is assumed to be a continuous measure. Although in this thesis we concentrate on the latter approach, we will introduce and compare both models. As will be shown in Section 1.3.4, the two approaches are essentially equivalent.

1.3.1 Continuous Flows Over Time

A *continuous multi-commodity transshipment over time* f in G is given by Lebesgue-measurable functions $f_{a,i} : \mathbb{R}^+ \rightarrow \mathbb{R}^+$, for every $a \in A$, $i \in K$. Here, the value $f_{a,i}(\theta)$ is the *rate of flow* (per time unit) at which flow of commodity i is entering arc a at time θ . In order to simplify notation, we sometimes use $f_{a,i}(\theta)$ for $\theta < 0$, implicitly assuming that $f_{a,i}(\theta) = 0$ in this case.

The transit time of an arc $a \in A$ is interpreted as the time it takes for flow to traverse a . More precisely, flow which is entering arc a at time θ , arrives at $\text{head}(a)$ at time $\theta + \tau_a$. We say that the flow over time f has *time horizon* T , if no flow is entering an arc $a \in A$ after time $T - \tau_a$, i.e., $f_a(\theta) := \sum_{i \in K} f_{a,i}(\theta) = 0$, for all $\theta \geq T - \tau_a$, $a \in A$.

We require that *flow conservation* holds in f . In our model, we allow storage of flow at intermediate nodes. That is, flow entering a node can be held back for some time before it is sent onward. This leads to the following

slightly different notion of flow conservation; we require

$$\sum_{a \in \delta^+(v)} \int_0^\xi f_{a,i}(\theta) d\theta - \sum_{a \in \delta^-(v)} \int_{\tau_a}^\xi f_{a,i}(\theta - \tau_a) d\theta \leq 0, \quad (1.3)$$

for all $\xi \in [0, T)$, $i \in K$, and $v \in V \setminus S_i^+$. The left sum in (1.3) denotes the total amount of flow of commodity i that is leaving node v until time ξ . Analogously, the right sum in (1.3) denotes the total amount of flow of commodity i that is entering node v until time ξ ; notice that flow which is entering node v at time θ via arc a must have entered this arc at time $\theta - \tau_a$. Hence, the left side of (1.3) defines the net outflow of f until time ξ with respect to commodity i . We require in (1.3) that the net outflow remains below zero in order to rule out a flow deficit at node v . Moreover, flow must not remain in any node other than the sinks at time T . Therefore, we demand that equality holds in (1.3) for every $i \in K$, $v \in V \setminus S_i$, at time $\xi = T$. If storage at intermediate nodes is forbidden, we additionally require that equality holds in (1.3) for all $\xi \in [0, T)$, $i \in K$, and $v \in V \setminus S_i$.

The flow f is called *feasible*, if the capacity u_a is an upper bound on the rate of flow entering arc a at any moment in time, i.e., $f_a(\theta) \leq u_a$, for all $\theta \in \mathbb{R}^+$ and $a \in A$.

The flow over time f satisfies multi-commodity supplies and demands if

$$\sum_{a \in \delta^+(v)} \int_0^T f_{a,i}(\theta) d\theta - \sum_{a \in \delta^-(v)} \int_{\tau_a}^T f_{a,i}(\theta - \tau_a) d\theta = d_{v,i}, \quad (1.4)$$

for every commodity $i \in K$, $v \in S_i$. The *cost* of f is defined as

$$c(f) := \sum_{a \in A} \sum_{i \in K} c_{a,i} \int_0^T f_{a,i}(\theta) d\theta.$$

If each commodity $i \in K$ has a single source s_i and a single sink t_i , we call f a *multi-commodity flow over time*. If only one commodity is given, possibly having several sources and sinks, we call f a *transshipment over time*. If only one commodity with a single source s and a single sink t is given, we use the term *s-t-flow over time*. The *value* of an *s-t-flow* over time f is given by

$$|f| := \sum_{a \in \delta^+(s)} \int_0^T f_a(\theta) d\theta - \sum_{a \in \delta^-(t)} \int_{\tau_a}^T f_a(\theta - \tau_a) d\theta.$$

Notice that $|f|$ is the total amount of flow leaving the source node s until time T and that, because of flow conservation, this value is equal to the total amount of flow arriving in the sink node t until time T .

We conclude this section with a special class of flows over time called temporally repeated flows. They were introduced by Ford and Fulkerson who were able to solve the maximum flow over time problem using temporally repeated flows; see Section 1.3.5. In a sense, temporally repeated flows form the simplest type of flows over time because they resemble static network flows.

Definition 1.1 (Temporally repeated flow). Let x be a feasible static multi-commodity transshipment in G with path decomposition $(x_P)_{P \in \mathcal{P}}$ such that the transit time τ_P of every path $P \in \mathcal{P}$ is bounded from above by T . For every path $P \in \mathcal{P}$, the *temporally repeated* multi-commodity transshipment f sends flow at constant rate x_P into path $P \in \mathcal{P}$ starting at time zero, ending at time $T - \tau_P$.

The temporally repeated multi-commodity transshipment f is a feasible multi-commodity transshipment over time: f naturally obeys flow conservation constraints since it is defined through flows on paths. It even satisfies the strict flow conservation constraints, i.e., no storage of flow at intermediate nodes occurs in f . By definition, the time horizon of f is bounded by T . Finally, it follows from the feasibility of x that f is a feasible flow over time; simply note that the flow rate into arc a at any point in time is bounded by $\sum_{P \in \mathcal{P}: a \in P} x_P = x_a \leq u_a$. The value of a temporally repeated s - t -flow can be expressed in terms of the underlying static flow.

Observation 1.2. The value of a temporally repeated s - t -flow f with underlying static flow $(x_P)_{P \in \mathcal{P}}$ is given by

$$|f| = \sum_{P \in \mathcal{P}} (T - \tau_P) x_P = T|x| - \sum_{a \in A} \tau_a x_a. \quad (1.5)$$

Proof. The total amount of flow sent into path $P \in \mathcal{P}$ is equal to $(T - \tau_P) x_P$. Adding this up over all paths, yields

$$\begin{aligned} |f| &= \sum_{P \in \mathcal{P}} (T - \tau_P) x_P \\ &= T \sum_{P \in \mathcal{P}} x_P - \sum_{P \in \mathcal{P}} \left(\sum_{a \in P} \tau_a \right) x_P \\ &= T|x| - \sum_{a \in A} \tau_a \sum_{P \in \mathcal{P}: a \in P} x_P \\ &= T|x| - \sum_{a \in A} \tau_a x_a, \end{aligned}$$

where the last equation holds because $(x_P)_{P \in \mathcal{P}}$ is a path decomposition of x . \square

It follows from the observation that the value of a temporally repeated s - t -flow is independent of the underlying path decomposition.

1.3.2 Discrete Flows Over Time

Assume that all transit times $(\tau_a)_{a \in A}$ are integral values. A *discrete multi-commodity transshipment over time* f in G assigns to every arc-commodity pair (a, i) a function $f_{a,i} : \mathbb{Z}^+ \rightarrow \mathbb{R}^+$. In contrast to the continuous model, $f_{a,i}(\theta)$ is interpreted as the total amount of flow of commodity i entering arc a at time step θ .

All definitions made in Section 1.3.1 directly carry over to discrete flows over time. Since transit times are integral, one can restrict the discussion to integral time horizons. The flow over time f has *time horizon* T , if no flow is entering an arc $a \in A$ after time $T - 1 - \tau_a$, i.e., $f_a(\theta) := \sum_{i \in K} f_{a,i}(\theta) = 0$, for all time steps $\theta \geq T - \tau_a$, $a \in A$.

Flow conservation constraints are the same as for continuous flows over time, but the integral over time can be replaced by a sum:

$$\sum_{a \in \delta^+(v)} \sum_{\theta=0}^{\xi} f_{a,i}(\theta) - \sum_{a \in \delta^-(v)} \sum_{\theta=\tau_a}^{\xi} f_{a,i}(\theta - \tau_a) \leq 0,$$

for all $\xi \leq T - 1$, $i \in K$, and $v \in V \setminus S_i^+$. Again, we require equality for every $i \in K$, $v \in V \setminus S_i$, at time $\xi = T - 1$.

The flow f is called *feasible*, if the capacity u_a is an upper bound on the amount of flow entering arc a at any time step, i.e., $f_a(\theta) \leq u_a$, for all $\theta \in \mathbb{Z}^+$ and $a \in A$.

The flow f satisfies supplies and demands if

$$\sum_{a \in \delta^+(v)} \sum_{\theta=0}^{T-1} f_{a,i}(\theta) - \sum_{a \in \delta^-(v)} \sum_{\theta=\tau_a}^{T-1} f_{a,i}(\theta - \tau_a) = d_{v,i}, \quad (1.6)$$

for every commodity $i \in K$, $v \in S_i$. The *cost* of a discrete flow over time is defined as

$$c(f) := \sum_{a \in E} \sum_{i \in K} c_{a,i} \sum_{\theta=0}^{T-1} f_{a,i}(\theta).$$

1.3.3 Time-Expanded Graphs

An important contribution of Ford and Fulkerson's work [18, 19] are *time-expanded graphs*. They are primarily used to design and analyze flow over

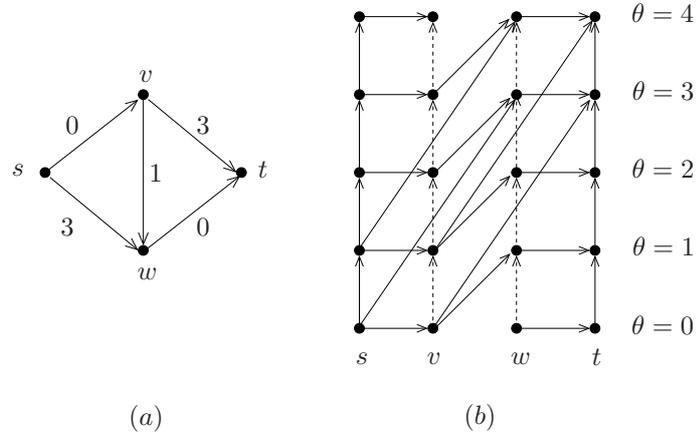


Figure 1.1: Figure (b) displays the T -time-expansion of the instance shown in (a) for time horizon $T = 5$. The numbers at the arcs indicate transit times.

time algorithms. Either the time-expansion is needed explicitly, because the algorithm runs on the time-expanded graph, or it is used implicitly to prove the correctness of the algorithm. For instance, Ford and Fulkerson prove the correctness of their maximum flow over time algorithm by specifying a tight cut in the corresponding time-expanded graph [18, 19].

Given a graph $G = (V, A)$ with integral transit times on the arcs and an integral time horizon T , the T -time-expanded graph of G , denoted $G(T)$, is obtained by creating T copies of V , labeled $V(0)$ through $V(T - 1)$, with the θ^{th} copy of node v denoted $v(\theta)$, $\theta \in \{0, \dots, T - 1\}$. For every arc $a = (v, w) \in A$ and $0 \leq \theta < T - \tau_a$, there is an arc $a(\theta)$ from $v(\theta)$ to $w(\theta + \tau_a)$ with the same capacity as arc a . In the setting with costs, the cost of arc $a(\theta)$ is identical to the cost of arc a . If storage of flow at node $v \in V$ is allowed, we include an infinite capacity *holdover arc* from $v(\theta)$ to $v(\theta + 1)$, for all $0 \leq \theta < T - 1$, which models the possibility to hold flow at node v . An example is given in Figure 1.1. The dashed holdover arcs are included in the time-expanded graph if storage at intermediate nodes is allowed, otherwise they are omitted.

In the T -time-expanded graph, we can consider static network flows. It is not difficult to see that every static multi-commodity transshipment x in $G(T)$ corresponds to a discrete multi-commodity transshipment over time f with time horizon T in G and vice versa; simply identify the static flow value $x_{a(\theta), i}$, assigned to the copy of arc a in the θ^{th} time layer, with the flow $f_{a, i}(\theta)$ entering arc a at time step θ . Notice that this identification also preserves costs.

In particular, a discrete s - t -flow over time f in G corresponds to a static

$s(0)$ - $t(T-1)$ -flow x in $G(T)$ and vice versa. To simplify notation, we call x a static s - t -flow in $G(T)$ implicitly identifying the source s with its copy $s(0) \in V(0)$ and the sink t with its copy $t(T-1) \in V(T-1)$.

As a consequence, every discrete flow over time problem can be formulated as a static flow problem in a time-expanded graph. Since the size of the latter is linear in T (and therefore exponential in $\log T$), a polynomial time static flow algorithm will in general only yield a pseudo-polynomial time algorithm for the corresponding time-dependent problem. Fleischer and Skutella [13, 14] overcome this limitation by using condensed time-expanded graphs. Based on this method, they develop an FPTAS for computing multi-commodity flows over time. We adopt some of their techniques in Chapter 3 when discussing multi-commodity flows in the setting of flow-dependent transit times.

1.3.4 Continuous versus Discrete Model

We now discuss the relationship between continuous and discrete flows over time in G . On the one hand, any discrete flow over time f with integral time horizon T in G corresponds to a continuous flow over time \tilde{f} with time horizon T in G : interpret the flow $f_{a,i}(\theta)$ entering arc a at time step $\theta \leq T-1-\tau_a$ as a constant flow rate on arc a during the whole time interval $[\theta, \theta+1)$. Since $f_a(\theta) \leq u_a$ at every (discrete) time step θ , also $\tilde{f}_a(\theta) \leq u_a$ at every (continuous) point in time θ . Consider an intermediate node v . We verify flow conservation constraints in v . The net outflow of \tilde{f} with respect to commodity $i \in K$ until time $\xi \in [0, T)$ can be expressed as follows:

$$\begin{aligned} & \sum_{a \in \delta^+(v)} \int_0^\xi \tilde{f}_{a,i}(\theta) d\theta - \sum_{a \in \delta^-(v)} \int_{\tau_a}^\xi \tilde{f}_{a,i}(\theta - \tau_a) d\theta \\ &= \sum_{a \in \delta^+(v)} \sum_{\theta=0}^{\lceil \xi \rceil - 1} f_{a,i}(\theta) - \sum_{a \in \delta^-(v)} \sum_{\theta=\tau_a}^{\lceil \xi \rceil - 1} f_{a,i}(\theta) \\ & \quad - (\lceil \xi \rceil - \xi) \left(\sum_{a \in \delta^+(v)} f_{a,i}(\lceil \xi \rceil - 1) - \sum_{a \in \delta^-(v)} f_{a,i}(\lceil \xi \rceil - 1) \right). \end{aligned}$$

This equality implies that the net outflow is given by a piecewise linear function in ξ with breakpoints in \mathbb{Z}^+ . To prove flow conservation constraints, we need to show that this function is nowhere greater than zero. It suffices to show that the function value is not greater than zero at each breakpoint. For $\xi \in \mathbb{Z}^+$, the function value is equal to the net outflow of f with respect to commodity i until time ξ . The latter is not larger than zero because f

satisfies flow conservation constraints in v . We conclude that \tilde{f} satisfies flow conservation constraints in v . A similar calculation shows that \tilde{f} satisfies the same supplies and demands as f .

On the other hand, a continuous flow over time \tilde{f} with integral time horizon T and integral transit times in G yields a discrete flow over time f of same time horizon T in G : set $f_{a,i}(\theta)$ to the total amount of flow sent into arc a during time interval $[\theta, \theta + 1)$, i.e.,

$$f_{a,i}(\theta) := \int_{\theta}^{\theta+1} \tilde{f}_{a,i}(\xi) d\xi, \quad (1.7)$$

for all $a \in A$, $i \in K$, and $0 \leq \theta \leq T - 1 - \tau_a$. The flow f is feasible. Namely, for every integral time step θ , we can bound $f_a(\theta)$ as follows:

$$f_a(\theta) = \int_{\theta}^{\theta+1} \tilde{f}_a(\xi) d\xi \leq \int_{\theta}^{\theta+1} u_a d\xi \leq u_a,$$

where the first inequality holds because \tilde{f} is feasible. Using (1.7), it is easy to verify that flow conservation constraints hold and that f satisfies all supplies and demands.

Notice that in both transformations the cost of the flow is preserved. Hence, for integral data, every continuous flow over time problem can be formulated as a discrete flow over time problem which, in turn, can be formulated as a static flow problem in the T -time-expanded graph G^T ; see Section 1.3.3. Fleischer and Tardos [17] show that a large number of discrete flow over time algorithms can be extended to solve the analogous continuous flow over time problem, even if T is not integral.

Remark 1.3. Since in this thesis we concentrate on the continuous model, by default, a flow over time is a *continuous* flow over time and it is *discrete* if explicitly stated.

1.3.5 Known Results

We give a brief overview of the central results known for continuous flows over time. A more exhaustive discussion of the literature can be found, for instance, in [3, 15, 33, 57]. Some of the results will be relevant later when we develop algorithms for the setting of flow-dependent transit times. Therefore, we investigate them in more detail.

Many of the optimization problems under consideration are \mathcal{NP} -hard problems, i.e., no polynomial time algorithms exist for these problems, unless $\mathcal{P} = \mathcal{NP}$. However, not all \mathcal{NP} -hard problems are equally hard. The

theory of approximation algorithms provides tools to measure the difficulty of an optimization problem. As a large part of the literature on flows over time is devoted to the design and analysis of approximation algorithms, we review the basic definitions.

Definition 1.4 (Approximation algorithm). Let X be a minimization (respectively, maximization) problem. For an instance $I \in X$, let $OPT(I)$ denote the objective value of an optimal solution. Let $\varepsilon > 0$ and set $\rho := 1 + \varepsilon$ (respectively, $\rho := 1 - \varepsilon$). An algorithm A is called a ρ -approximation algorithm for problem X , if for all instances I of X it delivers a feasible solution with objective value $A(I)$ such that

$$|A(I) - OPT(I)| \leq \varepsilon \cdot OPT(I).$$

Moreover, we require that the time complexity of algorithm A is polynomial in the input size of the problem. The value ρ is called *performance guarantee* or *performance ratio* of the approximation algorithm A .

Definition 1.5 (Approximation scheme). Let X be a minimization (respectively, maximization) problem.

- An approximation scheme for problem X is a family of $(1 + \varepsilon)$ -approximation algorithms A_ε (respectively, a family of $(1 - \varepsilon)$ -approximation algorithms A_ε) for problem X over all $0 < \varepsilon < 1$.
- A *polynomial approximation scheme (PTAS)* for problem X is an approximation scheme whose time complexity is polynomial in the input size of the problem.
- A *fully polynomial time approximation scheme (FPTAS)* for problem X is an approximation scheme whose time complexity is polynomial in the input size of the problem and also polynomial in $1/\varepsilon$.

We refer to [32, 64] for a more detailed introduction to the field of approximation algorithms.

We now give an overview of known results for classical flow over time problems. We start with a discussion of single source, single sink flows over time. The maximum flow over time problem was introduced by Ford and Fulkerson under the name maximal dynamic flow problem [18, 19].

Problem 1.6 (Maximum flow over time). Determine an s - t -flow over time f that send as much flow as possible from the source s to the sink t within a given time T .

Originally, Ford and Fulkerson consider this problem in the setting of discrete flows over time; see Section 1.3.2. They prove that the problem can be solved efficiently by transforming it to a static minimum cost flow problem in a related graph. Fleischer and Tardos [17] show that this algorithm directly extends to continuous flows over time. Since we will employ some of the underlying insights of this algorithm, we give a short description of it.

The algorithm of Ford and Fulkerson computes a temporally repeated s - t -flow of maximum value. Recall that the value of a temporally repeated flow f with underlying static flow x is given by $T|x| - \sum_{a \in A} \tau_a x_a$; see Observation 1.2. This suggests the following static flow formulation:

$$\begin{aligned} \max \quad & T|x| - \sum_{a \in A} \tau_a x_a \\ \text{s.t.} \quad & x \text{ static } s\text{-}t\text{-flow in } G. \end{aligned} \tag{1.8}$$

Any solution to this problem defines a temporally repeated flow with time horizon T . This follows immediately from the optimality of x with respect to the objective function:

Observation 1.7. Let x be a static flow solution to (1.8) and let $(x_P)_{P \in \mathcal{P}}$ be an arbitrary path decomposition¹ of x . Then the transit time τ_P of any path $P \in \mathcal{P}$ is bounded by T .

Proof. The objective value in (1.8) can be rewritten as $\sum_{P \in \mathcal{P}} (T - \tau_P) x_P$; see Observation 1.2. Assume that there is a path $P \in \mathcal{P}$ with $\tau_P > T$. By decreasing the amount of flow on path P , the objective value can be increased which is a contradiction to the optimality of x . \square

Hence, a solution x to (1.8) with path decomposition $(x_P)_{P \in \mathcal{P}}$ generates a temporally repeated flow with time horizon T of maximal value. In particular, we can compute the optimal temporally repeated flow solution to Problem 1.6. A priori, it is not clear that an optimal temporally repeated flow defines a globally optimal solution to Problem 1.6. Ford and Fulkerson show that this is indeed true, i.e., a temporally repeated flow of maximal value is a maximum s - t -flow over time.

A solution to (1.8) can be, for example, obtained by adding the arc (t, s) to the original graph with transit time $-T$ and computing a (static) minimum cost circulation in G with transit times interpreted as costs. Using, e.g., the minimum mean cycle-canceling algorithm of Goldberg and Tarjan [25], this

¹We can assume without loss of generality that no cycles are needed in the flow decomposition; otherwise we can decrease flow on cycles without decreasing the objective value.

can be done in strongly polynomial time. A complexity survey for minimum-cost circulation can be found, e.g., in the book of Schrijver [62].

Theorem 1.8 (Ford and Fulkerson [18]). A maximum temporally repeated flow with time horizon T is a maximum flow over time with time horizon T . Moreover, it can be computed by one minimum cost circulation computation.

Closely related to the maximum flow over time problem is the quickest flow problem.

Problem 1.9 (Quickest flow). Determine an s - t -flow over time f that satisfies demand d within minimum time T .

Using the algorithm of Ford and Fulkerson, one can solve the quickest s - t -flow problem in polynomial time: Apply binary search to determine the optimal time horizon T . In each search step, solve the maximum flow over time problem for the current value of T . However, in the continuous setting, the time horizon of a quickest s - t -flow need not be integral. Therefore, it is not clear that a binary search algorithm finds the optimal time horizon T in a polynomial number of iterations. Fleischer and Tardos [17] show that, if demand and transit times are integral, the minimum time horizon T can be expressed as a rational number with denominator bounded by the size of a minimum s - t -cut in the network. Thus, a binary search method can be applied to compute the quickest (continuous) s - t -flow. Theorem 1.8 implies that there exists a temporally repeated solution to the problem.

Corollary 1.10. There exists a temporally repeated flow solution to the quickest flow problem. Moreover, it can be computed in polynomial time.

Burkard, Dlaska, and Klinz [6] present an algorithm which solves the quickest flow problem in strongly polynomial time applying Megiddo's method of parametric search [47].

Theorem 1.11 (Burkard, Dlaska, Klinz [6]). Let \bar{T} denote the time horizon of a quickest s - t -flow. Then \bar{T} and a static flow solution to (1.8) with $T = \bar{T}$ can be computed in strongly polynomial time.

Related to these problems is the following optimization problem, which asks for a flow over time that is of maximal value at every intermediate time step.

Problem 1.12 (Earliest arrival flow). Determine an s - t -flow over time which simultaneously maximizes the amount of flow arriving at the sink before time θ , for all $\theta \in [0, T)$.

Note that it is a priori not clear that a solution to this problem exists. Gale [22] proves the existence of a discrete earliest arrival flow and Philpott [56] extends this result to continuous flows over time. Wilkinson [65] and Minieka [50] suggest to incorporate the successive shortest path algorithm into Ford and Fulkerson's maximum flow over time algorithm; see Theorem 1.8. This method gives rise to a pseudo-polynomial algorithm for finding earliest arrival flows. To our knowledge, the complexity of the earliest arrival flow problem is not known. Zadeh [66] presents a family of instances for which the successive shortest path algorithm needs an exponential number of augmentations. A simple modification of the instances shows that the successive shortest path algorithm applied to the earliest arrival flow problem produces solutions encoded by $\Omega(T)$ paths. Possibly, the flow solutions for these instances require exponential output size.

The earliest arrival flow computed by the successive shortest path algorithm has the property that it simultaneously maximizes the amount of flow departing from the source after time θ , for all $\theta \in [0, T)$. Such a flow is called a *latest departure flow*. Flows over time featuring both properties are called *universally maximal*. A polynomial time approximation scheme for computing universally maximal flows over time was found by Hoppe and Tardos [34]. The algorithm sends a $1 - \varepsilon$ fraction of the maximal flow that can reach the sink t by time θ , $\theta \in [0, T)$, and it sends a $1 - \varepsilon$ fraction of the maximal flow that can leave the source s after time θ , $\theta \in [0, T)$. The algorithm combines capacity scaling with the successive shortest path algorithm.

We proceed with a discussion of flow over time problems that involve costs.

Problem 1.13 (Minimum cost flow over time). Determine an s - t -flow over time f that satisfies demand d within given time T at minimum cost.

As already observed by Klinz and Woeginger [42], this problem is \mathcal{NP} -hard, since finding a minimum cost s - t -flow over time of value $d = 1$ amounts to finding a minimum cost s - t -path with transit time bounded by $T - 1$. The latter problem is known as the constrained shortest path problem and is NP -hard; see, e.g., Garey and Johnson [23]. The minimum cost flow over time problem can be solved in pseudo-polynomial time by translating it to a static minimum cost flow problem in the T -time-expanded graph; see Section 1.3.4.

Klinz and Woeginger [42] consider different variants of Problem 1.13. For example, they show that the problem remains \mathcal{NP} -hard if d is set to the maximum value of a flow over time with time horizon T .

We have seen earlier that the maximum flow over time problem and the quickest flow problem both have a temporally repeated flow solution; see Theorem 1.8. If costs are added, this is no longer true. An interesting related

problem is to find a temporally repeated flow with minimum cost. Klinz and Woeginger [42] show that this is a strongly \mathcal{NP} -hard problem. In particular, it cannot be solved as a static flow problem in the T -time-expanded graph.

Fleischer and Skutella [13, 14] present approximation schemes for the minimum cost flow over time problem. We will discuss these algorithms in more detail in the context of multi-commodity flows over time.

Next, we will consider flow over time problems involving multiple source, sink pairs. Recall that for the quickest transshipment problem we are given a single commodity which may have several source and sink nodes.

Problem 1.14 (Quickest transshipment). Determine a transshipment over time f that satisfies all supplies and demands $d_v, v \in S$, as quickly as possible, i.e., within minimum time T .

We have seen in Section 1.2 that finding a feasible static transshipment easily reduces to a static s - t -flow problem. To prove the equivalence of both problems, a super source s (respectively, super sink t) is added in G which is adjacent to all source nodes (respectively, sink nodes). The capacities on these new arcs ensure that the amount of flow leaving a source node $v \in S^+$ is equal to its supply d_v and that the amount of flow entering a sink node $v \in S^-$ is equal to its demand. This is no longer true for flows over time. Recall that the capacity of an arc limits the rate of flow into an arc and not the total amount of flow entering an arc. Therefore, it is difficult to regulate the flow sent from the super source into the source nodes $v \in S^+$ over the entire time horizon. However, Hoppe and Tardos [34] were able to come up with a strongly polynomial time algorithm for the quickest transshipment problem. Moreover, their algorithm produces a solution that does not make use of storage at intermediate nodes. Their approach relies on *chain-decomposable flows* which generalize the class of temporally repeated flows. These flows are represented by a set of paths, but, unlike temporally repeated flows, these paths may use backward arcs. The downside of this algorithm is that it requires a submodular function minimization oracle as a subroutine and is therefore not of practical use.

Problem 1.15 (Quickest transshipment with costs). Determine a transshipment over time f that satisfies all supplies and demands $d_v, v \in S$, as quickly as possible at cost bounded by a given budget C .

Adding costs turns the transshipment problem into an \mathcal{NP} -hard problem, since its single source, single sink version is already \mathcal{NP} -hard; see [42] and the above paragraph on minimum cost flows over time. Again, translating the problem to a static flow problem in a time-expanded graph yields a pseudo-polynomial time algorithm.

Fleischer and Skutella [13, 14] present approximation schemes for the quickest transshipment problem with costs. The approximation scheme presented in [14] produces a solution that does not make use of storage at intermediate nodes. Moreover, they prove that in the optimal transshipment with costs no storage is needed. We will discuss these algorithms in more detail in the paragraph on multi-commodity flows over time.

Under certain assumptions on the transit times or on the network topology, Hall, Hippler, and Skutella [27] give polynomial time algorithms for the quickest transshipment problem with budget constraint. For instance, they give a strongly polynomial time algorithm that can be applied in tree networks.

We now turn to the multi-commodity version of the problem.

Problem 1.16 (Quickest multi-commodity flow). Determine a multi-commodity flow over time f that satisfies all demands $d_{v,i}, i \in K, v \in S_i$, within minimum time T .

In the decision problem, we ask whether, for a given T , there exists a multi-commodity flow over time f that satisfies all demands within time T . Hall, Hippler, and Skutella [27] prove that the decision problem is \mathcal{NP} -hard, even when restricted to series-parallel networks or to the special case of only two commodities. Moreover, they show that the problem is strongly NP -hard if storage at intermediate nodes is forbidden and flow may only be sent along simple paths. Notice that, without the former assumptions, the problem can be solved in pseudo-polynomial time as a static multi-commodity flow problem in the T -time-expanded network.

Fleischer and Skutella [13, 14] present a $(2 + \varepsilon)$ -approximation algorithm and an FPTAS for the quickest multi-commodity flow problem. Their $(2 + \varepsilon)$ -approximation algorithm generates a temporally repeated flow solution that does not make use of storage at intermediate nodes. In particular, allowing storage of flow at intermediate nodes saves at most a factor of 2 in the optimal time horizon. On the other hand, they present instances where the optimal time horizon without storage at intermediate nodes is $4/3$ times the optimal time horizon with storage.

Problem 1.17 (Quickest multi-commodity transshipment with costs). Determine a multi-commodity transshipment f that satisfies all supplies and demands $d_{i,v}, i \in K, v \in S_i$, as quickly as possible at cost bounded by a given budget C .

Fleischer and Skutella [13, 14] present both a $(2 + \varepsilon)$ -approximation algorithm and an FPTAS for this problem. The $(2 + \varepsilon)$ -approximation algorithm

is based on static length-bounded flow computations in the original graph G . The algorithm outputs a temporally repeated multi-commodity transshipment. The FPTAS relies on static flow computations in a 'condensed' time-expanded graph. This graph has a rougher discretization of time and is therefore of polynomial size. In contrast to the $(2 + \varepsilon)$ -approximation, the solutions produced by the FPTAS might use storage at intermediate nodes. We will discuss both algorithms in more detail in Chapter 3, where we apply some of their results to solve multi-commodity transshipment problems in the setting of flow-dependent transit times.

Hall, Hippler, and Skutella [27] propose exact polynomial time algorithms for the quickest multi-commodity transshipment problem with costs that are applicable under certain restrictions on transit times or network topology.

1.4 FLOW-DEPENDENT TRANSIT TIMES

So far we have considered flows over time with fixed transit times on the arcs. In this setting, the time it takes to traverse an arc does not depend on the current flow situation on the arc. Everybody who was ever caught up in a traffic jam knows that the latter assumption often fails to capture essential characteristics of real-life situations. In many applications, such as road traffic control, production systems, and communication networks (e. g., the Internet), the amount of time needed to traverse an arc of the underlying network increases as the arc becomes more congested.

The presumed dependency of the actual transit time of an arc on the current (and maybe also past) flow situation is the most crucial parameter for modeling traffic flow. Unfortunately, it is a highly nontrivial and open problem to map this parameter into an appropriate and, above all, tractable mathematical network flow model. A fully realistic model of flow-dependent transit times on arcs should take density, speed, and flow rate evolving along an arc into consideration; see, e. g., the book of Sheffi [63] and the report by Gartner, Messer, and Rathi [24] for details on traffic flows. However, there are hardly any algorithmic techniques known which capture these parameters and are capable of providing reasonable solutions even for networks of rather modest size. For problem instances of realistic size, as those occurring in real-life applications, already the solution of mathematical programs relying on simplifying assumptions is in general still beyond the means of state-of-the-art computers.

In practical applications such as traffic flows, precise information on the behavior of the transit time τ_a of an arc a can generally only be found for the case of static flows. In this case, every arc a has a constant flow rate x_a

and the transit time τ_a is given as a function of this flow rate. The transit time function measures how fast the transit time on an arc increases as the flow rate grows. The commonly used transit time functions are monotone increasing and convex. Examples are “Davidson’s function” and a function developed by the U.S. Bureau of Public Roads for traffic flow applications; for details we refer to [63] and to Section 2.5 in which we discuss both transit time functions in more detail.

Research in combinatorial optimization has paid great attention to static traffic flow models. For instance, Jahn, Möhring, Schulz, and Stier Moses [36] consider static traffic flow models and, based on them, they develop algorithms to improve the efficiency and accuracy of route-guidance systems. Their approach is based on a static multi-commodity flow formulation in which every arc has an associated convex cost function given by the transit time function. Additional constraints on the flow paths are imposed to ensure that each driver is assigned a path of acceptable length.

Also Roughgarden and Tardos [60] focus on static traffic flow models; they analyze the relation between the system optimum and the user equilibrium in static traffic flow models. While in a system optimum the total transit time is minimized, in a user equilibrium each individual driver makes a selfish route choice. They show that the total transit time in a selfish routing is never larger than the total transit time incurred by optimally routing twice as much traffic flow.

A significant drawback of static traffic flows is that they only model steady state traffic as can be observed, for instance, during rush hours. Here, the number of cars traversing a street per time unit is essentially constant over a time period and thus traffic exhibits a (static) flow-like behavior. For such stable traffic configurations, static flows are surely an adequate model. Yet, most real-world applications are of dynamic nature. Therefore, in this thesis we are concerned with time-varying traffic models. All models under consideration are derived from the static traffic model described above. More precisely, we assume that, for every arc $a \in A$, a nonnegative, nondecreasing transit time function is given which measures the time it takes to traverse an arc based on the arc flow rate.

Remark 1.18. Transit time functions can be specified in different ways. For instance in road traffic applications, the transit time of an arc is usually given as a concave function, e.g., a polynomial. In this case, we often require only $\mathcal{O}(1)$ information to specify the function. In nonlinear optimization, it is frequently assumed that all input functions are piecewise linear. In that case, we encode the function by specifying the breakpoints and the slopes between these breakpoints.

The encoding of the transit time functions is important when analyzing the running time of an algorithms. An algorithm that solves problems in the piecewise linear model in polynomial time might not solve them in the concise-function model in polynomial time.

To formulate our results as generally as possible, we use the concept of oracle algorithms. An oracle can be regarded as a subroutine whose running time we do not take into account; for more details on oracle algorithms see, e.g., [54]. We call an oracle an *evaluation oracle* for function $h : \mathbb{R}^+ \rightarrow \mathbb{R}^+$, if the following holds: given a rational number x , the oracle returns $h(x)$. Moreover, we require that the encoding length of $h(x)$ is polynomially bounded in the encoding length of x .

We assume that we have an evaluation oracle for each transit time function τ_a , $a \in A$. Later, we need to approximate τ_a by a piecewise constant function. For that purpose, we need the technical assumption that the oracle associated with τ_a can also evaluate the function² $\tau \mapsto \max\{x \mid \tau_a(x) \leq \tau\}$. At last, we need the technical assumption that the oracle can evaluate $\ell_a := \min\{\tau_a(x) + d/(|A|x) \mid x \in (0, u_a]\}$. In fact, it suffices if the oracle can compute a good estimate of ℓ_a , i.e., a constant approximation of ℓ_a . The latter assumption is needed to derive good upper and lower bounds on the objective value of a quickest flow in the setting of inflow-dependent transit times.

Notice that, for the special case that the transit time function τ_a is piecewise linear, we can always implement the oracle as a subroutine with strongly polynomial running time. However, we emphasize that a similar implementation is possible for many other functions, as well.

1.4.1 Time-Dependent Flows

In this section, we will generalize the model of flows over time and will consider so-called time-dependent flows. The generalized model is universal in the sense that an exact specification of transit times is not required. Transit times might be fixed or they might be flow-dependent. Therefore, the model of time-dependent flows serves as a general framework which comprises the basic properties of the considered flow models. Subsequent to this section, we will introduce two models of flow-dependent transit times which fit into this framework.

A *time-dependent multi-commodity transshipment* f is given by Lebesgue-measurable functions $f_{a,i} : \mathbb{R}^+ \rightarrow \mathbb{R}^+$, $a \in A$, $i \in K$. The flow value $f_{a,i}(\theta)$ is the *rate of flow* (per time unit) of commodity i entering arc a at time θ .

²Notice that the function is well-defined because τ is assumed to be left-continuous.

The time-dependent flow f has *time horizon* T , if all arcs are empty after time T .

At any point in time $\theta \in [0, T)$, for every commodity $i \in K$, and for any node $v \in V \setminus S_i^+$, *flow conservation* must hold, i.e., the net outflow at node v until time T with respect to commodity i is never strictly positive. Moreover, at time T , we require that for any node $v \in V \setminus S_i$ the net outflow until time T is zero.

The flow f is called *feasible*, if, for every arc $a \in A$, the capacity u_a is an upper bound on the rate of flow entering arc a at any moment in time, i.e., $f_a(\theta) := \sum_{i \in K} f_{a,i}(\theta) \leq u_a$, for all $\theta \in \mathbb{R}^+$ and $a \in A$.

The time-dependent flow f satisfies the multi-commodity supplies and demands if, for every $i \in K$, $v \in S_i$, the net outflow at node v until time T with respect to commodity i is equal to $d_{v,i}$. The *cost* of a time-dependent flow f is defined as $c(f) := \sum_{a \in A} \sum_{i \in K} c_{a,i} \int_0^T f_{a,i}(\theta) d\theta$.

Notice that every flow over time f with constant transit times $(\tau_a)_{a \in A}$ is a time-dependent flow. We now introduce two models of flow-dependent transit times which fit into this framework.

1.4.2 Inflow-Dependent Transit Times

The focus of this thesis is on a model that we refer to as *flows over time with inflow-dependent transit times*. It is an extension of the flow over time model defined in Section 1.3.1. There, it is assumed that transit times are fixed, so that flow on arc a progresses at constant speed. In the following, we will define the more general model of inflow-dependent transit times. Here, the transit time experienced by an infinitesimal unit of flow on an arc is determined when entering this arc and only depends on the inflow rate at that moment in time. Each arc $a \in A$ has an associated nonnegative, nondecreasing transit time function $\tau_a : [0, u_a] \rightarrow \mathbb{R}^+$ measuring the time it takes for flow to traverse arc a .

A *multi-commodity transshipment over time with inflow-dependent transit times* is a time-dependent multi-commodity transshipment that obeys the following additional rule: flow entering arc a at time θ arrives at $\text{head}(a)$ at time $\theta + \tau_a(f_a(\theta))$. In particular, the transit time of an arc only depends on the current inflow rate. Since in a time-dependent flow, we require that all arcs must be empty from time T on, the following implication must hold for all $a \in A$ and $\theta \in \mathbb{R}^+$: if $f_a(\theta) > 0$, then $\theta + \tau_a(f_a(\theta)) < T$. *Flow conservation*

now reads as follows:

$$\sum_{a \in \delta^+(v)} \int_{0 \leq \theta < \xi} f_{a,i}(\theta) d\theta - \sum_{a \in \delta^-(v)} \int_{\substack{\theta \geq 0: \\ \theta + \tau_a(f_a(\theta)) \leq \xi}} f_{a,i}(\theta) d\theta \leq 0, \quad (1.9)$$

for all $\xi \in [0, T)$, $i \in K$, and $v \in V \setminus S_i^+$, where equality should hold for every $i \in K$, $v \in V \setminus S_i$, at time $\xi = T$.³

The flow over time f satisfies the multi-commodity supplies and demands if

$$\sum_{a \in \delta^+(v)} \int_{0 \leq \theta < \xi} f_a(\theta) d\theta - \sum_{a \in \delta^-(v)} \int_{\substack{\theta \geq 0: \\ \theta + \tau_a(f_{a,i}(\theta)) \leq \xi}} f_{a,i}(\theta) d\theta = d_i, \quad (1.10)$$

for every commodity $i \in K$, $v \in S_i$.

The *value* of an s - t -flow over time f with inflow-dependent transit times is given by

$$|f| := \sum_{a \in \delta^+(s)} \int_0^T f_a(\theta) d\theta - \sum_{a \in \delta^-(s)} \int_0^T f_a(\theta) d\theta.$$

Note that the above requirements coincide with those made in Section 1.3.1 when restricted to constant transit time functions $(\tau_a)_{a \in A}$. Later, we will need the following simple observation. It relies on the fact that intermediate storage of flow is allowed.

Observation 1.19. For every arc $a \in A$, let $\tau_a : [0, u_a] \rightarrow \mathbb{R}^+$ and $\tau'_a : [0, u_a] \rightarrow \mathbb{R}^+$ denote transit time functions on arc a such that $\tau_a(x) \leq \tau'_a(x)$, for all $x \in [0, u_a]$. Then, a flow over time with inflow-dependent transit times $(\tau'_a)_{a \in A}$ and time horizon T naturally defines a flow over time with inflow-dependent transit times $(\tau_a)_{a \in A}$ and time horizon T .

1.4.3 Load-Dependent Transit Times

Köhler and Skutella [44] investigate the model of *flows over time with load-dependent transit times*. The *load* of an arc is the total amount of flow on the arc. The underlying assumption of the model is that the speed on an

³By assumption, τ_a is nondecreasing and thus Lebesgue-measurable. Hence, f_a and τ_a are both Lebesgue-measurable functions implying that the set $\{\theta \geq 0 \mid \theta + \tau_a(f_a(\theta)) < \xi\}$ is Lebesgue-measurable as well. Therefore, the integral in (1.9) is well-defined.

arc is a function of the load. We discuss this model in more detail because we will later point out certain relations between the model of load- and the model of inflow-dependent transit times.

As mentioned at the beginning of this section, the transit time of an arc is typically given for the case of static flows. Then, $\tau_a(x_a)$ is interpreted as the transit time on arc a for the static flow rate x_a . If l_a denotes the load of arc a , it is easy to see, that, for a static flow x , the following relation holds.

$$l_a = x_a \tau_a(x_a). \quad (1.11)$$

As observed in [44], if τ_a is monotonically increasing and convex, then, in a static flow, the flow rate x_a is a strictly increasing and concave function of the load l_a . Hence, for the case of static flows, the transit time can also be interpreted as an increasing function $\hat{\tau}_a$ of the load l_a , i.e.,

$$\tau_a(x_a) = \hat{\tau}_a(l_a). \quad (1.12)$$

If we interpret the static flow value x_a as the flow rate over time on arc a , the speed on arc a is proportional⁴ to the inverse of $\hat{\tau}_a(l_a)$.

For a time-dependent flow f , let $l_a(\theta)$ denote the total amount of flow on arc a at time θ . Again, we refer to $l_a(\theta)$ as the load on arc a . A *multi-commodity transshipment over time with load-dependent transit times* is a time-dependent multi-commodity transshipment, for which the following additional condition holds: at any point in time θ , the speed of the flow on arc a is proportional to the inverse of $\hat{\tau}_a(l_a(\theta))$. In contrast to the model of inflow-dependent transit times, flow units which currently travel on the same arc experience the same speed on that arc.

1.4.4 Temporally Repeated Flows

For flows over time with fixed transit times we have defined the notion of temporally repeated flows which, in a sense, resemble static flows: after a certain inflow phase, during which the temporally repeated flow “fills” all its flow paths, the flow rates on every arc remain fixed until the flow exits the network again. We will extend this notion to the setting of flow-dependent transit times.

Definition 1.20 (Flow-dependent temporally repeated flow). Let x be a feasible static multi-commodity transshipment in G with path decomposition $(x_P)_{P \in \mathcal{P}}$, where \mathcal{P} is a set of paths such that the transit time $\tau_P(x)$

⁴We assume that every arc has a certain geographic length that determines the constant of proportionality.

of every path $P \in \mathcal{P}$ is bounded from above by T . The *temporally repeated multi-commodity transshipment f with flow-dependent transit times* $(\tau_a)_{a \in A}$ and time horizon T is defined as follows.

- (i) For every path $P \in \mathcal{P}$, flow f enters path $P \in \mathcal{P}$ at constant rate x_P starting at time zero, ending at time $T - \tau_P(x)$.
- (ii) The transit time of every arc $a \in A$ is fixed to $\tau_a(x_a)$, i.e., at every point in time $\theta \in [0, T)$, flow units entering arc a at time θ reach $\text{head}(a)$ at time $\theta + \tau_a(x_a)$.

In the definition, the transit time on an arc a only depends on the static flow value x_a . In particular, flow in f travels through arc a at uniform speed. Next we prove that f defines a feasible time-dependent flow. Later we will see that f can even be interpreted as a flow over time with inflow-dependent (respectively, load-dependent) transit times.

The flow f naturally satisfies flow conservation constraints because it is defined through flows on paths. Moreover, it is feasible since the flow rate $f_a(\theta)$ is always upper-bounded by $x_a \leq u_a$. Hence feasibility follows from the feasibility of x . It follows from property (ii) that flow units in f traveling along path $P \in \mathcal{P}$ need exactly $\tau_P(x)$ units of time to reach the sink. Thus, the value of a temporally repeated s - t -flow f with flow-dependent transit times can be directly derived from x .

Observation 1.21. The value of a temporally repeated flow f with flow-dependent transit times $(\tau_a)_{a \in A}$ and underlying path decomposition $(x_P)_{P \in \mathcal{P}}$ is given by

$$|f| = \sum_{P \in \mathcal{P}} (T - \tau_P(x)) x_P = T|x| - \sum_{a \in A} \tau_a(x_a) x_a. \quad (1.13)$$

Proof. We have verified the statement for fixed transit times; see Observation 1.2. A similar calculation proves the statement for flow-dependent transit times. \square

It follows from this observation that the value of a temporally repeated flow with flow-dependent transit times is independent of the underlying path decomposition.

Many of the algorithms presented in this thesis produce temporally repeated flow solutions. The crucial observation is that the set of temporally repeated flows with flow-dependent transit times is contained both in the set of flows over time with inflow-dependent transit times and in the set of flows over time with load-dependent transit times.

Claim 1.22. A temporally repeated multi-commodity transshipment f with flow-dependent transit times $(\tau_a)_{a \in A}$ naturally induces a multi-commodity transshipment over time

- (i) with inflow-dependent transit times $(\tau_a)_{a \in A}$,
- (ii) with load-dependent transit times $(\tau_a)_{a \in A}$.

Proof. First we prove (i). The flow f certainly defines a flow over time with inflow-dependent transit time $(\tau'_a)_{a \in A}$, where $\tau'_a : [0, u_a] \rightarrow \mathbb{R}^+$ is the constant function of fixed value $\tau_a(x_a)$. Since $f_a(\theta) \leq x_a$, for all $\theta \in [0, T)$, we can replace the capacity u_a by $u'_a := x_a$. Then, the statement follows from Observation 1.19. Namely, $\tau_a : [0, u'_a] \rightarrow \mathbb{R}^+$ and $\tau'_a : [0, u'_a] \rightarrow \mathbb{R}^+$ satisfy $\tau_a(x) \leq \tau'_a(x)$, for all $x \in [0, u'_a]$. Thus, Observation 1.19 implies that f naturally defines a flow over time with inflow-dependent transit times $(\tau_a)_{a \in A}$.

Next we prove (ii). We again argue that flow in f is traveling not faster than prescribed by the load-dependent transit times. The flow rate $f_a(\theta)$ never exceeds x_a . Since the transit time of arc a is fixed to $\tau_a(x_a)$, the load of arc a in f is always upper-bounded by $l_a := x_a \tau_a(x_a)$. But for fixed load l_a , the transit time experienced on arc a is equal to $\hat{\tau}_a(l_a) = \tau_a(x_a)$; see (1.12). Since $\hat{\tau}_a$ is an increasing function, it follows that flow in f is traveling not faster than prescribed by the transit time functions. Thus, the statement follows because storage at intermediate nodes is not forbidden. \square

Notice that in both models, inflow-dependent and load-dependent transit times, a temporally repeated flow makes use of the possibility to hold storage at intermediate nodes.

1.4.5 Dynamic Traffic Assignment

The main objective of dynamic traffic assignment is to study, model, and optimize the dynamic behavior of transportation networks. In contrast to static traffic assignment, research in this area focuses on time-varying flows. The rapid developments in driver information systems including route guidance has generated considerable interest in this area. In the following, we give an overview of the literature on dynamic traffic assignment. Notice that we do not consider simulation-based approaches since they are in general not suitable for analytically controllable optimization methods.

A good traffic assignment model should mirror the essential properties of road traffic and it should be tractable from a mathematical programming viewpoint. To our knowledge, all existing formulations are unsatisfactory with respect to either of these requirements. As Friesz, Luque, Tobin, and

Wie [21] remark: “There is yet to emerge from the literature a commonly agreed upon statement of the dynamic traffic assignment problem.”

The articles of Merchant and Nemhauser [48, 49] are widely considered to be seminal work in the field of dynamic traffic assignment. Merchant and Nemhauser [48] present a discrete time model for dynamic traffic assignment with a single destination. In this model, the assignment problem can be formulated as a nonconvex program. We briefly describe the model.

The given time horizon T is divided into equal time intervals. Each arc a in the given graph has a cost function h_a and an exit function g_a . If x is the amount of traffic on arc a at the beginning of time period τ , then a cost of $h_a(x)$ is incurred and $g_a(x)$ units of flow exit arc a during period τ . Notice that no transit time functions are specified; instead, an exit function g_a measures the amount of flow that can exit the arc depending on the load of the arc. All arcs are uncapacitated. For every source node v and for every time period τ , the value $d_{v,\tau}$ specifies the supply at node v during time period τ . The objective is to determine a flow that satisfies the given supplies within time T at minimum total cost.

Unless the functions g_a are all linear, the constraint set is nonconvex. Thus, Merchant and Nemhauser suggest to consider a piecewise linear approximation of the problem instead. With some additional assumptions on the cost function, they can show that a global optimum to the approximate problem can be found using a one-pass simplex algorithm.

Merchant and Nemhauser [49] discuss necessary optimality conditions of the mathematical program described above. In particular, they show that the optimality conditions generalize optimality conditions of a conventional static traffic assignment problem. Moreover, they examine the behavior of the dynamic model under static demand conditions and show that in this case their model is a generalized version of a standard static model.

Carey [8] modifies the model of Merchant and Nemhauser [48] in order to derive a convex program formulation for the traffic assignment problem with a single destination. Associated with each arc a are variables $x_{a,\tau}$, $d_{a,\tau}$, and $b_{a,\tau}$ that measure the flow volume, the inflow, and the outflow, respectively, on arc a during period τ . Moreover, there is an outflow function g_a , which measures the “natural” outflow on arc a given the flow volume on a , and there is a cost function h_a , which measures the travel cost incurred by flow using arc a . The objective is to minimize total cost such that flow conservation holds: at each point in time, the actual outflow must not be larger than the natural outflow. Provided that all functions g_a are concave, the constraint set is convex. If, moreover, all cost functions h_a are convex, the problem formulation induces a convex program and can thus be solved using a variety of well-known algorithms developed for convex programming.

Janson [37] proposes a discrete time mixed integer nonlinear programming formulation for dynamic traffic assignment with multiple sources and sinks. While Merchant and Nemhauser [48] and Carey [8] minimize total cost and hence seek for a system optimal solution, Janson [37] suggest a user equilibrium approach. Here, the objective is to determine an equilibrium in which each individual makes a selfish route choice. The time horizon T is divided into equal time intervals. Each arc has a so-called impedance function f_a that determines the length of the arc during time period τ depending on the current flow volume on arc a . For each path P in the given graph, for each arc $a \in P$, and for every time period τ , there is a zero-one variable that indicates whether flow traveling along path P uses arc a in time period τ . In particular, the problem formulation assumes complete enumeration of all possible paths. The objective is chosen such that, in an optimal solution, flow traveling from the same origin to the same destination experience the same impedance.

A heuristic iterative procedure is presented to compute a solution to this problem; the idea is to route flow along shortest path trees. Here, the length of an arc is computed based on an estimate of the flow volume on that arc in future time intervals. The length of an arc is updated with respect to the current routing.

A similar mixed integer program is considered by Jayakrishnan, Tsai, and Chen [38]. They propose a different iterative algorithm to compute solutions to the mixed integer program. In each iteration, transit times are fixed; a static traffic assignment problem is solved in a time-expanded network. Then, transit times are updated accordingly. Like the algorithm of Janson [37], this algorithm is only a heuristic.

Another iterative procedure is proposed by Kaufman, Smith, and Wunderlich [41]. They suggest a fixed point method to determine a dynamic user equilibrium. They assume that an “assignment mapping” is given as part of the input. This mapping assigns transit times to the arcs given a fixed vehicle routing. The algorithm starts with an initial routing and assigns transit times using the assignment mapping. In each iteration, an optimal routing is determined and transit times are updated using the assignment mapping. It is shown that a fixed point of this iterative procedure defines a user equilibrium. In particular, no individual can alter his route choice so that his trip duration is reduced. However, such a fixed point does not exist in general. Instead of considering the optimal routing in each iteration, Kaufman et al. also suggest to consider a weighted sum of the current optimal routing and the routings computed in previous iterations. A convergence of this procedure to a fixed point is proven under very restrictive assumptions on the assignment mapping and the routing procedure.

Carey and Subrahmanian [9] consider a generalized time-expanded graph for modeling flow-dependent transit times. They assume that each arc $a = (v, w)$ has a piecewise linear transit time function τ_a given by breakpoints $0 = x_0 < x_1 < \dots < x_\ell$, where $\tau_a(x_i) := i$, $i = 0, \dots, \ell$. In the generalized time-expanded graph, they introduce a copy $(v(\theta), w(\theta+i))$ with capacity x_i , for each point in time θ and each transit time $i = 0, \dots, T-1-\theta$. Carey and Subrahmanian consider static network flow formulations in this generalized time-expanded graph with additional bundle constraints linking the flow of the arcs $(v(\theta), w(\theta+i))$, $i = 0, \dots, T-1-\theta$. They derive necessary conditions which guarantee that at most two neighboring arcs $(v(\theta), w(\theta+i))$ and $(v(\theta), w(\theta+i+1))$ carry flow. In chapter 2, we will consider a very similar model and analyze it.

Kaufman, Nonis, and Smith [39] formulate a mixed integer linear program in the generalized time-expanded graph proposed by Carey and Subrahmanian [9]. More precisely, they formulate a static flow problem where zero-one variables indicate whether an arc in the time-expanded graph carries flow. For each arc $a = (v, w)$ and each point in time θ , at most one of the arcs $(v(\theta), w(\theta+i))$, $i = 0, \dots, T-1-\theta$, may carry flow. They suggest to apply a branch-and-bound strategy to solve the resulting mixed integer program.

Friesz, Luque, Tobin, and Wie [21] interpret the dynamic traffic assignment problem as an optimal control problem. Roughly speaking, control theory considers mathematical systems that describe a dynamic process arising, e.g., in a physical, economical, or engineering context. The underlying assumption is that certain state variables $\mathbf{x}(\tau)$, $\tau \geq 0$, and certain control variables $\mathbf{u}(\tau)$, $\tau \geq 0$, describe, respectively control the dynamic process. More precisely, given the initial condition of the state variables $\mathbf{x}(0)$ and given $\mathbf{u}(\tau)$, $\tau \geq 0$, one can determine the state $\mathbf{x}(\tau)$ of the system at any given point in time $\tau \geq 0$. Friesz et al. [21] propose two continuous time formulations, one that corresponds to system optimization and the other to user optimization. Associated with each arc a in the given graph is an inflow function $u_a(\tau)$, a load function $x_a(\tau)$, a load-dependent cost function $c_a(x)$, and a load-dependent exit function $g_a(x)$. The load $x_a(\tau)$ on an arc a at time $\tau \in [0, T]$ is considered as a state variable, the inflow $u_a(\tau)$ of an arc a at time $\tau \in [0, T]$ is considered as a control variable. The function g_a is assumed to be concave. Under certain reasonable assumptions on the initial state of the system and on the given cost and exit functions, Friesz et al. [21] derive necessary and sufficient conditions for the existence of a solution to the system. However, they do not propose any solution techniques. They only note that the treatment of arc exit flow as a nonlinear function g_a might yield some computational difficulties for the case of multiple destinations.

A large part of the literature interprets the dynamic traffic assignment

problem as a variational inequality problem; for more details see, e.g., the book of Ran and Boyce [59]. Variational inequality theory is a methodology for the study of equilibrium problems. It is used in various disciplines including economics, operations research, and engineering. Given a closed convex set and a continuous function $F : K \rightarrow \mathbb{R}^n$, the variational inequality problem is to determine a vector $x^* \in K$, such that the inner product $F(x^*)^T \cdot (x - x^*)$ is greater than or equal to 0, for all $x \in K$. For instance, the static user equilibrium can be characterized as the solution of a variational inequality; see, e.g., the book of Ran and Boyce [59].

Friesz, Bernstein, Smith, Tobin, and Wie [20] define a continuous time dynamic user equilibrium that can be represented as an infinite-dimensional variational inequality. In such an equilibrium, all users with a common travel purpose (i.e., the same origin, destination, and desired arrival time) experience the same travel cost regardless of the selected route. In addition, no unused route or departure time has a lower cost. Their model requires solution of a complex program expressed by a system of simultaneous integral equations. Friesz et al. [20] do not present any solution techniques.

Ran and Boyce [58] and Chen and Hsueh [10] formulate discrete-time dynamic user-optimal route choice problems using the variational inequality approach. Ran and Boyce [58] analyze necessary and sufficient conditions for the existence of a solution to this problem. Chen and Hsueh [10] propose an iterative procedure for solving this problem. In each iteration, transit times are estimated, a time-expanded network is constructed, and a static assignment problem is solved. It is not clear that this procedure converges to an optimal solution.

CHAPTER 2

QUICKEST s - t -FLOWS

2.1 INTRODUCTION

For fixed transit times, the classical problems related to s - t -flows over time have been extensively studied. For most of the problems either efficient algorithms were found or it was shown that the problem is \mathcal{NP} -hard in which case approximation algorithms were developed; see Chapter 1. In this chapter we investigate s - t -flows over time with inflow-dependent transit times.

As a starting point, we introduce a relaxed model of inflow-dependent transit times in Section 2.2. This relaxation relies on an expanded graph with fixed transit times on the arcs. To mimic flow-dependent transit times, every arc of the original graph is replaced by a bunch of parallel arcs with different transit times. This expansion has the following intuitive interpretation in the context of road traffic: every expanded arc represents a multi-lane road; traffic on the same lane travels with the same fixed speed. A driver can choose a lane when entering the road. In contrast to real-life traffic, the driver must remain on this lane and follow the prescribed speed. Flow-dependent transit times are realized as follows. High-speed lanes have a relatively small capacity. If the capacity of such a lane is used up, drivers must enter a lane with lower speed.

Based on this relaxation, we present approximation results for the quickest flow problem in Section 2.3. The results demonstrate the quality of this relaxation and thus justify its consideration for practical purposes. In Section 2.4 we discuss the complexity of the quickest flow problem with inflow-dependent transit times. We prove that this problem is \mathcal{NP} -hard in the strong sense. We conclude this chapter with a presentation of computational results that confirm the practical usefulness of our algorithms.

A substantial part of this chapter is based on joint work with Ekkehard Köhler, Martin Skutella, and Alex Hall. Extended abstracts appeared in [28] and [43].

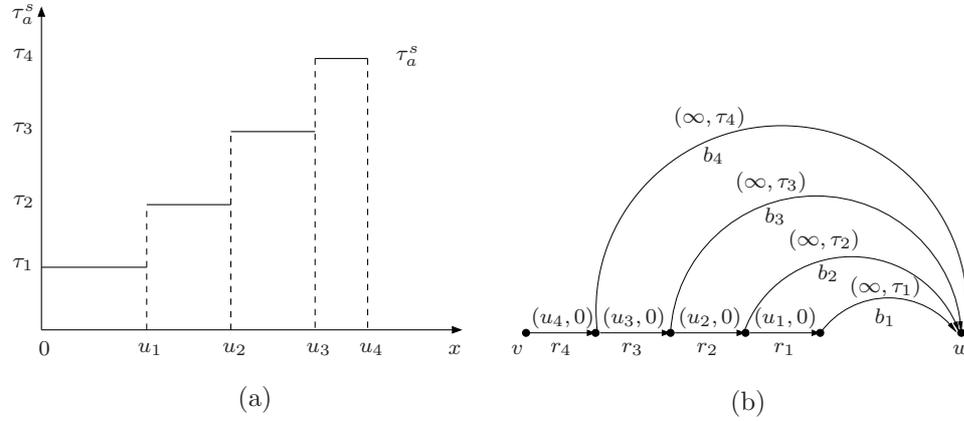


Figure 2.1: Figure (b) depicts the expansion of a single arc $a = (v, w)$ according to its transit time function τ_a^s as given in Figure (a).

2.2 A RELAXATION

Our eventual goal is to design good approximation algorithms for flows over time with inflow-dependent transit times. On the way there, we introduce a simpler model which on the one hand enables us to apply algorithms known for flows over time with fixed transit times and on the other hand allows for a dependency of transit times on the inflow. In other words, the simpler model defines a relaxation of flows over time with inflow-dependent transit times. Throughout this section we make the following assumption:

Assumption. All transit time functions are given as piecewise constant, nondecreasing, and left-continuous functions $(\tau_a^s)_{a \in A}$.

To stress the step function character of this transit time function, we denote it by τ_a^s . Later we will use the fact that general transit time functions can be approximated by step functions within arbitrary precision.

2.2.1 The Bow Graph

The bow graph, denoted $G^B = (V^B, A^B)$, arises from the original graph G by expanding each arc $a \in A$ according to its transit time function. Consider the example in Figure 2.1 (b), where an arc is expanded according to the step function in Figure 2.1 (a). In G^B , every arc $e \in A^B$ has a capacity u_e and a constant transit time $\tau_e \in \mathbb{R}^+$.

For the definition, let us consider a particular arc $a \in A$. Let the transit

time function τ_a^s be given by breakpoints $0 = u_0 < u_1 < \dots < u_\ell = u_a$ and corresponding transit times $\tau_1 < \dots < \tau_\ell$, where $\tau_a^s(x) := \tau_i$, for $x \in (u_{i-1}, u_i]$.

Arc a is replaced by arcs of two types; *bow arcs*, denoted b_1, \dots, b_ℓ , and *regulating arcs*, denoted r_1, \dots, r_ℓ . Every bow arc b_i represents a possible transit time of arc a . More precisely, the transit time τ_{b_i} of arc b_i is set to τ_i , $i = 1, \dots, \ell$. All bow arcs are uncapacitated. The regulating arcs limit the amount of flow entering the bow arcs. Their capacities are chosen according to the breakpoints of transit time function τ_a^s : the capacity u_{r_i} of arc r_i is set to u_i , $i = 1, \dots, \ell$. All regulating arcs have transit time zero. We denote the set of bow arcs and regulating arcs associated to an arc $a \in A$ by A_a^B and refer to A_a^B as the *expansion* of arc a . For every arc $e \in A_a^B$, let $a(e)$ denote the corresponding arc a in A . Note that the size of A_a^B is linear in the number of breakpoints of τ_a^s . We call the nodes in V^B which correspond to nodes in V —in Figure 2.1 (b) the nodes v and w —*original nodes*, the remaining nodes *artificial*.

2.2.2 Relaxation Property of the Bow Graph

We now discuss the relationship between flows over time in the bow graph G^B and flows over time with inflow-dependent transit times in G . In particular, we will see that the bow graph defines a relaxation of the model of inflow-dependent transit times.

Any flow over time f with inflow-dependent transit times $(\tau_a^s)_{a \in A}$ and time horizon T in G can be interpreted as a flow over time (with constant transit times) f^B with the same time horizon T in G^B . If in the original graph G flow is entering arc $a \in A$ at time θ with flow rate $f_a(\theta)$, this flow reaches $\text{head}(a)$ at time $\theta + \tau_a^s(f_a(\theta))$. We can simulate this behavior in the bow graph by sending this flow onto the arc $e \in A_a^B$ representing transit time $\tau_a^s(f_a(\theta))$. To be more precise, let b_1, \dots, b_ℓ be the bow arcs in A_a^B and let $i \in \{1, \dots, \ell\}$ be chosen such that $\tau_a^s(f_a(\theta)) = \tau_{b_i}$. We define f^B on the expansion of arc a by setting

$$f_e^B(\theta) := \begin{cases} f_a(\theta) & \text{if } e = b_i \text{ or } e \in \{r_j \mid i \leq j \leq \ell\}, \\ 0 & \text{otherwise.} \end{cases} \quad (2.1)$$

Notice that f^B obeys capacity constraints and flow conservation constraints at all intermediate nodes.

Let $\mathcal{F}(T)$ denote the set of flows over time in G with inflow-dependent transit times $(\tau_a^s)_{a \in A}$ and time horizon T . Similarly, let $\mathcal{F}^B(T)$ denote the set of flows over time in G^B (with constant transit times) and time horizon T .

Let $\iota : \mathcal{F}(T) \rightarrow \mathcal{F}^B(T)$ denote the embedding that maps a flow $f \in \mathcal{F}(T)$ to the corresponding flow over time $f^B \in G^B$; see Equation (2.1).

Observation 2.1. Let f be a flow over time with inflow-dependent transit times $(\tau_a^s)_{a \in A}$ in G that sends d units of flow from s to t within time T . Then $\iota(f)$ defines a flow over time (with constant transit times) in G^B which sends d units of flow from s to t within time T .

While every flow over time with inflow-dependent transit times in G can be regarded as a flow over time in G^B , the converse is not true: By our definition of inflow-dependent transit times, flow particles entering arc $a \in A$ at the same time simultaneously arrive at the head node of a . In the bow graph, however, flow units entering the expansion of an arc simultaneously, do not necessarily travel through the expansion at the same speed. The flow is allowed to split up and use bow arcs representing different transit times. Only a portion of the flow is traversing the expansion of arc a at the speed prescribed by the transit time function or possibly slower. The rest might travel at a faster speed. Hence, the above defined embedding ι is not surjective, unless all transit time functions are constant, in which case G and G^B essentially coincide. The following definition characterizes flows over time in G^B that lie in the image of ι .

Definition 2.2. Let f^B be a flow over time in G^B . We call f^B *inflow-preserving* if

- (i) flow in f^B is only stored in original nodes,
- (ii) for every original arc $a \in A$ and at every point in time θ , the flow f^B sends flow into at most one bow arc in A_a^B .

The embedding ι maps every flow over time with inflow-dependent transit times and time horizon T in G^B to an inflow-preserving flow over time; this follows immediately from the definition in (2.1). Still, not every inflow-preserving flow over time in G^B lies in the image of ι . An inflow-preserving flow over time can traverse the expansion of arc a via a bow arc with slower transit time than prescribed by the transit time function. We define a mapping π that projects any inflow-preserving flows over time in $\mathcal{F}^B(T)$ onto the set $\mathcal{F}(T)$.

Let f^B be an inflow-preserving flow over time in $\mathcal{F}^B(T)$. We define a flow over time f with inflow-dependent transit times in G as follows. Consider an arc $a \in A$ and let b_1, \dots, b_ℓ and r_1, \dots, r_ℓ be the set of bow arcs and regulating arcs, respectively, of arc a . For any point in time $\theta \in [0, T)$, we set

$$f_a(\theta) := f_{r_\ell}^B(\theta). \quad (2.2)$$

Claim 2.3. The flow f defines a feasible flow over time with inflow-dependent transit times $(\tau_a^s)_{a \in A}$ and time horizon T in G .

Proof. The flow f satisfies the capacity constraints because f^B does: $f_a(\theta) = f_{r_\ell}^B(\theta) \leq u_{r_\ell} = u_a$, where the latter equality follows from the definition of the bow graph; see Section 2.2.1. It remains to show that f satisfies the flow conservation constraints. It suffices to prove that flow in f travels through arc $a \in A$ not slower than flow in f^B and therefore reaches $\text{head}(a)$ on time. Note that if flow in f travels faster through arc a than flow in f^B , we can compensate for this by storing flow at $\text{head}(a)$.

Consider a point in time θ . If $f_a(\theta) = 0$, nothing needs to be shown, hence, let us assume that $f_a(\theta) = f_{r_\ell}^B(\theta) > 0$. By definition, flow entering arc a at time θ needs $\tau_a^s(f_a(\theta))$ time units to reach $\text{head}(a)$. Consider the corresponding flow in f^B , which enters the expansion of arc a at time θ ; because of property (i) and (ii), there is a unique bow arc b^a with $f_{b^a}^B(\theta) = f_{r_\ell}^B(\theta)$. Since f^B obeys the capacity constraints, the value $f_{b^a}^B(\theta)$ can be bounded by u_{b^a} ; in particular $f_{r_\ell}^B(\theta) \leq u_{b^a}$. Together with (2.2) we derive that $f_a(\theta) \leq u_{b^a}$. Since the transit time function τ_a^s is nondecreasing, we can conclude that $\tau_a^s(f_a(\theta)) \leq \tau_a^s(u_{b^a}) = \tau_{b^a}$, where the latter equality follows from the definition of the bow graph; see Section 2.2.1. We conclude that flow in f travels not slower than flow in f^B . \square

Let π denote the projection that maps an inflow-preserving flow over time f^B in $\mathcal{F}^B(T)$ to the corresponding flow f in $\mathcal{F}(T)$; see Claim 2.3. Notice that $\pi \circ \iota : \mathcal{F}(T) \rightarrow \mathcal{F}(T)$ is the identity map.

Observation 2.4. Let f be an inflow-preserving flow over time in G^B that sends d units of flow from s to t within time T . Then $\pi(f)$ defines a flow over time with inflow-dependent transit times $(\tau_a^s)_{a \in A}$ in G which sends d units of flow from s to t within time T .

We have seen above that flows over time in G^B can be regarded as a relaxation of flows over time with inflow-dependent transit times in G ; simply identify $\mathcal{F}(T)$ with the subset $\iota(\mathcal{F}(T))$ of $\mathcal{F}^B(T)$. Using this insight, we will develop an algorithm that in a first step computes a flow over time solution in the bow graph G^B . By means of various refinement operations, we turn the produced solution into an inflow-preserving flow over time in G^B , while ensuring that the objective value does not increase too much. Applying the projection π , such an inflow-preserving flow over time in G^B can be easily transformed into a solution to the original problem.

The advantage of this relaxation is that it relies on a graph having constant transit times on the arcs. Therefore, we can apply algorithms that are

known for this much simpler model of flows over time. In particular, this approach allows us to use standard network flow algorithms on the time-expansion of G^B ; see Section 1.3.4.

In the remainder of this section we analyze the quality of the relaxation for two simple cases. First, we consider the case that the given graph consists of a single arc only. Then the class of flows over time and the class of inflow-preserving flows over time in G^B essentially coincide. After that, we discuss a simple example in which this property does not hold.

Claim 2.5. Let G consist of a single arc $a = (s, t)$ and let f be a flow over time in G^B with time horizon T that does not use storage in artificial nodes. Then there exists an inflow-preserving flow over time g in G^B that satisfies the same demand as f within time T .

Proof. Let A_a^B consist of bow arcs b_1, \dots, b_ℓ and regulating arcs r_1, \dots, r_ℓ , and let $\theta \in [0, T)$. Then define g as follows:

$$g_{b_i}(\theta) := \begin{cases} f_{r_i}(\theta) & \text{if } f_{b_i}(\theta) > 0 \text{ and } f_{b_j}(\theta) = 0 \text{ for all } j > i, \\ 0 & \text{else,} \end{cases}$$

$$g_{r_i}(\theta) := \begin{cases} f_{r_i}(\theta) & \text{if } f_{b_j}(\theta) = 0 \text{ for all } j > i, \\ 0 & \text{else.} \end{cases}$$

At every point in time, flow in g is sent onto the highest bow arc that is currently used by f . This guarantees that the time horizon of g is not larger than T . Since f satisfies capacity constraints, so does g . \square

We have seen in the last chapter that, for the maximum flow over time problem and for the quickest flow problem, there always exists a solution that does not use storage at intermediate nodes; see Theorem 1.8 and Corollary 1.10. Consequently, Claim 2.5 implies that for single-arc instances, the class of inflow-preserving flows over time in G^B contains a quickest flow and a maximum flow over time. Thus, for this special case, the relaxation defined by the bow graph is tight.

The following instance consisting of two arcs shows that this does not hold in general. More precisely, the time horizon of a quickest flow in G^B can be strictly smaller than the time horizon of any inflow-preserving flow over time in G^B satisfying the same demand.

Example 2.6. Let G consist of a path P of length 2. The first arc on P , denoted a_1 , has capacity $u_{a_1} := 2$ and transit time function

$$\tau_{a_1}^s(x) := \begin{cases} 0 & \text{if } 0 \leq x \leq 1, \\ 1 & \text{if } 1 < x \leq 2. \end{cases}$$

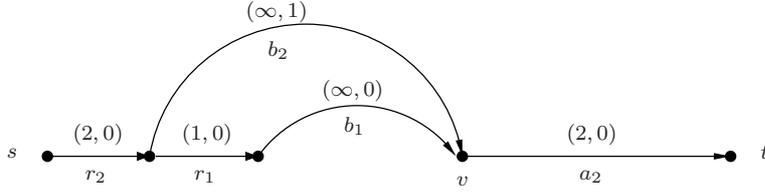


Figure 2.2: The figure depicts an instance, for which the class of inflow-preserving flows over time in G^B neither contains a maximum flow over time in G^B nor a quickest flow in G^B .

The second arc, denoted a_2 , has capacity $u_{a_2} := 2$ and constant transit time $\tau_{a_2} := 0$. The corresponding bow graph is shown in Figure 2.2. Strictly speaking, the arc a_2 should be represented by its expansion which consists of one regulating arc and one bow arc. Since for fixed transit times this expansion is redundant, in this example arc a_2 is not explicitly expanded.

There exists a flow over time in G^B which sends $d := 3$ units of flow from s to t within time $T := 2$: consider the temporally repeated flow which sends flow at rate 1 into the s - t -path containing the lower bow arc b_1 during time interval $[0, 2)$ and flow at rate 1 into the s - t -path containing the upper bow arc b_2 during time interval $[0, 1)$.

We define an inflow-preserving flow over time f in G^B which satisfies demand $5/2$ within time T : it sends flow at rate 2 into bow arc b_2 during $[0, 1/2)$, then it sends flow at rate 1 into bow arc b_1 until time 2. Figure 2.3 illustrates how the flow is routed. Figure (a) depicts the outflow rate of f at node s as defined above. The dark shaded area represents flow that is sent along bow arc b_2 . Analogously, the light shaded area represents flow that is sent along b_1 . Figure (b) displays the inflow rate of f at node v , Figure (c) describes the inflow rate of f at node t . Notice that f uses intermediate storage at node v .

We show that f is maximal, i.e., any inflow-preserving flow over time can satisfy at most demand $5/2$ within time T . In such a flow g , let g_i , $i \in \{1, 2\}$, denote the flow which is sent along bow arc b_i in $[0, 1)$ and let g'_1 denote the flow which is sent along bow arc b_1 in $[1, 2)$. Note that no flow can be sent into bow arc b_2 after time 1 since this flow cannot reach the sink on time. The value of g can then be expressed as $|g| = |g_1| + |g'_1| + |g_2|$.

We distinguish two cases. First, assume $|g_1| \leq 1/2$. Since the capacity of arc a_2 is 2, we can upper-bound $|g'_1| + |g_2|$ by 2. Hence, $|g| = |g_1| + |g'_1| + |g_2| \leq 2 + 1/2$. Next, assume $|g_1| > 1/2$, i.e., $|g_1| = 1/2 + \varepsilon$, where $\varepsilon > 0$. Then, the flow g_1 is entering bow arc b_1 for at least $1/2 + \varepsilon$ time units.

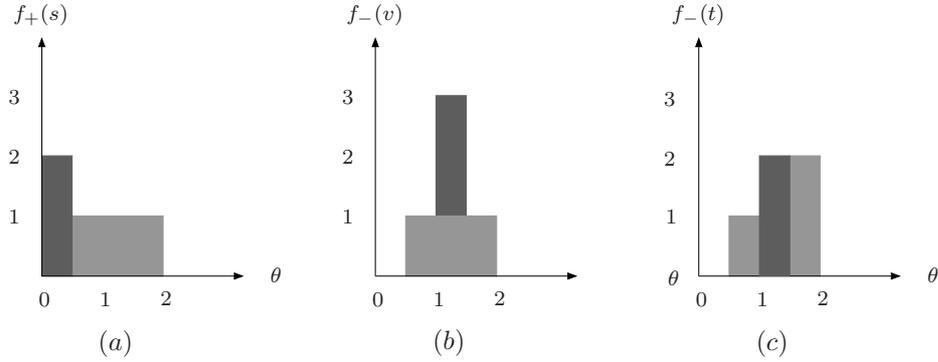


Figure 2.3: Description of an optimal inflow-preserving flow over time in the bow graph which is depicted in Figure 2.2. Figure (a) depicts the outflow rate out at node s , Figure (b) the inflow rate at node v , and Figure (c) the inflow rate at node t . In total, $5/2$ units of flow reach t before time 2.

Thus, $|g_2| \leq 2(1/2 - \varepsilon)$. By capacity constraints, $|g'_1| \leq 1$. In total, $|g| = |g_1| + |g_2| + |g'_1| \leq 1/2 + \varepsilon + 2(1/2 - \varepsilon) + 1 < 5/2$. Thus, in both cases the value of g can be upper-bounded by $5/2$. We conclude that f is maximal.

There exists an inflow-preserving flow over time in G^B that satisfies demand $d = 3$ within time $5/2$: send flow at rate 2 into the s - t -path containing bow arc b_2 during time interval $[0, 3/2)$. One can check that any inflow-preserving flow over time in G^B needs at least time $5/2$ to satisfy demand d .

The following corollary summarizes our observations.

Observation 2.7. The following two statements hold:

- (i) If G consists of a single arc (s, t) , then the time horizon of a quickest inflow-preserving flow of value d in G^B is equal to the time horizon of a quickest flow of value d in G^B .
- (ii) There exist instances in which the time horizon of any inflow-preserving flow over time of value d in G^B is at least $5/4$ times larger than the time horizon of a quickest flow of value d in G^B .

2.3 CONSTANT FACTOR APPROXIMATIONS FOR QUICKEST FLOWS

The purpose of this section is to demonstrate the strength and usefulness of the bow graph presented in the last section for solving time-dependent flow problems with inflow-dependent transit times. This is done by presenting approximation algorithms for the quickest flow problem with inflow-dependent

transit times that are based on optimal solutions to the relaxation given by G^B .

Problem 2.8 (Quickest inflow-dependent flow). Determine an s - t -flow over time f with inflow-dependent transit times that satisfies demand d within minimum time \bar{T} .

2.3.1 Piecewise Constant Transit Times

We now present an approximation algorithm for the case of piecewise constant transit time functions. In the subsequent section we discuss general transit time functions.

Assumption. All transit time functions are given as piecewise constant, nondecreasing, and left-continuous functions $(\tau_a^s)_{a \in A}$.

The first step in the algorithm is to construct the bow graph G^B with respect to the given transit time functions $(\tau_a^s)_{a \in A}$ as described in Section 2.2.1. Next, one determines a quickest flow f^B with time horizon \bar{T}^B in G^B . As Burkard et al. [6] show, the time horizon \bar{T}^B and a solution x^B to the static flow problem (1.8) with time horizon set to \bar{T}^B can be computed in strongly polynomial time by applying Megiddo's method of parametric search; see Theorem 1.11. We denote the resulting temporally repeated flow solution in G^B by f^B . By Equation (1.5), the value of f^B is

$$|f^B| = \bar{T}^B |x^B| - \sum_{e \in A^B} \tau_e x_e^B = d. \quad (2.3)$$

By Observation 2.1, the quickest flow problem on G^B can be seen as a relaxation of the quickest inflow-dependent flow problem on G . Thus the value \bar{T}^B is a lower bound on the optimal time horizon \bar{T} in G .

Later, we will need the following lemma, which follows from the optimality of x^B . We consider the bow graph expansion of a particular arc $a \in A$ to bow arcs b_1, \dots, b_ℓ and regulating arcs r_1, \dots, r_ℓ , where the capacity of r_j is given by u_j .

Lemma 2.9. If x^B sends flow along bow arc b_i , then for all bow arcs b_j with index j smaller than i , we have $x_{b_j}^B = u_j - u_{j-1}$. Less formally speaking, the flow x^B fills the bows from bottom to top.

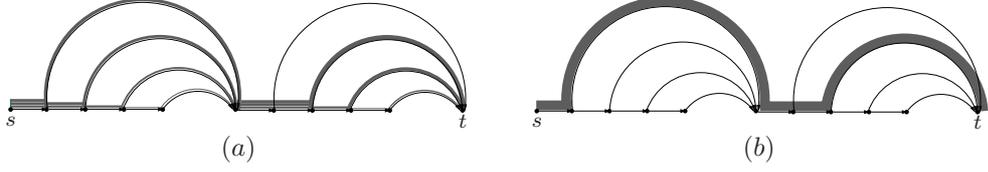


Figure 2.4: Turning a static flow x^B (a) into a static flow \tilde{x}^B (b) that uses only one arc b^a of the expansion of arc $a \in A$. The static flow in (b) generates an inflow-preserving temporally repeated flow in G^B .

Proof. For j smaller than i , the transit time of arc b_j is smaller than the transit time of arc b_i . Therefore, shifting flow from arc b_i to arc b_j improves the value of the objective function (1.8) of x^B . The result thus follows from the optimality of x^B and the choice of capacities on the regulating arcs. \square

Eventually, we are interested in a flow over time in G^B which is inflow-preserving. Unfortunately, as it is discussed in Section 2.2.2, the flow f^B does in general not satisfy this requirement. One reason is that flow units entering the expansion of an arc $a \in A$ simultaneously might experience different transit times in f^B due to different transit times on bow arcs in A_a^B . Therefore, we reroute the static flow x^B to make sure that it does not split among bow arcs representing different transit times of the same arc $a \in A$. This is achieved by pushing flow from “fast” bow arcs up to the “slowest” flow-carrying bow arcs in x^B . An illustration is given in Figure 2.4. More formally speaking, for the expansion A_a^B of an arc a , given by bow arcs b_1, \dots, b_ℓ and regulating arcs r_1, \dots, r_ℓ , the modified static flow \tilde{x}^B is defined by setting

$$\tilde{x}_{b_i}^B := \begin{cases} x_{r_i}^B & \text{if } x_{b_i}^B > 0 \text{ and } x_{b_j}^B = 0 \text{ for all } j > i, \\ 0 & \text{else,} \end{cases}$$

$$\tilde{x}_{r_i}^B := \begin{cases} x_{r_i}^B & \text{if } x_{b_j}^B = 0 \text{ for all } j > i, \\ 0 & \text{else.} \end{cases}$$

Since only the regulating arcs r_1, \dots, r_k are capacitated in G^B , it follows immediately from the feasibility of x^B that \tilde{x}^B is feasible as well. Notice that the value of the flow remains unchanged, i. e.,

$$|\tilde{x}^B| = |x^B|. \quad (2.4)$$

We denote the unique bow arc $b \in A_a^B$ which carries flow in \tilde{x}^B by b^a . Later, we will need the following observation which follows by construction of \tilde{x}^B and Lemma 2.9.

Observation 2.10. Let $a \in A$; either no flow in \tilde{x}^B is routed through A_a^B or the flow in \tilde{x}^B is routed through a unique bow arc $b^a \in A_a^B$ with transit time $\tau_{b^a} = \tau_a^s(\tilde{x}_{b^a}^B)$.

We show that the modified flow \tilde{x}^B yields a flow over time in G^B with value d and time horizon at most $2\bar{T}^B$. For this, we decompose \tilde{x}^B into flows on s - t -paths¹ $P \in \tilde{\mathcal{P}}$ in G^B with flow values \tilde{x}_P^B .

Claim 2.11. The transit time τ_P of every path $P \in \tilde{\mathcal{P}}$ is bounded from above by \bar{T}^B .

Proof. Since all flow-carrying arcs in \tilde{x}^B carry flow in x^B as well, the result follows from Observation 1.7. \square

As a consequence, the path-decomposition $(x_P^B)_{P \in \tilde{\mathcal{P}}}$ of \tilde{x}^B induces a temporally repeated flow $\tilde{f}^B(T)$ in G^B for any time horizon $T \geq \bar{T}^B$. We choose T' such that $|\tilde{f}^B(T')| = T' |\tilde{x}^B| - \sum_{e \in AB} \tau_e \tilde{x}_e^B = d$.

Lemma 2.12. The value of T' is bounded from above by $2\bar{T}^B$.

Proof. Notice that $|\tilde{f}^B(T)|$ is an increasing function of T . Therefore, it suffices to show that $|\tilde{f}^B(2\bar{T}^B)| \geq d$:

$$\begin{aligned}
|\tilde{f}^B(2\bar{T}^B)| &= 2\bar{T}^B |\tilde{x}^B| - \sum_{e \in AB} \tau_e \tilde{x}_e^B && \text{by Observation 1.2,} \\
&= 2\bar{T}^B |\tilde{x}^B| - \sum_{P \in \tilde{\mathcal{P}}} \tau_P \tilde{x}_P^B \\
&= \bar{T}^B |\tilde{x}^B| + \sum_{P \in \tilde{\mathcal{P}}} (\bar{T}^B - \tau_P) \tilde{x}_P^B \\
&\geq \bar{T}^B |\tilde{x}^B| && \text{by Claim 2.11,} \\
&= \bar{T}^B |x^B| && \text{by (2.4),} \\
&\geq |f^B| = d && \text{by (2.3).}
\end{aligned}$$

This concludes the proof. \square

Claim 2.13. The flow over time \tilde{f}^B is inflow-preserving.

¹We can assume without loss of generality that no cycles are needed in the flow decomposition of x^B ; otherwise we can decrease flow on cycles without decreasing the objective value in (1.8). In particular, no cycles are needed in the flow decomposition of \tilde{x}^B .

Proof. We have to check that Properties (i) and (ii) in Definition 2.2 hold. Property (i) holds since a temporally repeated flow (with constant transit times) does not make use of storage, by definition. Property (ii) follows from the construction of \tilde{x}^B . \square

Applying Observation 2.4, turns \tilde{f}^B into an s - t -flow over time with inflow-dependent transit times $(\tau_a^s)_{a \in A}$ in G . The following claim summarizes this result.

Claim 2.14. There exists an s - t -flow over time with inflow-dependent transit times $(\tau_a^s)_{a \in A}$ in G satisfying demand d within time horizon at most $2\overline{T}^B$.

In Section 1.4.4 we have introduced temporally repeated flows with flow-dependent transit times as a special class of flows over time with inflow-dependent transit times. We next show that the flow over time \tilde{f}^B induces a temporally repeated flow with flow-dependent transit times $(\tau_a^s)_{a \in A}$ in G . Take as the underlying static flow x in G the flow defined by

$$x_a := \tilde{x}_{b^a}^B \quad (2.5)$$

and choose the path decomposition induced by the path decomposition of \tilde{x}^B . We have to check that, for every arc $a \in A$, flow in f traverses arc a with fixed transit time $\tau_a^s(x_a)$. Flow in \tilde{x}^B is traveling through the expansion of arc a with fixed transit time τ_{b^a} . By Observation 2.10, this is equal to $\tau_a^s(\tilde{x}_{b^a}^B)$ which, by (2.5), is equal to $\tau_a^s(x_a)$.

Later we will need the following observation.

Observation 2.15. Let $(x_P)_{P \in \mathcal{P}}$ be a path decomposition of x . Then the transit time $\tau_P^s(x)$ of every path $P \in \mathcal{P}$ is bounded by \overline{T} .

Proof. This follows immediately from Claim 2.11, since any path $P \in \mathcal{P}$ induces a flow-carrying path \tilde{P} in \tilde{x}^B with the same transit time. \square

Theorem 2.16. Consider an instance of the quickest inflow-dependent flow problem 2.8 where all transit time functions are step functions. Let \overline{T} denote the optimal time horizon. Then there exists a temporally repeated flow f with flow-dependent transit times $(\tau_a^s)_{a \in A}$ satisfying demand d within time horizon at most $2\overline{T}$. Moreover, such a flow can be computed in strongly polynomial time.

Proof. It follows from the description above that f can be computed in strongly polynomial time. Moreover, $|f| = |\tilde{f}^B| = d$ and the time horizon of f is equal to T , the time horizon of \tilde{f}^B . Finally, by Lemma 2.12, $T \leq 2\overline{T}^B$ and by the relaxation property of the bow graph this is not larger than $2\overline{T}$. \square

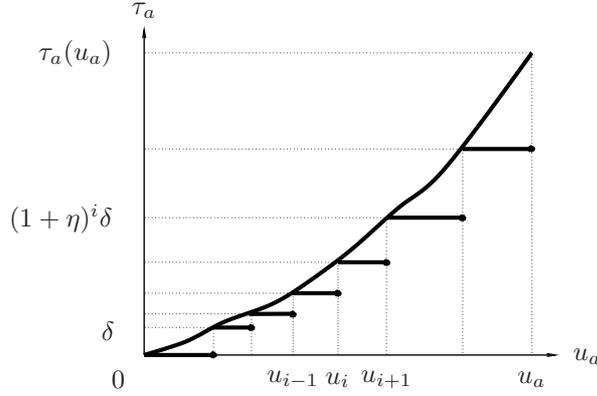


Figure 2.5: Example of a step function τ_a^s approximating transit time function τ_a as explained in Observation 2.18.

Notice that in our analysis we have compared the value T of the computed solution f to the lower bound \bar{T}^B given by an optimal solution f^B to the relaxation of the problem defined by the bow graph G^B . This yields the following result on the quality of this relaxation.

Corollary 2.17. The relaxation of the quickest inflow-dependent flow problem on the bow graph G^B is a 2-relaxation, that is, the value of an optimal solution to the quickest inflow-dependent flow problem is within a factor of 2 of the value of an optimal solution to this relaxation.

2.3.2 General Transit Times

So far we have derived a 2-approximation for the case that all transit time functions are nondecreasing step functions. In this section the approach is generalized to arbitrary nondecreasing transit time functions. The idea is to approximate them by step functions. In order to do this, we need the technical requirement that the transit time functions are left-continuous. See Figure 2.5 for a visualization of a possible step function as claimed in the following observation.

Observation 2.18. Let $\delta, \eta > 0$. For every nonnegative, nondecreasing, and left-continuous function $\tau : [0, u] \rightarrow \mathbb{R}^+$, there exists a step function $\tau^s : [0, u] \rightarrow \mathbb{R}^+$, with

- (i) $\tau^s(x) \leq \tau(x) \leq (1 + \eta) \tau^s(x) + \delta$ for every $x \in [0, u]$,
- (ii) the number of breakpoints of τ^s is bounded by $\lceil \log_{1+\eta}(\tau(u)/\delta) \rceil + 1$.

Proof. Without loss of generality we can assume that $\tau(0) = 0$. In case this does not hold, determine a step function $\tilde{\tau}^s$ satisfying requirements (i) and (ii) for $\tilde{\tau}(x) := \tau(x) - \tau(0)$, $x \in [0, u]$. One can easily check that the step function $\tau^s(x) := \tilde{\tau}^s(x) + \tau(0)$, $x \in [0, u]$, then satisfies the requirements for $\tau(x)$, $x \in [0, u]$. Furthermore, we can assume that $\tau(u) > \delta$. If this is not the case, the step function $\tau^s(x) = 0$, $x \in [0, u]$, satisfies (i) and (ii).

Let $\alpha \in \mathbb{N}$ be chosen minimum with the property that $(1 + \eta)^\alpha \delta > \tau(u)$, i.e., $\alpha = \lceil \log_{1+\eta}(\tau(u)/\delta) \rceil$. For every $i \in \{1, \dots, \alpha\}$, choose $u_i := \max\{x \mid \tau(x) \leq (1 + \eta)^{i-1} \delta\}$. Notice that u_i is well-defined because τ is assumed to be left-continuous. We define the step function τ^s by $\tau^s(x) := 0$, for $x \in (0, u_1]$, and $\tau^s(x) := \tau(u_i)$, for $x \in (u_i, u_{i+1}]$, $i \in \{1, \dots, \alpha\}$. This step function has the desired properties. \square

Remark 2.19. Recall that each transit time function $(\tau_a)_{a \in A}$, is given by an oracle; see Remark 1.18. By assumption, the oracle associated with τ_a can evaluate the function $\tau \mapsto \max\{x \mid \tau_a(x) \leq \tau\}$. At this point the assumption is needed. In the proof of Observation 2.18, we describe a method to compute a step function τ_a^s that approximates τ_a . To guarantee that this method can be performed in strongly polynomial time, we need the technical assumption that the oracle can compute each breakpoint $u_i = \max\{x \mid \tau(x) \leq (1 + \eta)^{i-1} \delta\}$, $i \in \{1, \dots, \alpha\}$.

In the following, we consider transit time functions τ_a^s for the arc a in graph G that fulfill the requirements stated in Observation 2.18. Then, an instance of the quickest inflow-dependent flow problem on G with transit time functions $(\tau_a^s)_{a \in A}$, is a relaxation of the corresponding instance with transit time functions $(\tau_a)_{a \in A}$, since $\tau_a^s(x) \leq \tau_a(x)$, for all $a \in A$ and $x \in [0, u_a]$; see Observation 1.19. In particular, the optimal time horizon \overline{T}^s for transit times τ_a^s is a lower bound on the optimal time horizon \overline{T} for transit times $(\tau_a)_{a \in A}$. Conversely we have:

Lemma 2.20. Let $\delta, \eta > 0$. A temporally repeated flow f^s for transit time functions $(\tau_a^s)_{a \in A}$ in G with time horizon T^s naturally induces a temporally repeated flow f for transit time functions τ_a in G with $|f| \geq |f^s|$ and time horizon $(1 + \eta)T^s + \delta|A|$.

Proof. Let \mathcal{P} denote the set of s - t -paths used by f^s and x_P^s the flow rate on path $P \in \mathcal{P}$. Evidently, the path flows $(x_P^s)_{P \in \mathcal{P}}$ define a temporally repeated flow for transit time functions $(\tau_a)_{a \in A}$ as well. For every arc $a \in A$, let $x_a^s := \sum_{P \in \mathcal{P}: a \in P} x_P^s$. In f^s , the transit time on arc $a \in A$ is fixed to $\tau_a^s(x_a^s)$ and flow traveling along path $P \in \mathcal{P}$ needs $\tau_P^s(x^s) = \sum_{a \in P} \tau_a^s(x_a^s) \leq T^s$ time to reach the sink. If we return to the original transit time functions τ_a , the

transit time on path P increases to $\tau_P(x^s) = \sum_{a \in P} \tau_a(x_a^s) \leq (1 + \eta)\tau_P^s(x^s) + \delta|A|$; see Observation 2.18 (i). Therefore, the value of the corresponding temporally repeated flow f for transit time functions τ_a and time horizon $T' := (1 + \eta)T^s + \delta|A|$ is

$$\begin{aligned} |f| &= T' |x^s| - \sum_{P \in \mathcal{P}} \tau_P(x^s) x_P^s \\ &\geq T' |x^s| - \sum_{P \in \mathcal{P}} ((1 + \eta)\tau_P^s(x^s) + \delta|A|) x_P^s \\ &\geq T' |x^s| - (\eta T^s + \delta|A|) |x^s| - \sum_{P \in \mathcal{P}} \tau_P^s(x^s) x_P^s \\ &= T^s |x^s| - \sum_{P \in \mathcal{P}} \tau_P^s(x^s) x_P^s = |f^s|. \end{aligned}$$

This concludes the proof. \square

If the optimal time horizon \bar{T} or at least a good estimate is known in advance, a provably good solution to the quickest inflow-dependent flow problem in G can be computed in strongly polynomial time.

Lemma 2.21. Given a lower bound L on \bar{T} with $L \leq \bar{T} \leq p(|A|)L$, for some polynomial p , a temporally repeated flow with flow-dependent transit times $(\tau_a)_{a \in A}$ in G satisfying demand d within time horizon at most $(2 + \epsilon)\bar{T}$ can be computed in strongly polynomial time.

Proof. Since any flow over time with time horizon \bar{T} can send flow with rate at most $u'_a := \max\{x \in [0, u_a] \mid \tau_a(x) \leq p(|A|)L\}$ into arc a at any moment in time, we can set $u_a := u'_a$, for all $a \in A$, without changing the optimal time horizon \bar{T} . Set $\delta := \epsilon L / (2|A|)$ and $\eta := \epsilon/4$. Then, by Observation 2.18 (ii), the number of breakpoints of τ_a^s is in $\mathcal{O}(\log(|A|/\epsilon)/\epsilon)$ and thus polynomially bounded in the input size and $1/\epsilon$.

Hence, by Theorem 2.16, for transit time functions τ_a^s a temporally repeated flow f^s in G with flow-dependent transit times $(\tau_a^s)_{a \in A}$ and with time horizon at most $2\bar{T}^s$ of value d can be computed in strongly polynomial time. By Lemma 2.20, f^s induces a solution f to the quickest flow problem for transit time functions τ_a with time horizon at most $(1 + \eta)2\bar{T}^s + \delta|A| \leq (2 + \epsilon)\bar{T}$. \square

Corollary 2.22. For every $\epsilon > 0$, there exists a temporally repeated flow with flow-dependent transit times $(\tau_a)_{a \in A}$ in G satisfying demand d in at most $(2 + \epsilon)\bar{T}$ time.

Proof. The result follows from Lemma 2.21 by setting $L = \bar{T}$. \square

The number of breakpoints of the step functions $(\tau_a^s)_{a \in A}$ chosen in the proof of Lemma 2.21 determines the size of G^B .

Corollary 2.23. The number of arcs (respectively, nodes) in the bow graph constructed with respect to $(\tau_a^s)_{a \in A}$ lies in $\mathcal{O}(|A| \log(|A|/\epsilon)/\epsilon)$.

In view of Lemma 2.21, it remains to show that a lower bound L with $L \leq \bar{T} \leq p(|A|)L$, for some polynomial p , can be found in strongly polynomial time. For an arc $a \in A$, let $\ell_a := \min\{\tau_a(x) + d/(|A|x) \mid x \in (0, u_a]\} > 0$. Notice that ℓ_a is the minimum amount of time that is needed to send $d/|A|$ units of flow through arc a in a temporally repeated fashion. That is, the value ℓ_a essentially measures how fast the arc a can transport flow. If, for instance, transit time function τ_a is constant, the minimum is attained for $x = u_a$.

Remark 2.24. Notice that it is assumed that the oracle associated with transit time function τ_a can determine the value ℓ_a up to a constant factor; see Remark 1.18.

Lemma 2.25. There exists an algorithm with strongly polynomial running time which computes a lower bound L such that $L \leq \bar{T} \leq p(|A|)L$, for some polynomial p .

Proof. For an arbitrary s - t -path P in G , let $\ell_P := \max_{a \in P} \ell_a$. Then, ℓ_P is a lower bound on the amount of time that is needed to send $d/|A|$ units of flow through path P in a temporally repeated flow. Furthermore, we set

$$L := \min\{\ell_P/2 \mid P \text{ is an } s\text{-}t\text{-path in } G\}. \quad (2.6)$$

Notice that L (or a constant approximation of L) can be computed in strongly polynomial time: First determine ℓ_a (or a constant approximation of ℓ_a), for every $a \in A$, using the presumed oracle algorithm associated with τ_a ; see Remark 1.18. The minimum bottleneck capacity of a path—and thus L —can be computed with a Dijkstra-type algorithm; a description of Dijkstra's algorithm can for example be found in [45].

We claim that L is a lower bound on \bar{T} . It follows from Corollary 2.22 that, for every $\epsilon > 0$, there exists a temporally repeated flow f_ϵ in G with flow-dependent transit times $(\tau_a)_{a \in A}$ that sends d units of flow within time $(2 + \epsilon)\bar{T}$. Since the s - t -paths used by f_ϵ result from a path decomposition of the underlying static flow in G , their number can be bounded by $|A|$. In particular, there must exist an s - t -path P_ϵ which carries at least $d/|A|$ units of flow in the temporally repeated solution. This yields $\ell_{P_\epsilon} \leq (2 + \epsilon)\bar{T}$ and therefore $L \leq \ell_{P_\epsilon}/2 \leq (1 + \epsilon/2)\bar{T}$, for every $\epsilon > 0$. Hence, $L \leq \bar{T}$.

Finally, we show that $\bar{T} \leq 2|A|^2 L$. Let P be an s - t -path for which the minimum in (2.6) is attained. By definition of ℓ_P , and since P consists of at most $|A|$ edges, we can send $d/|A|$ units of flow from s to t on P within time $2|A|L$. By repeating this flow $|A|$ time units, we can send d units of flow within time $2|A|^2 L$. \square

Later we will need the following corollary.

Corollary 2.26. Let \bar{T}^t denote the time horizon of a quickest temporally repeated flow with flow-dependent transit times $(\tau_a)_{a \in A}$ and value d in G . Then, the lower bound L in Lemma 2.25 satisfies $L \leq \bar{T}^t \leq p(|A|)L$, for some polynomial p .

Proof. The statement follows immediately from the proof of Lemma 2.25. \square

We can now state the main result of this section, which follows from Lemmas 2.21 and 2.25.

Theorem 2.27. Assume that each transit time function $(\tau_a)_{a \in A}$ is a nonnegative, nondecreasing, left-continuous function given by an oracle as described in Remark 1.18. Then there exists a $(2 + \epsilon)$ -approximation algorithm for the quickest inflow-dependent flow problem with strongly polynomial running time in the input size and $1/\epsilon$.

We point out that the output of the algorithm is a temporally repeated flow. In particular, the inflow rate function on every arc is piecewise constant. A compact description of the final algorithm is given on the following page.

Step 2 of the algorithm can be viewed as a first relaxation of the problem since replacing all transit time functions by lower step functions enlarges the space of flows over which we optimize; see Observation 1.19. Step 4 further relaxes the problem by considering the corresponding bow graph solution.

The running time of this algorithm is clearly dominated by Step 4. This step can be implemented using the algorithm by Burkard et al. [6]. On a graph with n nodes and m arcs, their algorithm has a running time of $\mathcal{O}(\text{MCC}(n, m)^2)$, where MCC is the running time of a minimum cost (static) circulation algorithm which makes only use of certain arc cost operations². They suggest the algorithm by Orlin [53] which fulfills these requirements and is moreover one of the currently asymptotically best algorithms for solving minimum cost flow problems [62]. Applying this result, they achieve a running time of $\mathcal{O}(m^2 \log^2 n (m + n \log n)^2)$. In our case, n and m represent the number of nodes, respectively, arcs in the bow graph G^B . By Corollary 2.23, the size of G^B lies in $\mathcal{O}(|A| \log(|A|/\epsilon)/\epsilon)$.

²The algorithm is allowed to perform: additions and subtractions of two costs, multiplications of an arc cost with a real number, and pairwise comparisons between arc costs.

Algorithm 1: Approximating quickest inflow-dependent flows.

Input: Directed graph $G = (V, A)$ with positive capacities u_a and nonnegative, nondecreasing, left-continuous transit time functions τ_a on the arcs, source-sink node pair $(s, t) \in V \times V$ and flow demand d , parameter $\varepsilon > 0$.

Output: Flow over time f with inflow-dependent transit times in G satisfying demand d within time $(2 + \varepsilon)\bar{T}$.

- 1 Compute lower bound L on \bar{T} (see proof of Lemma 2.25) and set $\eta := \varepsilon/4$ and $\delta := \varepsilon L/(2|A|)$;
 - 2 Replace transit time functions τ_a by step functions τ_a^s (see Observation 2.18);
 - 3 Construct bow graph G^B with respect to $(\tau_a^s)_{a \in A}$;
 - 4 Compute quickest (temporally repeated) flow in G^B ;
 $x^B \leftarrow$ underlying static flow;
 - 5 Turn x^B into static flow x in G according to (2.5);
 - 6 Determine the time horizon T such that $T|x| - \sum_{a \in A} \tau_a(x_a) x_a = d$;
 $f \leftarrow$ temporally repeated flow over time with inflow-dependent transit times and with time horizon T generated by x ;
-

2.3.3 An Improved Result for Concave Transit Times

In this section we show that for the special case of nonnegative, nondecreasing, and concave transit time functions Algorithm 1 achieves a better performance ratio.

Theorem 2.28. Consider an instance of the quickest inflow-dependent flow problem times where the transit time function of every arc is nonnegative, nondecreasing, and concave. Then, a variant of Algorithm 1 achieves performance ratio $3/2 + \varepsilon$.

To this end, we slightly modify the algorithm by defining $\eta := \varepsilon/6$ (instead of $\varepsilon/4$ as in the proof of Lemma 2.21). The parameter $\delta := \varepsilon L/(2|A|)$ remains unchanged.

Let x^B denote the static flow in G^B computed in Step 4 of the algorithm and x the corresponding static flow in G computed in Step 5. We first bound the total transit time $\sum_{a \in A} \tau_a(x_a) x_a$ of x with respect to the original transit time functions $(\tau_a)_{a \in A}$.

Lemma 2.29. Let \bar{T} be the optimal time horizon of a quickest inflow-

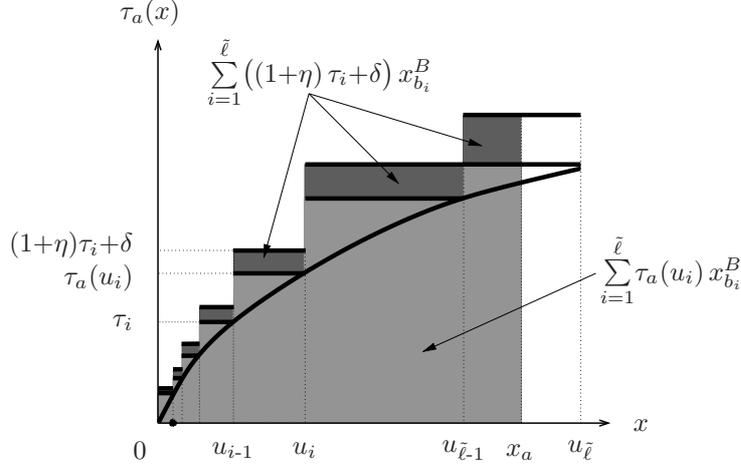


Figure 2.6: Illustration of the proof of Inequality (2.8), first part: The area under the upper step function is larger than the area under the concave transit time function τ_a .

dependent flow in G . Then,

$$\frac{1}{2} \sum_{a \in A} \tau_a(x_a) x_a \leq (1 + \varepsilon/6) \sum_{e \in A^B} \tau_e x_e^B + \varepsilon |x| \bar{T}/2. \quad (2.7)$$

Proof. Consider the expansion A_a^B of a fixed arc $a \in A$ consisting of bow arcs b_1, \dots, b_ℓ with transit times τ_1, \dots, τ_ℓ and of regulating arcs r_1, \dots, r_ℓ with capacities u_1, \dots, u_ℓ . Since all regulating arcs have zero transit time, they do not contribute to the sum on the right hand side of (2.7). Moreover, by definition of x , we have $\sum_{i=1}^{\ell} x_{b_i}^B = x_a \leq |x|$.³ Therefore, it suffices to show that

$$\frac{1}{2} \tau_a(x_a) x_a \leq \sum_{i=1}^{\ell} ((1 + \eta) \tau_i + \delta) x_{b_i}^B, \quad (2.8)$$

since summing up these inequalities over all arcs $a \in A$ yields the statement of the lemma. The value of the step function approximating τ_a when evaluated at point u_i is τ_i . Therefore, the first property stated in Observation 2.18 yields

$$(1 + \eta) \tau_i + \delta \geq \tau_a(u_i); \quad (2.9)$$

see also Figure 2.6. Assume that x^B is sending flow along bow arcs b_1 to $b_{\tilde{\ell}}$.

³Since x has a decomposition into flow on paths (no cycles), $x_a \leq |x|$ holds on every arc $a \in A$.

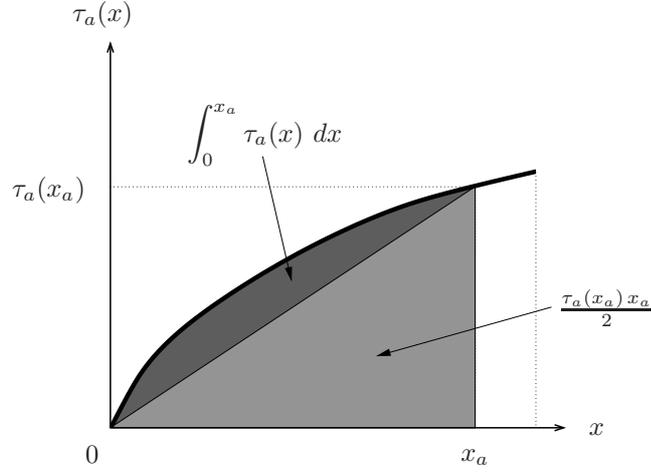


Figure 2.7: Illustration of the proof of Inequality (2.8), second part: the area under the concave transit time function τ_a is at least as large as the area of the triangle below τ_a .

Then, the right hand side of (2.8) can be bounded as follows

$$\begin{aligned}
\sum_{i=1}^{\ell} ((1 + \eta) \tau_i + \delta) x_{b_i}^B &\geq \sum_{i=1}^{\tilde{\ell}} \tau_a(u_i) x_{b_i}^B && \text{by (2.9),} \\
&= \sum_{i=1}^{\tilde{\ell}-1} \tau_a(u_i) (u_i - u_{i-1}) + \tau_a(u_{\tilde{\ell}}) (x_a - u_{\tilde{\ell}-1}) && \text{by Lemma 2.9,} \\
&\geq \int_0^{x_a} \tau_a(x) dx \geq \tau_a(x_a) x_a / 2.
\end{aligned}$$

An illustration of the first two inequalities is given in Figure 2.6. The last inequality holds since the function τ_a is nonnegative, nondecreasing, and concave; see Figure 2.7. \square

Proof of Theorem 2.28. Let $(x_P)_{P \in \mathcal{P}}$ be a path decomposition of x . Then the transit time $\tau_P(x)$ of path $P \in \mathcal{P}$ can be bounded as follows:

$$\begin{aligned}
\tau_P(x) = \sum_{a \in P} \tau_a(x_a) &\leq \sum_{a \in P} ((1 + \eta) \tau_a^s(x_a) + \delta) && \text{by Observation 2.18 (i),} \\
&\leq (1 + \eta) \tau_P^s(x) + |A| \delta \\
&\leq (1 + \eta) \overline{T}^s + |A| \delta && \text{by Observation 2.15,} \\
&\leq (1 + 2/3 \varepsilon) \overline{T}. && (2.10)
\end{aligned}$$

As a consequence we get

$$\begin{aligned}
& (3/2 + \varepsilon) \bar{T} |x| - \sum_{a \in A} \tau_a(x_a) x_a \\
&= (1 + 2/3 \varepsilon) \bar{T} |x| + \frac{1}{2} \sum_{P \in \mathcal{P}} \left((1 + 2/3 \varepsilon) \bar{T} - \tau_P(x) \right) x_P - \frac{1}{2} \sum_{a \in A} \tau_a(x_a) x_a \\
&\geq (1 + 2/3 \varepsilon) \bar{T} |x| - \frac{1}{2} \sum_{a \in A} \tau_a(x_a) x_a && \text{by (2.10),} \\
&\geq (1 + \varepsilon/6) \bar{T} |x| - (1 + \varepsilon/6) \sum_{e \in A^B} \tau_e x_e^B && \text{by Lemma 2.29,} \\
&= (1 + \varepsilon/6) \left(\bar{T} |x^B| - \sum_{e \in A^B} \tau_e x_e^B \right) \\
&= (1 + \varepsilon/6) d \geq d.
\end{aligned}$$

This proves that the time horizon of the computed solution is upper-bounded by $(3/2 + \varepsilon) \bar{T}$. In particular, the performance guarantee of Algorithm 1 is at most $3/2 + \varepsilon$. This concludes the proof of Theorem 2.28. \square

Lower Bounds. We show that the analysis of the above algorithm is tight.

Claim 2.30. The performance ratio of the described approximation algorithm can be arbitrarily close to 2, for general transit time functions, and arbitrarily close to $3/2$, for concave transit time functions.

Proof. First, we define an instance with concave transit times for which the performance of the algorithm is not better than $3/2$. Consider an instance G consisting of a single arc $a = (s, t)$ with linear transit time function τ_a and capacity u_a . As depicted in Figure 2.8 (a), we assume that the function τ_a is going through the origin. For a given demand d , let \bar{T} denote the time horizon of a quickest flow of value d in G with inflow-dependent transit time τ_a . We will prove that Algorithm 1 computes a solution with time horizon T at least $(1 - \varepsilon)3/2 \bar{T}$.

Assume that we have approximated τ_a by a lower step function τ_a^s as proposed by the algorithm. Let $A_a^B := \{b_1, \dots, b_\ell, r_1, \dots, r_\ell\}$ be the expansion of arc a with respect to τ_a^s . We denote the time horizon of a quickest flow over time of value d in G^B by \bar{T}^B . Without loss of generality, we assume that we have specified demand d such that \bar{T}^B is smaller than $\tau_a^s(u_a)$. In particular, $\bar{T}^B < \tau_{b_\ell}$.

Furthermore, let x^B be the static s - t -flow in G^B computed by Ford and Fulkerson's algorithm. Then x^B generates a temporally repeated flow over

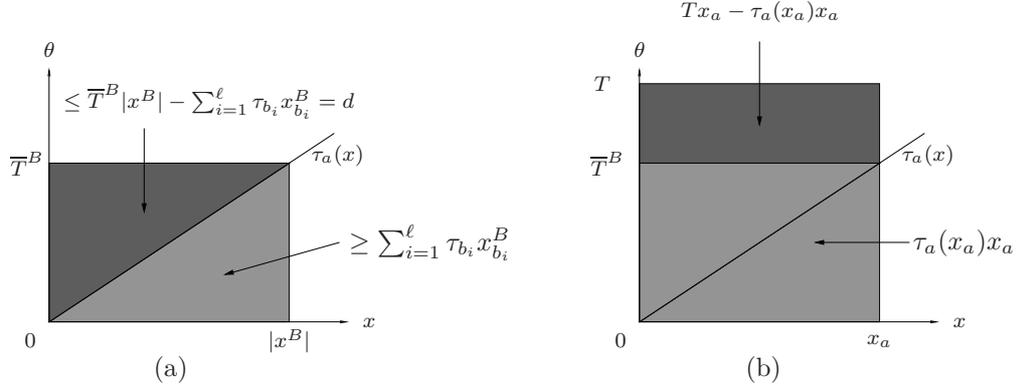


Figure 2.8: A single-arc instance of the quickest inflow-dependent flow problem showing that, for linear transit time functions, the performance guarantee of Algorithm 1 is not better than $3/2$.

time in G^B of value

$$d = \bar{T}^B |x^B| - \sum_{e \in A_a^B} \tau_e x_e^B = \bar{T}^B |x^B| - \sum_{i=1}^{\ell} \tau_{b_i} x_{b_i}^B. \quad (2.11)$$

As already observed in Lemma 2.9, the static flow fills the bow arcs “from bottom to top” assigning flow to all bow arcs b_i with $\tau_{b_i} \leq \bar{T}^B$. We already used in the proof of Claim 2.29 that the total transit time $\sum_{i=1}^{\ell} \tau_{b_i} x_{b_i}^B$ approximates the area under the transit time function τ_a between 0 and $|x^B|$; see Figure 2.8 (a). Hence, by (2.11), the value d is approximately the dark shaded area above τ_a .

Let x_a denote the static s - t -flow on arc a computed in Step 5 of the algorithm; by definition, $x_a = |x^B|$. Its total transit time $\tau_a(x_a)x_a$ is shown in Figure 2.8 (b). Given a time horizon $T \geq \bar{T}^B$, the value of the corresponding temporally repeated flow over time with inflow-dependent transit times in G is $T x_a - \tau(x_a)x_a$. It follows immediately from Figure 2.8 (b) that T needs to be at least $3/2 \bar{T}^B$ in order to satisfy demand d . Together with Claim 2.31, we can conclude that the time horizon T must be at least $(1 - \varepsilon)3/2 \bar{T}$, showing that the performance of the algorithm is not better than $3/2$.

Similarly, it can be shown that, for general transit time functions, the performance guarantee of the algorithm is not better than 2. Consider an instance consisting of a single arc a with transit time function $\tau_a(x) := x^\alpha$, for a given $\alpha \in [1, \infty)$. Following the same analysis as above, it is easy to see that, if α is chosen small enough, the performance ratio of the algorithm can be arbitrarily close to 2; see Figure 2.9. Less formally speaking, the more convex the transit time function is, the worse the performance of the algorithm

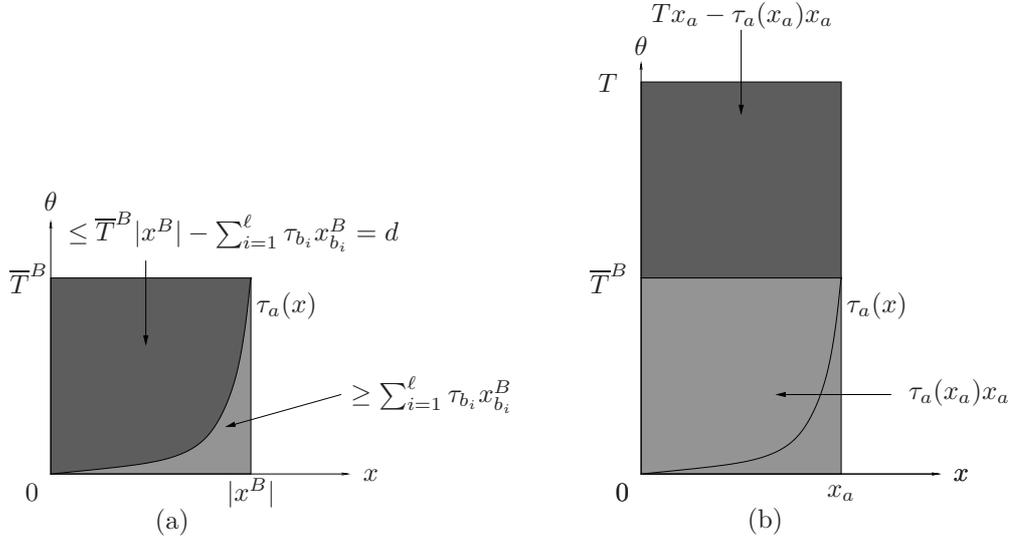


Figure 2.9: A single-arc instance of the quickest inflow-dependent flow problem showing that, for general transit time functions, the performance guarantee of Algorithm 1 is not better than 2.

can get. The computational results presented in Section 2.5 confirm this statement. \square

Claim 2.31. Let G consist of a single arc a with transit time function τ_a ; let G^B be the bow graph constructed in Step 3 of Algorithm 1. The following inequalities hold for \bar{T} and \bar{T}^B :

$$\bar{T}^B \leq \bar{T} \leq \frac{1}{1 - \varepsilon} \bar{T}^B. \quad (2.12)$$

Proof. The first inequality follows immediately from the relaxation property of the bow graph and thus, it only remains to prove the second inequality. Let f^B be a quickest flow over time in G^B . By Corollary 1.10, we can assume that f^B is a temporally repeated flow. Then, f^B naturally induces a temporally repeated flow g^B in the bow graph constructed with respect to the increased transit time function $\tau'_a(x) := (1 + \eta)\tau_a^s(x) + \delta$, $x \in [0, u_a]$; compare Lemma 2.20. Moreover, the time horizon of g^B can be bounded by $(1 + \eta)\bar{T}^B + \delta|A| \leq \bar{T}^B + \varepsilon\bar{T}$. It follows from the first statement of Claim 2.5 that we can assume that g^B is inflow-preserving. By Observation 2.4, the flow g^B corresponds to a flow over time g in G with inflow-dependent transit time τ'_a and same time horizon as g^B . Since $\tau'_a(x) \geq \tau_a(x)$, $x \in [0, u_a]$, the flow g is a flow over time with inflow-dependent transit time τ_a ; see Observation 1.19. Hence, the time horizon \bar{T} is not larger than $\bar{T}^B + \varepsilon\bar{T}$ and the statement follows. \square

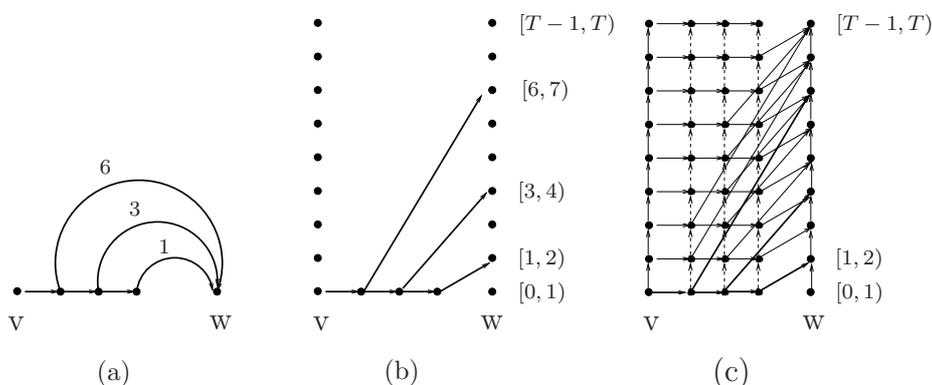


Figure 2.10: Definition of the fan graph; expansion of a single arc $a = (v, w)$.

2.3.4 The Fan Graph

In Sections 1.3.3 we have introduced time-expanded graphs. It was shown that flow over time problems in G translate to static flow problems in a T -time-expanded graph $G(T)$. Also, we have seen in the last section that flows over time in the bow graph serve as a useful model when dealing with flow-dependent transit times. Therefore, it is natural to consider the time-expansion of the bow graph.

Assume that the transit time function τ_a of an arc $a \in A$ is given as a piecewise constant, nondecreasing, and left-continuous function with only integral values. Of course, the latter assumption can easily be relaxed to allow arbitrary rational values if time is scaled by an appropriate factor. Let G^B be the according bow graph. Then we call $G^B(T)$ the *fan graph* of G with (integral) time horizon T . The name “fan graph” is chosen as a figurative description alluding to the bundle of arcs in $G^B(T)$ introduced for an arc in G at each time step; see Figure 2.10 (b). Notice that, for the special case of fixed transit times, the fan graph essentially equals the standard time-expanded graph.

Flows over time in the bow graph constitute a relaxation to inflow-dependent transit times. Hence, the solution space defined by static flows on the fan graph can be also seen as a relaxation of the space of flows over time with inflow-dependent transit times. Moreover, the fan graph can be applied to solve a large variety of time-dependent flow problems, such as multi-commodity flows, flows with costs on the arcs, and so forth. The fan graph will be again relevant in Chapter 3 where we derive approximation algorithms for multi-commodity flows over time with inflow-dependent transit times.

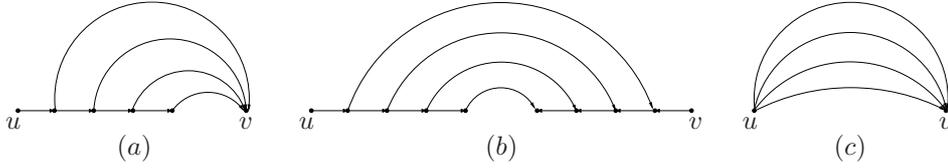


Figure 2.11: Three variants of the bow graph; Figure (a) displays the bow graph as defined in Section 2.2.1. Figure (b) depicts a bow graph that captures in- and outflow-dependent transit times. Figure (c) is the underlying bow graph of an FPTAS for solving multi-commodity flow problems in the setting of inflow-dependent transit times. The FPTAS is presented in Chapter 3.

Remark 2.32. The fan graph is defined as the time-expansion of the bow graph. From the perspective of practical applications, one very natural variation is a bow graph that captures in- and outflow-dependent transit times. This can be achieved by introducing regulating arcs also at the head node of each arc; see Figure 2.11 (b). This symmetric model better reflects the behavior of flows in practical applications, as for example traffic in street networks.

Strictly speaking, flows over time in this enhanced bow graph model do not constitute a relaxation of flows over time with inflow-dependent transit times in G . This is due to the fact that, in a flow over time with inflow-dependent transit times, possibly very large outflow rates can occur on the arcs which is forbidden in model (b). However, the algorithm presented in Section 2.3.2 can also be applied to bow graph model (b); it is not difficult to see that it performs identically on bow graph types (a) and (b). This observation shows that the time horizon of a quickest flow in the bow graph of type (b) and the time horizon of a quickest inflow-dependent flow in G are within a constant factor of each other. Thus, the fan graph arising from bow graph model (b) might be an interesting variation of the problem setting.

In Chapter 3, we will consider a much simpler bow graph and its fan graph; see Figure 2.11 (c). This bow graph has no regulating arcs at all. The expansion of an original arc $a \in A$ only contains parallel bow arcs. As we will see, flows over time in this bow graph again constitute a relaxation of flows over time with inflow-dependent transit times in G . In Chapter 3, we will consider static flows in the corresponding fan graph. Imposing additional bundle constraints that couple the arc flow values of each fan in G^F will be the key to solving multi-commodity flow problems in the setting of inflow-dependent transit times.

Interestingly, Carey and Subrahmanian [9] consider the fan graph induced by the bow graph model shown in Figure 2.11 (c). Yet, in [9] the transit

times in the fan graph have a slightly different interpretation. Carey and Subrahmanian also consider static flow formulations in the fan graph with extra bundle constraints linking the arc flow values of each fan. However, they do not provide a theoretical analysis of this model.

2.3.5 Convex Transit Times

We present an algorithm of Köhler and Skutella [44] which was originally designed to solve the quickest flow problem in the setting of load-dependent transit time; see Section 1.4.3. It relies on a static convex cost flow formulation, and, like the algorithm presented in Section 2.3.2, it computes temporally repeated flow solutions. In the following, we will show that the analysis can easily be generalized to a larger class of time-dependent flows. For the remainder of this section we will assume that all transit time functions $(\tau_a)_{a \in A}$ are convex.

We have introduced time-dependent flows in Section 1.4.1 as a generalized model containing, for example, the classes of flows over time with fixed, inflow-dependent, and load-dependent transit times.

For a time-dependent flow f with time horizon T , we define a corresponding static flow x given by

$$x_a := \frac{1}{T} \int_0^T f_a(\theta) d\theta,$$

for all $a \in A$, and refer to it as the *average rate flow* of f . Since f satisfies the capacity constraints, x does so, too.

Later, we need to be able to bound the *total transit time* of f on an arc a . Generally speaking, the total transit time is the total amount of time spent by all units of flow on that arc. In the context of inflow-dependent transit times, for instance, it is defined by

$$\int_0^T f_a(\theta) \tau_a(f_a(\theta)) d\theta.$$

Similarly, in the setting of load-dependent transit times it can be expressed as

$$\int_0^T \ell_a(\theta) d\theta,$$

where $\ell_a(\theta)$ measures the current load on arc a .

Definition 2.33. We call a time-dependent flow f with time horizon T *lazy* if, on every arc $a \in A$, its total transit time is at least as large as the total transit time $Tx_a\tau_a(x_a)$ of the corresponding static average rate flow.

A key lemma in [44] says that every load-dependent flow over time is lazy. We prove the analog for the case of inflow-dependent transit times.

Lemma 2.34. Every flow over time f with inflow-dependent transit times and time horizon T is lazy, i.e., it satisfies

$$\int_0^T f_a(\theta) \tau_a(f_a(\theta)) d\theta \geq T x_a \tau_a(x_a), \quad (2.13)$$

for all $a \in A$.

Proof. Since τ_a is a nonnegative, convex function, $\xi \rightarrow \xi \tau_a(\xi)$ fulfills these conditions, as well. After dividing both sides of (2.13) by T , the statement follows from Jensen's inequality. \square

Now, we present the approximation algorithm given in [44] and sketch the proof. We consider the following static maximum flow problem with bounded convex cost. Here, the cost of flow x on arc a is $x_a \tau_a(x_a)$.

$$\begin{aligned} \max \quad & |x| \\ \text{s.t.} \quad & \sum_{a \in A} x_a \tau_a(x_a) \leq d \\ & x \text{ (static) } s\text{-}t\text{-flow in } G. \end{aligned} \quad (2.14)$$

Lemma 2.35. If there exists a lazy time-dependent flow f which sends d units of flow from s to t within time T , then there exists a static flow of value at least d/T for the static flow problem (2.14).

Proof. Consider the static average rate flow x of f . By definition, its value is equal to d/T . It remains to show that its total transit time is bounded by d . On the one hand, the total transit time of f is not larger than Td since every unit of flow needs at most T time units to travel through the network. On the other hand, the total transit time of f is not smaller than $T \sum_{a \in A} x_a \tau_a(x_a)$ because f is lazy. It follows that

$$Td \geq T \sum_{a \in A} x_a \tau_a(x_a),$$

which proves the lemma. \square

As Köhler and Skutella [44] observe, a sufficiently good solution to (2.14) can be computed in polynomial time. More precisely, if there is a lazy time-dependent flow which sends d units of flow from s to t within time T , then a static flow x of value at least $(1 - \varepsilon/3) d/T$ and cost at most d can be

computed in polynomial time. The algorithm determines such a flow x and decomposes it into path flows $(x_P)_{P \in \mathcal{P}}$. For fixed $\tilde{T} \geq 0$, we can eliminate all paths in the path decomposition with transit time larger than \tilde{T} . Then the remaining paths yields a temporally repeated flow f with flow-dependent transit times in G of value

$$d(\tilde{T}) = \sum_{P \in \mathcal{P}: \tau_P(x) \leq \tilde{T}} x_P (\tilde{T} - \tau_P(x)). \quad (2.15)$$

From this expression one can determine the minimum time horizon that is needed to satisfy demand d : order the paths in \mathcal{P} by nondecreasing length $\tau_{P_1}(y) \leq \tau_{P_2}(y) \leq \dots \leq \tau_{P_q}(y)$ and observe that the function d is affine linear and increasing within every interval $[\tau_{P_i}(y), \tau_{P_{i+1}}(y)]$, $1 \leq i \leq q$. Köhler and Skutella prove that a time horizon of $(2 + \varepsilon)T$ suffices to guarantee that f satisfies demand d .

Theorem 2.36 (Köhler, Skutella [44]). If there is a lazy time-dependent flow which sends d units of flow from s to t within time T , then there exists a temporally repeated flow with flow-dependent transit times satisfying demand d within time horizon at most $2T$. Moreover, for every $\varepsilon > 0$, one can compute a temporally repeated flow within polynomial time which satisfies demand d within time horizon at most $(2 + \varepsilon)T$.

Originally, this theorem was formulated for flows over time with load-dependent transit times. But, since the class of lazy time-dependent flows contains both the class of flows over time with load-dependent transit times and the class of flows over time with inflow-dependent transit times (see Lemma 2.34), the statement of Theorem 2.36 is even stronger. Moreover, it has the following surprising implication.

Corollary 2.37. Let \bar{T}^ℓ denote the time horizon of a quickest flow with load-dependent transit times satisfying demand d . Similarly, let \bar{T}^i be the time horizon of a quickest flow with inflow-dependent transit times satisfying demand d . Then, $1/2\bar{T}^\ell \leq \bar{T}^i \leq 2\bar{T}^\ell$.

Proof. It follows from Theorem 2.36 that there exists a temporally repeated flow f with flow-dependent transit times of value d with time horizon $T \leq 2 \min\{\bar{T}^\ell, \bar{T}^i\}$. Moreover, it follows from Claim 1.22 that f defines a feasible flow over time both in the setting of inflow-dependent transit times and in the setting of load-dependent transit times and thus $\max\{\bar{T}^\ell, \bar{T}^i\} \leq T$. \square

Algorithm 2: Approximating quickest inflow-dependent flows.

Input: Directed graph $G = (V, A)$ with positive capacities u_a and nonnegative, nondecreasing, convex transit time functions τ_a on the arcs, source-sink node pair $(s, t) \in V \times V$ and flow demand d , parameter $\varepsilon > 0$.

Output: Flow over time f with inflow-dependent transit times in G satisfying demand d within time $(2 + \varepsilon)\bar{T}$.

- 1 Compute a solution x to the static constrained maximum flow problem (2.14) of value at least $(1 - \varepsilon/3)$ the optimum flow value;
 - 2 Decompose x into path flows $(x_P)_{P \in \mathcal{P}}$;
 - 3 Determine the time horizon T such that

$$\sum_{P \in \mathcal{P}: \tau_P(x) \leq T} x_P (T - \tau_P(x)) = d;$$
 - 4 $f \leftarrow$ temporally repeated flow over time with inflow-dependent transit times and time horizon T generated by x ;
-

In summary, the analysis of the algorithm shows that the optimal temporally repeated flow of value d is at most twice as long as any lazy time-dependent flow of value d . Recall that time-dependent flows serve as a very general framework to describe time-varying flows. The main ingredient that is needed in the analysis is the “laziness” of the considered class of flows over time. Hence, this algorithm might be applied to other models of flow-dependent transit times, as well.

An overview of the main steps of the algorithm is given above. For simplicity, we have formulated the algorithm for the setting of inflow-dependent transit times. In [44] the capacity scaling algorithm of Ahuja and Orlin [2] is suggested for Step 1 of the algorithm. It computes an optimal integral solution to the constrained maximum flow problem (2.14) in polynomial time. By appropriate scaling of the data, one can make sure that the value of an optimal integral solution is arbitrarily close to the fractional optimum. In this way, Step 1 can be performed in polynomial time and therefore the algorithm has polynomial running time.

Lower Bounds. The analysis of the algorithm is tight, i.e., the performance ratio of Algorithm 2 is not better than 2. To see this, consider the following example consisting of only one arc a from s to t with fixed transit time $\tau_a \geq 1$ and unit capacity. A quickest s - t -flow for demand $d := 1$ sends flow at rate 1 through the arc and therefore needs $1 + \tau_a$ time units to complete. Algorithm 2 outputs a flow over time with fixed flow rate $1/\tau_a$ and time horizon $2\tau_a$. Thus, for large τ_a , the ratio of the two solution values approaches 2.

Notice that Algorithm 1 computes the optimal solution for the above

instance. This follows from the fact that the bow graph precisely captures the case of fixed transit times and, hence, Algorithm 1 performs optimally on instances with fixed transit times. In contrast to Algorithm 2, it can be applied to a considerably richer family of transit time functions. Also, the stronger performance ratio for the case of concave transit time functions does not seem to translate to Algorithm 2.

2.3.6 Computing Temporally Repeated Flows

So far we have presented two algorithms which produce temporally repeated flow solutions to the quickest *s-t*-flow problem and which have performance guarantee $2 + \varepsilon$. Algorithm 1 is based on a relaxation for inflow-dependent transit times and performs all computations in the bow graph. Algorithm 2 solves a static maximum flow problem with bounded convex cost in G , where the cost is given as the total transit time of the static flow. The question arises if there is a more direct and intuitive method to compute the quickest temporally repeated flow with flow-dependent transit times in G . Indeed, the algorithm of Ford and Fulkerson [18, 19] for the maximum flow over time problem (see Section 1.3.5) can be generalized to flow-dependent transit times.

In this section we assume that all transit time functions $(\tau_a)_{a \in A}$ are convex. For that case, we show that the problem of finding a quickest temporally repeated flow solution reduces to a series of minimum convex cost circulation problems.

We have seen in (1.13) that the value of a temporally repeated flow with time horizon T and underlying static flow x is given by $T|x| - \sum_{a \in A} \tau_a(x_a) x_a$. As suggested by Ford and Fulkerson for the case of constant transit times, we consider the corresponding maximization problem:

$$\begin{aligned} \max \quad & T|x| - \sum_{a \in A} \tau_a(x_a) x_a \\ \text{s.t.} \quad & x \text{ static } s\text{-}t\text{-flow in } G. \end{aligned} \tag{2.16}$$

Let $(x_P)_{P \in \mathcal{P}}$ be a path decomposition of a flow solution to (2.16). Then the transit time $\tau_P(x)$ of every path $P \in \mathcal{P}$ is bounded by T : the objective in (2.16) can be rewritten as $\sum_{P \in \mathcal{P}} (T - \tau_P(x)) x_P$ and thus an optimal solution will not assign flow to a path with transit time larger than T .

We conclude that a solution to (2.16) yields a temporally repeated flow with flow-dependent transit times and time horizon T of maximum value. Such a solution can be found, for example, by adding the arc (t, s) to the original graph with transit time $-T$ and computing a (static) minimum con-

vex cost circulation in G with transit times interpreted as costs. In the book of Ahuja, Magnanti, and Orlin [1] several algorithms are suggested which solve minimum convex cost flow problems. Most of these algorithms, however, compute an integer optimal solution. For example, Hochbaum and Shanthikumar [31] propose a scaling based algorithm that solves separable convex integer programs defined by totally unimodular constraint matrices. Since these conditions apply to problem formulation (2.16), their algorithm can be used to find an optimal integer solution in polynomial time. By appropriately scaling the data, one can make sure that the value of an optimal integral solution is arbitrarily close to the fractional optimum.

These considerations suggest the following alternative approximation algorithm for computing quickest s - t -flows with inflow-dependent (respectively, load-dependent) transit times. Let \overline{T}^t denote the time horizon of a temporally repeated flow with flow-dependent transit times $(\tau_a)_{a \in A}$ that satisfies demand d in minimum time. Theorem 2.36 implies that $\overline{T}^t \leq 2\overline{T}$, where \overline{T} denotes the time horizon of a quickest flow with inflow-dependent (respectively, load-dependent) transit times.

Search for an estimate \tilde{T} of \overline{T}^t satisfying $\overline{T}^t \leq \tilde{T} \leq (1 + \varepsilon/3)\overline{T}^t$ and solve (2.16) with $T = \tilde{T}$. More precisely, compute an approximate solution x to the convex cost flow problem with objective value $\tilde{T}|x| - \sum_{a \in A} \tau_a x_a \geq (1 - \varepsilon/3)d$. By increasing the time horizon to $(1 + \varepsilon/2)\tilde{T}$, we can ensure that the static flow x generates a temporally repeated flow of value at least

$$(1 + \varepsilon/2)\tilde{T}|x| - \sum_{a \in A} \tau_a x_a \geq (1 + \varepsilon/2)(1 - \varepsilon/3)d \geq d.$$

Its time horizon is bounded by $(1 + \varepsilon/2)\tilde{T} \leq (1 + \varepsilon)\overline{T}^t$.

Note that the estimate \tilde{T} can be computed in strongly polynomial time: determine a lower bound L with $L \leq \overline{T}^t \leq p(|A|)L$, for some polynomial p , as suggested in Corollary 2.26 and apply geometric mean binary search. The main steps of this approximation algorithm are summarized on the next page. The discussion above proves the following theorem.

Theorem 2.38. Algorithm 3 has performance ratio $2 + \varepsilon$ for the quickest flow problem both in the setting of load-dependent transit times and in the setting of inflow-dependent transit times. In the latter setting, it has an improved performance guarantee of $3/2 + \varepsilon$ provided that all transit time functions are linear.

Proof. The second part of the statement follows from Theorem 2.28. Since Algorithm 3 assumes that all transit time functions are convex, the improved

Algorithm 3: Approximating quickest inflow-dependent flows.

Input: Directed graph $G = (V, A)$ with positive capacities u_a and nonnegative, nondecreasing, convex transit time functions τ_a on the arcs, source-sink node pair $(s, t) \in V \times V$ and flow demand d , parameter $\varepsilon > 0$.

Output: Flow over time f with inflow-dependent transit times in G satisfying demand d within time $(2 + \varepsilon)\bar{T}$.

- 1 Compute lower bound L with $\bar{T}^t \in [L, p(|A|)L]$ (see Lemma 2.25);
 - 2 Using geometric mean binary search on $[L, p(|A|)L]$, compute a solution x to the static flow problem (2.16) with $\bar{T}^t \leq T \leq (1 + \varepsilon/3)\bar{T}^t$ and $T|x| - \sum_{a \in A} \tau_a x_a \geq (1 - \varepsilon/3)d$;
 - 3 Decompose x into path flows $(x_P)_{P \in \mathcal{P}}$;
 $f \leftarrow$ temporally repeated flow over time with inflow-dependent transit times and time horizon $(1 + \varepsilon/2)T$ generated by x ;
-

performance guarantee only holds for linear (convex and concave) transit time functions. Otherwise one would have to solve a nonconvex optimization problem; see (2.16). \square

Lower Bounds. For Algorithm 1 and Algorithm 2, respectively, we have shown that there exist instances for which the upper bound on the performance guarantee is asymptotically attained. Since Algorithm 3 computes the optimal temporally repeated solution, it might perform better than the theoretical bounds given in Theorem 2.38 suggest. We can derive the following lower bounds on its performance ratio.

Claim 2.39.

- (i) There exist instances of the quickest inflow-dependent flow problem with convex transit time functions where any temporally repeated solution needs at least a factor of $e^{1/e}$ (≈ 1.445) times longer than the optimal solution needs to satisfy the demand. In particular, the performance ratio of Algorithm 3 is not better than $e^{1/e}$.
- (ii) There exist instances of the quickest inflow-dependent flow problem with linear transit time functions where any temporally repeated solution needs at least a factor of $\sqrt{2}$ (≈ 1.414) times longer than the optimal solution needs to satisfy the demand.

Proof. The following instance consisting of a single arc shows that the performance ratio of Algorithm 3 is not better than $e^{1/e}$.

Consider an arc $a = (s, t)$ with transit time function $\tau_a(x) = x^\alpha$ on $[0, \infty)$, where $\alpha \in \mathbb{R}^+$ is a parameter we will specify later. We want to send a given demand d from s to t as quickly as possible. Consider a fixed time horizon T . Setting the inflow-rate to $(T - \theta)^{1/\alpha}$ at time $\theta \in [0, T)$, we manage to send

$$\int_0^T (T - \theta)^{1/\alpha} d\theta = [\alpha/(1 + \alpha) \cdot \theta^{(1+\alpha)/\alpha}]_0^T = \alpha/(1 + \alpha) \cdot T^{(1+\alpha)/\alpha}$$

units of flow from s to t . Notice that this is in fact the maximal demand we can send within given time T ; for every point in time, we are sending flow at the highest possible rate such that every flow unit arrives no later than time T . We can conclude that we can satisfy the demand d within time $\bar{T} := (d(1 + \alpha)/\alpha)^{\alpha/(1+\alpha)}$.

We want to compare this solution to an optimal temporally repeated solution. Again, consider a fixed time horizon T . Having fixed the inflow rate to x , one manages to send $(T - x^\alpha)x$ units of flow from s to t in a temporally repeated fashion. Choosing a constant flow rate $x := (T/(1 + \alpha))^{1/\alpha}$ maximizes this expression and yields a value of

$$(T - T/(1 + \alpha)) \cdot (T/(1 + \alpha))^{1/\alpha} = \alpha T^{\frac{1+\alpha}{\alpha}} / (1 + \alpha)^{\frac{1+\alpha}{\alpha}}.$$

For the latter to be larger than the demand d , the time horizon has to be chosen to be at least $T^t := (d/\alpha)^{\alpha/(1+\alpha)} \cdot (1 + \alpha)$. Comparing this to the optimal time horizon \bar{T} yields a ratio of $T^t/\bar{T} = (1 + \alpha)^{1/(1+\alpha)}$. A simple calculation shows that this is maximized for $\alpha := e - 1$ and then yields a ratio of $e^{1/e}$.

To prove the second part of the statement, simply set α to 1. \square

2.3.7 Limits of the Static Approach

A temporally repeated flow is generated from a static network flow and is therefore, in a sense, a “static” time-dependent flow: after a certain inflow phase, a temporally repeated flow stabilizes to a static flow configuration. We have seen that restricting to this simpler class of time-dependent flows yields provably good solutions to the quickest flow problem in the setting of inflow-dependent transit times. In this section we want to point out to the limits of this approach. In particular, we will consider two classical network flow problems for which this approach fails.

Problem 2.40 (Maximum inflow-dependent flow). Determine a flow over time f with inflow-dependent transit times that sends the maximal amount of flow from the source to the sink within a given time T .

Claim 2.41. For every $\ell \in \mathbb{N}$, there exists an instance of the maximum inflow-dependent flow problem where any temporally repeated solution satisfies at most a fraction of $1/\ell$ of the optimum value.

Proof. We define an instance consisting of a single arc $a = (s, t)$ with a piecewise constant transit time function τ_a^s . The function τ_a^s is given by breakpoints $0 = u_0 < u_1 < \dots < u_{2\ell+1}$, where $u_i := 2^{i-1}$, $i = 1, \dots, 2\ell + 1$, and corresponding transit times $0 = \tau_0 < \tau_1 < \dots < \tau_{2\ell}$, where τ_i is recursively defined by $\tau_{i+1} := \tau_i + 2^{2\ell-i}$, $i = 0, \dots, 2\ell - 1$. Let $\tau_a^s(x) := \tau_i$, for $x \in (u_i, u_{i+1}]$, $i = 0, \dots, 2\ell$. For a given time horizon $T \geq \tau_{2\ell}$, the optimal flow is defined as follows. At every point in time, it sends flow at the highest possible rate such that the flow still reaches t before time T . It starts sending flow at rate $u_{2\ell+1}$ until time $T - \tau_{2\ell}$. Then it reduces the flow rate down to $u_{2\ell}$ to make sure that flow still reaches the sink on time and continues so until time $T - \tau_{2\ell-1}$ and so on. In general the optimal flow is defined by $f_a(\theta) := u_{2\ell+1}$, for $\theta \in [0, T - \tau_{2\ell})$, and $f_a(\theta) := u_{i+1}$, for $\theta \in [T - \tau_{i+1}, T - \tau_i)$, $i = 0, \dots, 2\ell - 1$. If we set $T := \tau_{2\ell}$, the value of f can be expressed as follows:

$$\begin{aligned} |f| &= (T - \tau_{2\ell}) u_{2\ell+1} + \sum_{i=0}^{2\ell-1} (\tau_{i+1} - \tau_i) u_{i+1} \\ &= \sum_{i=0}^{2\ell-1} (\tau_{i+1} - \tau_i) 2^i = 2\ell \cdot 2^{2\ell}. \end{aligned}$$

A temporally repeated flow manages to send at most $2^{2\ell+1}$ units of flow during the same time; namely, assume that it sends flow at constant rate u_{j+1} , $0 \leq j \leq 2\ell + 1$, into arc a . Then its value can be upper-bounded as follows.

$$\begin{aligned} (T - \tau_j) u_{j+1} &= \left((T - \tau_{2\ell}) + \sum_{i=j}^{2\ell-1} (\tau_{i+1} - \tau_i) \right) u_{j+1} \\ &= \left(\sum_{i=j}^{2\ell-1} 2^{2\ell-i} \right) 2^j \\ &= \left(\sum_{i=1}^{2\ell-j} 2^i \right) 2^j \\ &= (2^{2\ell-j+1} - 2) 2^j \leq 2^{2\ell+1}. \end{aligned}$$

We conclude that any temporally repeated flow can satisfy at most a fraction of $1/\ell$ of the optimum value within time T , and the claim follows. \square

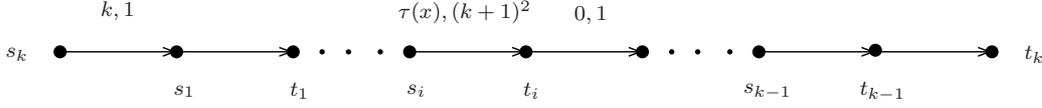


Figure 2.12: An instance with k commodities showing that a temporally repeated flow solution needs k times longer than the optimal solution does.

The claim implies that computing an optimal temporally repeated flow will not result in a constant factor approximation for the maximum inflow-dependent flow problem. Next we consider the multi-commodity version of the quickest flow problem.

Problem 2.42 (Quickest inflow-dependent multi-commodity flow). Determine a multi-commodity flow over time f with inflow-dependent transit times that satisfies all demands d_i , $i = 1, \dots, k$, within minimum time T .

Claim 2.43. For every $k \in \mathbb{N}$, there exists an instance of the quickest inflow-dependent multi-commodity flow problem with k commodities where any temporally repeated solution needs at least a factor of k times longer to satisfy the demands.

Proof. Consider the following instance which consists of a single path $P = (v_1, \dots, v_{2k+2})$ of length $2k+1$. The first $k-1$ commodity pairs are given by (v_{2i}, v_{2i+1}) , $i = 1, \dots, k-1$, and a k^{th} commodity is given by (v_1, v_{2k}) . There are three types of arcs. All 'even' arcs (v_{2i}, v_{2i+1}) , $i = 1, \dots, k-1$, have capacity $(k+1)^2$ and the following transit time function.

$$\tau(x) = \begin{cases} 0 & \text{if } x \in (0, 1], \\ k & \text{if } x \in (1, (k+1)^2]. \end{cases}$$

The first arc has constant transit time k and capacity 1. All other arcs have zero transit time and capacity 1. An illustration is given in Figure 2.12.

The objective is to send one unit of flow from s_k to t_k and $(k+1)^2$ units of flow from s_i to t_i , $i = 1, \dots, k-1$, as quickly as possible. The optimal flow finishes by time $T = k+1$; for one time unit, it sends flow at rate 1 from s_k to t_k and flow at rate $(k+1)^2$ from s_i to t_i , $i = 1, \dots, k-1$. The entire flow reaches its destination by time $k+1$. In a temporally repeated flow solution, the transit time of every arc must be fixed. Assume that the transit time of one of the arcs (s_i, t_i) , $i = 1, \dots, k-1$, is fixed to 0. In particular, flow may only enter that arc at rate not larger than 1. Then at

least $(k + 1)^2$ units of time are needed only to satisfy the demand of the corresponding commodity (s_i, t_i) . On the other hand, if the transit time of every arc (s_i, t_i) , $i = 1, \dots, k$, is fixed to $k + 1$, then, in order to satisfy the demand of commodity (s_k, t_k) , at least $k + 1 + (k - 1)(k + 1)$ time is needed. We can conclude that a temporally repeated flow needs at least $T = k(k + 1)$ time to satisfy all demands and is therefore k times slower than the optimal flow. \square

2.4 COMPLEXITY

For constant transit times, the quickest s - t -flow problem can be solved in strongly polynomial time as shown by Burkard et al. [6]. Moreover, their algorithms output quickest flow solutions that do not make use of intermediate storage in nodes which implies that prohibiting storage does not change the complexity of the problem. In the setting of inflow-dependent transit times, both variants of the problem turn out to be \mathcal{NP} -hard.

Theorem 2.44. The quickest s - t -flow problem with inflow-dependent transit times, with or without storage of flow at intermediate nodes, is NP-hard in the strong sense.

The proof uses a reduction from the well-known NP-complete problem 3-PARTITION.

Problem 2.45 (3-PARTITION). Given a set of $3n$ items with associated sizes $b_1, \dots, b_{3n} \in \mathbb{N}$, a bound $B \in \mathbb{N}$ such that each b_i satisfies $B/4 < b_i < B/2$ and such that $\sum_{i=1}^{3n} b_i = nB$. Decide whether the set $\{1, \dots, 3n\}$ can be partitioned into n disjoint sets I_1, \dots, I_n such that, for $j \in \{1, \dots, n\}$, $\sum_{i \in I_j} b_i = B$.

Given an instance of 3-PARTITION, we construct a network with inflow-dependent transit times as shown in Figure 2.13. Each item b_i is represented by a node v_i , each index set I_j is represented by a node w_j . The capacities are defined as follows.

$$u((s, v_i)) := nb_i + 1, \quad u((v_i, w_j)) := b_i + 1, \quad u((w_j, t)) := (n + 1)B + 3,$$

We define inflow-dependent transit times on (v_i, w_j) as

$$\tau_{(v_i, w_j)}(x) := \begin{cases} 0 & \text{if } x \leq b_i, \\ 1 & \text{else.} \end{cases}$$

All other arcs in the network have transit time zero. The task is to send $D := 2n^2B + 3n$ units of flow from s to t .

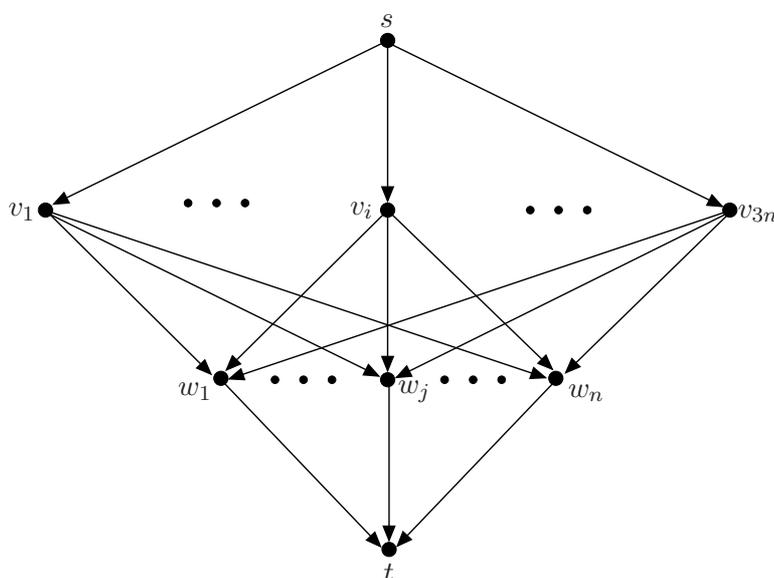


Figure 2.13: Reduction of the problem 3-PARTITION to an s - t -flow over time problem with inflow-dependent transit times.

Lemma 2.46. If the underlying instance of 3-PARTITION is a 'yes'-instance, then there exists an s - t -flow over time with inflow-dependent transit times which sends $2n^2B + 3n$ units of flow from s to t in time $T := 2$ without using storage of flow at intermediate nodes.

Proof. Given a partition I_1, \dots, I_n of the set $\{1, \dots, 3n\}$ such that, for $j \in \{1, \dots, n\}$, $\sum_{i \in I_j} b_i = B$, we define a flow over time with inflow-dependent transit times as follows. During the time interval $[0, 1)$ we send flow at constant rate $nb_i + 1$ into arc (s, v_i) , for every $i \in \{1, \dots, 3n\}$. This flow is sent on to the nodes w_1, \dots, w_n according to the following rule. During the time interval $[0, 1)$, we set the flow rate of arc (v_i, w_j) to $b_i + 1$, if $i \in I_j$, and to b_i , otherwise. During the time interval $[1, 2)$ we send flow at constant rate nb_i into arc (s, v_i) , for every $i \in \{1, \dots, 3n\}$, and define the flow rate into arc (v_i, w_j) , $j = 1, \dots, n$, to be b_i . With these definitions, it is easy to see that at every point in time flow is entering a node w_j , $j = 1, \dots, n$, at rate bounded by $(n + 1)B + 3$. Thus, this flow can be sent immediately on to t using the arc (w_j, t) . Obviously, flow conservation holds at every point in time and no storage at intermediate nodes is used. Moreover, all $2n^2B + 3n$ units of flow arrive in t before time 2. \square

It remains to show that the existence of a flow over time f with inflow-dependent transit times of value D with time horizon at most 2 yields a

feasible solution to the underlying instance of 3-PARTITION. To do this, we need to make the following reasonable assumption on f : all flow rate functions are essentially continuous, i.e., on every arc a of the given network we require that the flow rate function f_a has at most finitely many discontinuities.

In f , we color every (infinitesimal) unit of flow either red or green. If it enters the network before time 1, it is colored red, else it is colored green. We denote the corresponding flows by f^r and f^g .

Claim 2.47. For every $\theta \in [0, 1)$, the following properties hold:

$$\int_0^\theta f_{(s,v_i)}^r(\tau) d\tau = \theta (nb_i + 1) \quad \text{for all } i \in \{1, \dots, 3n\}, \quad (2.17)$$

$$\int_1^{1+\theta} f_{(v_i,w_j)}^g(\tau) d\tau = \theta b_i \quad \text{for all } i \in \{1, \dots, 3n\}, j \in \{1, \dots, n\}. \quad (2.18)$$

Proof. After time 1 flow can enter an arc (v_i, w_j) at rate at most b_i since, otherwise, it cannot reach t before time 2. Thus, in total, at most n^2B units of green flow can be sent to t . Then, in order to satisfy the demand $2n^2B + 3n$, at least $n^2B + 3n$ units of red flow must leave s . Since the capacity of arc (s, v_i) is bounded by $nb_i + 1$, for $i \in \{1, \dots, 3n\}$, at most $\sum_{i=1}^{3n} (nb_i + 1) = n^2B + 3n$ units of red flow can be sent in total. Hence, exactly $n^2B + 3n$ units of red flow and exactly n^2B units of green flow must travel from s to t . As a consequence, (2.17) and (2.18) must hold. \square

Consider a node v_i , $i \in \{1, \dots, 3n\}$. If flow is entering an arc (v_i, w_j) , $j \in \{1, \dots, n\}$, at rate at most b_i , this flow arrives in w_j instantaneously. Otherwise, this flow needs one unit of time to reach w_j and is therefore *delayed*. We now investigate how much red flow is delayed for each node v_i .

Claim 2.48. For all $i \in \{1, \dots, 3n\}$, the following properties hold:

1. At least $b_i + 1$ units of red flow passing through v_i are delayed.
2. If exactly $b_i + 1$ units of red flow passing through v_i are delayed, then at almost every⁴ point $\theta \in [0, 1)$

$$f_{(v_i,w_j)}^r(\theta) \in \{b_i, b_i + 1\}, \quad \text{for all } j \in \{1, \dots, n\}.$$

⁴the subset of $[0, 1)$ where the property fails has Lebesgue-measure zero.

Proof. Fix $i \in \{1, \dots, 3n\}$. At most nb_i units of red flow can be sent out of v_i instantaneously during $[0, 1)$ by setting the flow rate of every arc (v_i, w_j) , $j = 1, \dots, n$, to the threshold value b_i during time interval $[0, 1)$. By (2.17), at least one additional unit of red “excess” flow has to be sent out of v_i by exceeding this threshold value on some of the arcs (v_i, w_j) , $j = 1, \dots, n$. For every arc (v_i, w_j) , $j = 1, \dots, n$, let

$$\delta_j(\theta) := \begin{cases} f_{(v_i, w_j)}^r(\theta) - b_i & \text{if } f_{(v_i, w_j)}^r(\theta) > b_i, \\ 0 & \text{otherwise,} \end{cases}$$

denote the *excess rate* of arc (v_i, w_j) , then

$$\sum_{j=1}^n \int_0^1 \delta_j(\theta) d\theta \geq 1 \quad (2.19)$$

must hold. Since $0 \leq \delta_j(\theta) \leq 1$ at every point in time $\theta \in [0, 1)$,

$$b_i + \delta_j(\theta) \geq (b_i + 1)\delta_j(\theta). \quad (2.20)$$

Whenever the excess rate $\delta_j(\theta)$ is strictly greater than zero, not only the excess flow is delayed, but all flow entering the arc (v_i, w_j) at time θ . We conclude that the total amount of delayed flow can be lower-bounded as follows:

$$\sum_{j=1}^n \int_{\theta: \delta_j(\theta) > 0} (b_i + \delta_j(\theta)) d\theta \stackrel{(2.20)}{\geq} (b_i + 1) \sum_{j=1}^n \int_0^1 \delta_j(\theta) d\theta \stackrel{(2.19)}{\geq} b_i + 1. \quad (2.21)$$

This proves the first statement of the claim.

To prove the second statement, assume that exactly $b_i + 1$ units of red flow are delayed. In a first step, we prove that at almost every point $\theta \in [0, 1)$, the excess rate $\delta_j(\theta)$ is either 0 or 1. By contradiction, assume that there exists $j \in \{1 \dots n\}$ for which the property fails; let $\Theta := \{\theta \in [0, 1) : 0 < \delta_j(\theta) < 1\}$. For all $\theta \in \Theta$, the inequality in (2.20) is strict. Since $f_{(v_i, w_j)}^r$ has only a finite number of discontinuities, so does δ_j . Hence Θ contains a small interval where δ_j is continuous and so the first inequality in (2.21) must be strict, too. Thus, more than $b_i + 1$ units of flow are delayed leading to a contradiction. We conclude that at almost every point in time, for all $j \in \{1, \dots, n\}$, either $f_{(v_i, w_j)}^r(\theta) \leq b_i$ (if $\delta_j(\theta) = 0$) or $f_{(v_i, w_j)}^r(\theta) = b_i + 1$ (if $\delta_j(\theta) = 1$). Next assume that there exists $j \in \{1 \dots n\}$ for which there is a set with Lebesgue-measure greater than zero where $f_{(v_i, w_j)}^r$ is strictly less than b_i . Then, the excess flow $\sum_{j=1}^n \int_0^1 \delta_j(\theta) d\theta$ has to be strictly greater than 1, implying that the second inequality in (2.21) is strict. Again, more than $b_i + 1$ units of flow are delayed leading to a contradiction. This proves the second statement. \square

Claim 2.49. For all $i \in \{1, \dots, 3n\}$, exactly $b_i + 1$ units of red flow passing through v_i are delayed. Moreover, all arcs (w_j, t) , $j = 1, \dots, n$, are completely filled with green flow and delayed red flow during $[1, 2)$.

Proof. It follows from (2.18) that nB units of green flow must travel via the arcs (w_j, t) , $j \in \{1, \dots, n\}$. Thus, due to capacity constraints, during the interval $[1, 2)$ at most another $nB + 3n$ units of delayed red flow can pass through all of the arcs (w_j, t) , $j = 1, \dots, n$. It then follows from the first statement in Claim 2.48 that, for all $i \in \{1, \dots, 3n\}$, exactly $b_i + 1$ units of red flow passing through v_i are delayed and all arcs (w_j, t) , $j = 1, \dots, n$, are completely filled with green flow and delayed red flow during $[1, 2)$. \square

Lemma 2.50. If an s - t -flow over time f with inflow-dependent transit times exists which sends $D = 2n^2B + 3n$ units of flow from s to t in time $T := 2$, then the underlying instance of 3-PARTITION is a 'yes'-instance.

Proof. Pick a non-empty interval $(0, \mu)$ during which all flow rate functions $f_{(v_i, w_j)}$ are continuous. By Claim 2.49 and the second part of Claim 2.48, each of these flow rates must be constant, either b_i or $b_i + 1$. Claim (2.17) together with flow conservation implies that, for each $i \in \{1, \dots, 3n\}$, at most one arc leaving v_i has a flow rate of $b_i + 1$ during $(0, \mu)$. We define partition sets as follows: for all $j \in \{1, \dots, n\}$ let I_j be the set of items b_i for which $f_{(v_i, w_j)}(\theta) = b_i + 1$ during $(0, \mu)$. Notice that no item is contained in more than one partition set. We claim that each partition set I_j satisfies $\sum_{i \in I_j} b_i = B$. If not, there exists $j \in \{1, \dots, n\}$ such that $\sum_{i \in I_j} b_i < B$. Then, less than $\mu(B + 3)$ units of delayed red flow arrive in w_j during $(1, 1 + \mu)$. Again, by (2.18), at most another μnB units of green flow arrive in w_j during $(1, 1 + \mu)$ contradicting Claim 2.49. This concludes the proof of Lemma 2.50. \square

This concludes the proof of Theorem 2.44.

2.5 COMPUTATIONAL RESULTS

In this section we report on the results of our implementation of Algorithm 1. A summary of the main steps of this algorithm is given on page 52. It is proven in Section 2.3 that this algorithm computes a $(2 + \varepsilon)$ -approximate solution to the quickest flow problem in the setting of inflow-dependent transit times.

In a first step, we will briefly describe the data structure, transit time functions, and input data used in our implementation. Afterwards, we will

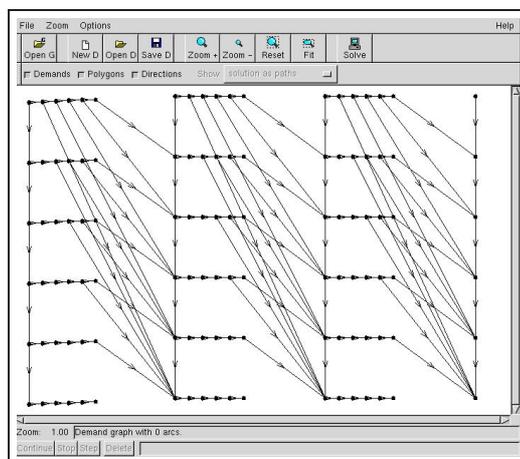


Figure 2.14: Screenshot of the graphical user interface designed by Olaf Jahn and used for our implementation; it depicts the fan graph over a path of length three.

discuss the practical performance of the algorithm for different problem instances. The implementation is based on joint work with Nadine Baumann and Lydia Franck.

Data Structure. Recall that the algorithm does not perform on the original graph G but on its expansion to a bow graph. The bow graph serves as a relaxation of the model of inflow-dependent transit times; see Section 2.2. Moreover, we have seen that flows over time in the bow graph correspond to static flows in the time-expanded bow graph, which we again refer to as the fan graph; see Section 2.3.4.

The layer structure of the considered graphs is reflected by our implemented data structure; the underlying graph G forms the base layer, the bow graph serves as intermediate layer, and the fan graph, as the time-expansion of the latter, forms the top layer. A screenshot of the graphical user interface is given in Figure 2.14. It depicts the fan graph over a path of length three.

We have realized each layer by a distinguished *traits class*; for more details on traits classes see, e.g., [51]. A software architecture based on traits classes allows to design graph algorithms that are independent of the underlying graph data structure. This approach leads to a clear separation of graph data structure and algorithms, which decreases the error-proneness of the implementation and simplifies code-reusability.

In our implementation, we provide a traits class called `GraphView` for each graph type (original graph, bow graph, and fan graph) and ensure that certain algorithmic requirements are met by all three graph types. The algorithm accesses the underlying graph data via the generic interface `GraphView`. In

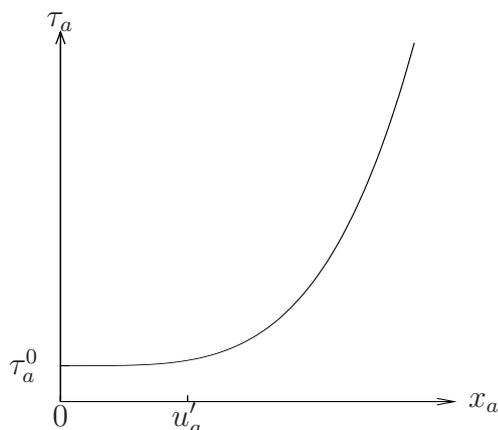


Figure 2.15: For the case of static road traffic, the U.S. Bureau of Public Roads developed a simplified function describing the dependency of the transit time on the flow.

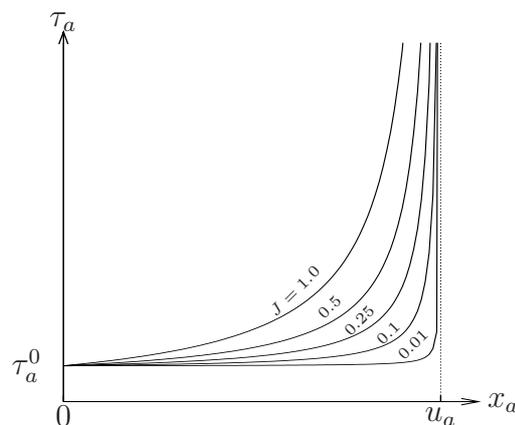


Figure 2.16: Davidson proposed an alternative family of transit time functions.

particular, the algorithm cannot distinguish between different graph types; potentially, it can be applied to all three graph layers. Another advantage of this approach is the following; with standard time-expansion, the fan graph may become very large. Using the concept of traits classes, it is possible to expand the graph only locally. The algorithm works virtually on the complete time-expanded graph; however, only those parts are actually expanded that are currently in use.

Transit Time Functions. Earlier, we have mentioned the difficulties in designing realistic traffic flow models; see Section 1.4. The dependency of the actual transit time of an arc on the current flow situation is exceedingly difficult to capture and it is unclear how to derive an appropriate and tractable mathematical model. As pointed out by Sheffi [63], the formulas which are effectively used in analyzing traffic networks are significantly simpler. According to Sheffi, a function that is often used in practice is a function developed by the U.S. Bureau of Public Roads (BPR):

$$\tau_a(x_a) := \tau_a^0 (1 + \alpha x_a / u'_a)^\beta. \quad (2.22)$$

A visualization of this function is given in Figure 2.15. Here, x_a represents the (static) flow rate on a road which is determined in practice, for instance, by taking the average number of cars entering the road during a pre-specified time interval. The value τ_a^0 is the *free-flow travel time* which measures the time it takes to traverse the empty road. The *practical capacity* u'_a is the flow

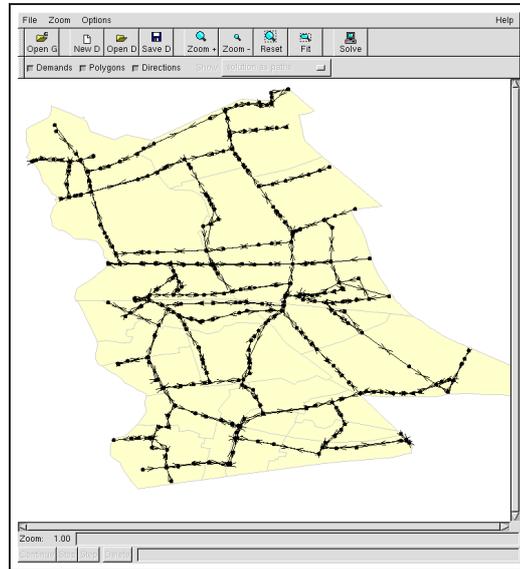


Figure 2.17: Screenshot of instance hld_97_96.

rate at which the transit time of a road is 15% larger than the free-flow travel time. The quantities α and β are model parameters, for which the values $\alpha := 0.15$ and $\beta := 4$ are typically chosen. Figure 2.15 depicts the transit time function for these parameters. Note that, for this choice of α , the transit time function evaluated at the practical capacity u'_a is indeed 15% higher than the free-flow transit time τ_a^0 . For our test instances, we have chosen the BPR function with parameter α set to 0.15 and β varying between 1 (linear) and 4 (strictly convex).

A different function which is frequently used according to Sheffi is Davidson's function. It is defined as follows:

$$\tau_a(x_a) := \tau_a^0 \left(1 + J \frac{x_a}{u_a - x_a} \right).$$

Here, J is a parameter of the model which typically varies between 0.1 and 1.0; see Figure 2.16 for an illustration.

Input Data. We have tested our algorithms on real-life instances given by parts of the street network of Berlin. One of these instances is depicted in Figure 2.17. Since the input data associated with each road complies with the parameters appearing in the BPR function, we have picked this function model for our test runs.

We have selected six sections of Berlin, ranging in the size between 201 and 12100 nodes (street crossings) and 339 and 19570 arcs (streets). In

Table 2.1: List of instances in order of increasing size; for each of the six sections of Berlin, three different commodity pairs are selected. They are listed in order of increasing geographic distance between source and sink node.

Instance	#nodes	#arcs	dist [m]
fdh_81_70	201	339	761
fdh_70_70			2280
fdh_81_77			4370
hld_97_96	345	528	1916
hld_10_96			4627
hld_96_96			8546
kpn_94_96	419	635	2786
kpn_97_99			10931
kpn_99_10			25067
zld_41_66	461	724	3824
zld_44_66			7948
zld_67_56			14043
csw_14_25	1800	2935	3692
csw_94_7			8695
csw_56_54			17670
bln_50_64	12100	19570	6739
bln_10_11			20854
bln_13_13			62987

each such section we have picked three source-sink pairs. The resulting 18 instances are listed in Table 2.1. The last column in the table shows the geographic distance between the source and the sink node.

Computational Experience. In this paragraph we examine the practical performance of Algorithm 1. The experiments were conducted on a 1.7 GHz AMD Athlon(TM) XP 2100+ machine running under Linux with 512MByte memory. The code is written in C++ and has been compiled with the GNU gcc-Compiler version 3. All running times are measured in seconds.

The main step of Algorithm 1 is to determine a quickest *s-t*-flow in the bow graph. As suggested by Burkard, Dlaska, and Klinz [6], we use a Newton-type search algorithm to find the optimal time horizon \bar{T}^B ; this search method is known to be very efficient for solving practical problems. In each search step, we solve a static minimum cost circulation problem to determine a max-

imum flow over time for the current search estimate. To solve the minimum cost circulation problem, we have implemented the successive shortest path algorithm [7, 35]. In the worst case, the successive shortest path algorithm needs an exponential number of iterations to compute the optimal solution. However, in practice it performs very well.

We have selected three instances, fdh_70_70, csw_94_7, and kpn_97_99, which we discuss in more detail in Tables 2.2, 2.3, and 2.4, respectively. Recall that the theoretical performance guarantee of the algorithm is $\rho = 2 + \varepsilon$. We have set $\varepsilon = 0.1$, hence the time horizon of the computed solution exceeds the optimal time horizon by at most a multiplicative factor of 2.1.

The tables are structured as follows; In the first column, different demands d are listed. The demand is the amount of flow (the number of cars) that is sent from the source to the sink. In our tests, the demand varies between 100 and 10000. In the succeeding columns, various indicators of the performance of Algorithm 1 are shown. In the following, we explain these indicators.

- \overline{T}^B denotes the time horizon of a quickest flow in the bow graph; see Step 4 of Algorithm 1. The value \overline{T}^B is a lower bound on the time horizon of a quickest flow with inflow-dependent transit times in the original graph.
- T_p^B is the time horizon of the inflow-preserving temporally repeated flow in the bow graph that is computed in Step 5 of Algorithm 1. The subscript alludes to the procedure in which flow is *pushed* from “fast” bow arcs up to the “slowest” flow-carrying bow arcs. In the theoretical analysis of the algorithm it is proven that T_p^B is at most twice as large as T^B .
- $d(2T_p^B)$ is the value of the inflow-preserving temporally repeated flow for time horizon $2T_p^B$; see Step 5 of Algorithm 1. It follows from the analysis of the algorithm that $d(2T_p^B) \geq d$.
- T denotes the time horizon of the final solution. To evaluate the practical performance of the algorithm we consider the quotient of T and the lower bound \overline{T}^B .
- $|\mathcal{P}|$ denotes the number of paths in the path decomposition of our final solution. As an indicator of *fairness*, we have chosen the maximum transit time of a flow path divided by the minimum transit time of a flow path in the solution.

Table 2.2: Instance fdh_70_70

d	\bar{T}^B	T_p^B	$d(2T_p^B)$	T	T/\bar{T}^B	$ \mathcal{P} $	fairness
100	469.1	660.1	239	682.3	1.45	2	1.00
1000	1693.7	2148.0	1964	2181.1	1.29	5	1.02
2000	2979.4	3433.7	3964	3466.9	1.16	5	1.02
5000	6836.5	7290.9	9964	7324.0	1.07	5	1.02
10000	13265.1	13719.4	19964	13752.6	1.04	5	1.02

Table 2.3: Instance kpn_97_99

d	\bar{T}^B	T_p^B	$d(2T_p^B)$	T	T/\bar{T}^B	$ \mathcal{P} $	fairness
100	1871.1	2453.0	315	2506.0	1.34	2	1.00
1000	7271.1	7853.0	2115	7906.0	1.09	2	1.00
2000	13271.1	13853.0	4115	13906.0	1.05	2	1.00
5000	31271.1	31853.0	10115	31906.0	1.02	2	1.00
10000	61271.1	61853.0	20115	61906.0	1.01	2	1.00

Table 2.4: Instance csw_94_7

d	\bar{T}^B	T_p^B	$d(2T_p^B)$	T	T/\bar{T}^B	$ \mathcal{P} $	fairness
100	1644.7	1876.7	692	1919.2	1.17	6	1.01
1000	3115.9	3897.4	2556	3946.4	1.27	10	1.01
2000	4615.9	5397.4	4556	5446.4	1.18	10	1.01
5000	9115.9	9897.4	10556	9946.4	1.09	10	1.01
10000	16615.9	17397.4	20556	17446.4	1.05	10	1.01

We only discuss the results presented in Table 2.2. Notice, however, that the results given in Table 2.3 and 2.4 have a very similar flavor and thus lead to the same conclusions. First, consider the performance ratio T/\bar{T}^B . It is always remarkably better than the theoretical performance ratio $\rho = 2.1$. The maximum performance ratio attained for this instance is 1.45. Also notice that, with increasing demand d , the practical performance improves considerably; for $d = 10000$, the algorithm achieves performance ratio 1.04. Secondly, the value $d(2T_p^B)$ is always clearly larger than the demand d ; on this instance, it is about twice as large as d . Thirdly, the fairness of the solution remains unchanged for $d \geq 1000$.

We give an explanation for this behavior. Recall that the algorithm com-

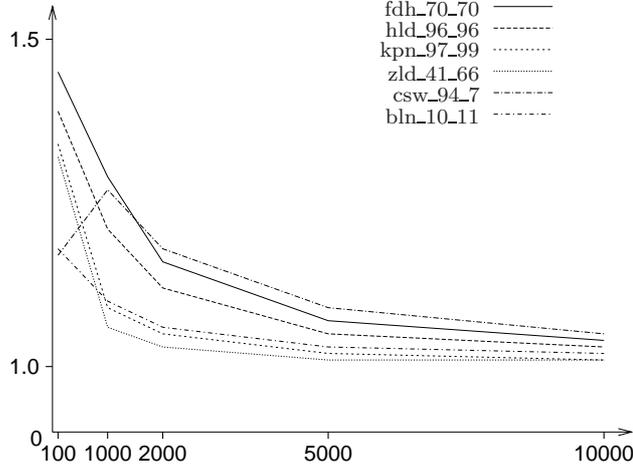


Figure 2.18: Upper bound T/\overline{T}^B on the performance ratio of the algorithm for different demands d .

puts a good temporally repeated flow solution. For large T , an optimal temporally repeated flow solution approximates a quickest flow very well. Namely, it is not difficult to see that the value of an s - t -flow over time with time horizon T is always upper-bounded by $T\delta$, where δ is the value of a minimum s - t -cut in G . On the other hand, the value of a temporally repeated flow that is generated from a maximum static s - t -flow is equal to $T\delta - \sum_{a \in A} \tau_a(x_a)x_a$; see (1.13) on page 29. For large T , this term is dominated by $T\delta$ and hence the temporally repeated flow is optimal in the limit.

In our test runs, this process seems to converge very fast. This explains why the quality of the solution improves for large demand d and why the fairness does not change for $d \geq 1000$.

Figure 2.18 visualizes the correlation between performance ratio and demand for six selected instances. Table 2.5 shows the average (\emptyset) and the maximum (max) performance ratio for different demands d . The average and the maximum are taken over all instances listed in Table 2.1. Again, we conclude that the performance ratio improves significantly with increasing demand d .

Next, we examine the dependency of the performance ratio on parameter β in more detail. The parameter β controls the degree of convexity of the transit time function; see (2.22) on page 76. In Section 2.3.3 we prove that the performance ratio of Algorithm 1 is bounded by $2 + \varepsilon$, for general

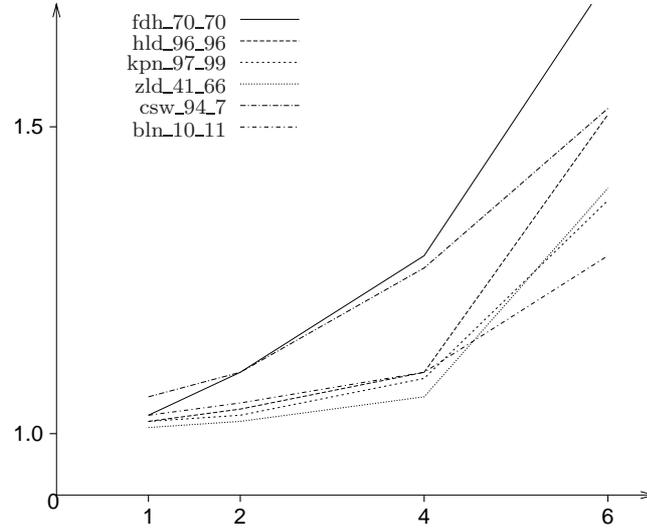


Figure 2.19: Performance ratio T/\bar{T}^B of the algorithm for different values of parameter β .

transit time functions, and it is bounded by $3/2 + \varepsilon$, for concave transit time functions. On the other hand, we present a family of instances showing that both bounds are tight; see Section 2.3.3. On these instances, the performance guarantee approaches 2 as the degree of convexity of the underlying transit time functions increases.

We have examined this behavior on our test instances. Figure 2.19 visualizes the dependency of the performance ratio on the parameter β for six instances. Table 2.6 depicts the average and the maximum performance ratio taken over all instances listed in Table 2.1. The results confirm that the quality of the solution degrades considerably with increasing parameter β .

We now examine the running time of the algorithm with respect to the parameters ε and β . In this context, we also discuss the size of the bow graph. The parameter ε controls the quality of the computed solutions.

Table 2.5: Average (\emptyset) and maximum (max) performance ratio T/\bar{T}^B of the algorithm for different demands d . Average and maximum are taken over all instances listed in Table 2.1.

inst.	$d = 100$	$d = 1000$	$d = 2000$	$d = 5000$	$d = 10000$
\emptyset	1.27	1.21	1.14	1.07	1.04
max	1.45	1.62	1.58	1.29	1.16

Table 2.6: Average (\emptyset) and maximum (max) performance ratio T/\overline{T}^B of the algorithm for different values of parameter β . Average and maximum are taken over all instances listed in Table 2.1.

inst.	1	2	4	6
\emptyset	1.04	1.07	1.21	1.46
max	1.07	1.18	1.62	1.68

More precisely, it determines the accuracy of the step functions, which are computed in Step 2 of the algorithm to approximate the original transit time functions. In particular, the precise choice of ε determines the number of breakpoints in each step function. Hence, the parameter ε has an immediate impact on the size of the bow graph and thus on the running time of the algorithm.

Figure 2.20 illustrates the trade-off between accuracy and size of the bow graph (respectively, running time). On the left hand side, the figure displays

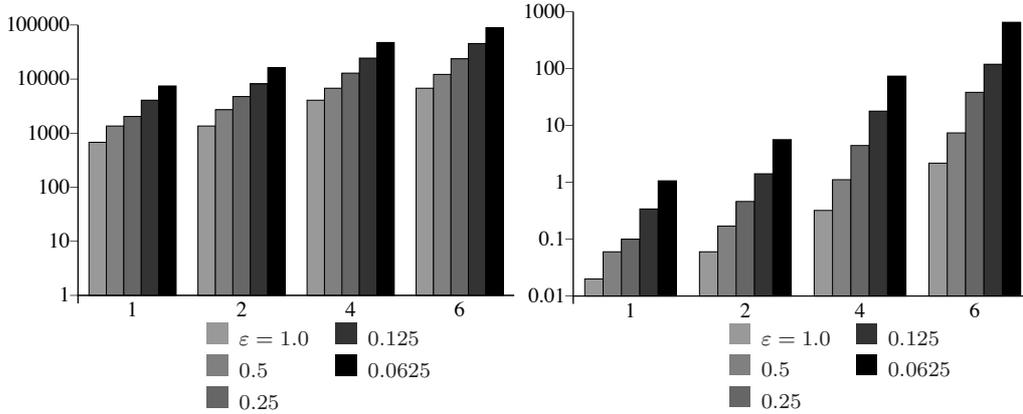


Figure 2.20: Instance `fdh_70_70`; size of the bow graph (left) and running time (in seconds) of the algorithm (right) depending on the choice of parameters ε and β . The values are plotted in logarithmic scale. The parameter β is chosen from $\{1, 2, 4, 6\}$, the demand d is fixed to 1000.

the number of arcs in the bow graph with respect to parameters ε and β for instance `fdh_70_70`. The number of arcs in the bow graph is plotted in logarithmic scale. The parameter ε varies between 0.0625 and 1, the parameter β is chosen from $\{1, 2, 4, 6\}$. For $\varepsilon = 0.0625$ and $\beta = 4$, the bow graph contains already 88810 arcs while the original graph of instance `fdh_70_70` contains only 339 arcs; see Table 2.1.

As it can be seen in Figure 2.20, the number of arcs in the bow graph

is also correlated with parameter β . The reason is the following. With increasing parameter β , the range of possible transit times on an arc increases; see (2.22) on page 76. Consequently, the number of arcs in the bow graph increases.

The running time of the algorithm is documented on the right hand side of Figure 2.20; it is plotted in logarithmic scale. For $\beta \leq 4$, the running time is never larger than 75 seconds. For $\beta = 6$, the algorithm needs already more than 10 minutes to attain accuracy $\varepsilon = 0.0625$.

In Table 2.7, the size of the bow graph and the running time of the algorithm are shown for six selected instances. We have chosen demand $d = 1000$, $\beta = 4$, and $\varepsilon = 0.1$. The bow graph of the largest instance `bln_10_11` contains more than 1.7 million arcs and the algorithm needs more than half an hour to compute a solution.

Table 2.7: Number of arcs in the bow graph ($|A^B|$) and CPU (in seconds) for demand $d = 1000$ and parameter $\beta = 4$.

inst.	$ A^B $	CPU
<code>fdh_70_70</code>	30510	27.9
<code>hld_96_96</code>	47520	25.2
<code>kpn_97_99</code>	57150	32.5
<code>zld_41_66</code>	65160	29.2
<code>csw_94_7</code>	264150	384.0
<code>bln_10_11</code>	1761300	2036.7

In Section 2.3.5 the algorithm of Köhler and Skutella [44] is described, which computes quickest flows in the setting of load-dependent transit times. An overview of the main steps of the algorithm is given on page 63. We again refer to this algorithm as Algorithm 2.

It is shown that Algorithm 2 is also applicable in the setting of inflow-dependent transit times. We have implemented this algorithm to compare its practical performance with that of Algorithm 1. Both algorithms have a theoretical performance guarantee of $\rho = 2 + \varepsilon$. In our test runs, we have set $\varepsilon = 0.1$.

In Figure 2.21, we examine the practical performance ratios of both algorithms for five selected instances. In these test runs, we have set $d = 100$. On the left hand side, the parameter β is set to 1 (linear transit time functions), on the right hand side, it is set to 4 (strictly convex transit time functions).

For $\beta = 1$, Algorithm 1 performs significantly better than Algorithm 2. For $\beta = 4$, the latter algorithm seems to be preferable. It performs better on

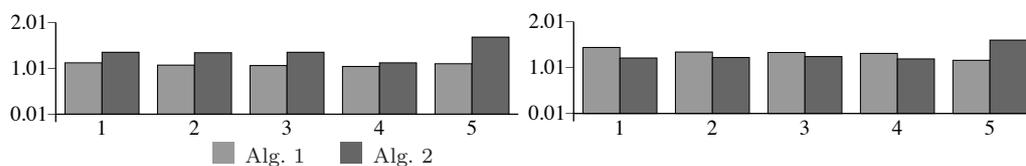


Figure 2.21: Performance ratios of Algorithm 1 and Algorithm 2 for parameter $\beta = 1$ (left) and $\beta = 4$ (right); we have selected five instances (1=fdh_70_70, 2=hld_96_96, 3=kpn_97_99, 4=zld_41_66, 5=csw_94_7), the demand d is set to 100.

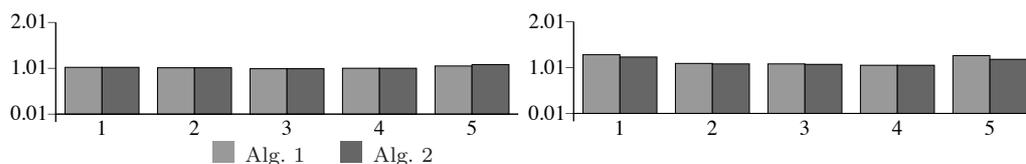


Figure 2.22: As in Figure 2.21 with demand d set to 1000.

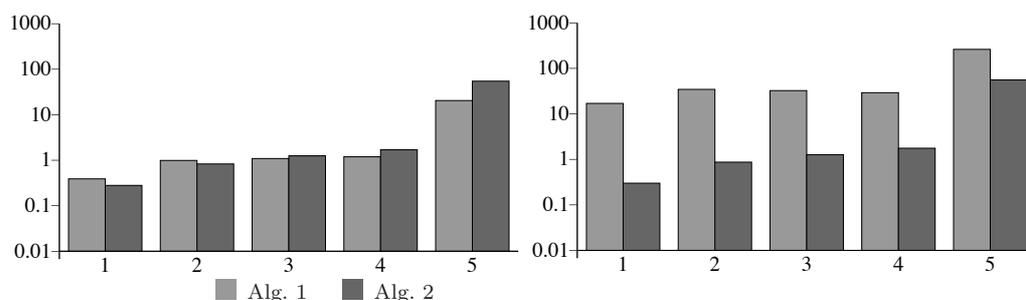


Figure 2.23: Running times of Algorithm 1 and Algorithm 2 for parameter $\beta = 1$ (left) and $\beta = 4$ (right); we have selected five instances (1=fdh_70_70, 2=hld_96_96, 3=kpn_97_99, 4=zld_41_66, 5=csw_94_7), the demand d is set to 100. The running time (in seconds) is plotted in logarithmic scale.

all instances except on the largest instance bln_10_11. This result underlines that Algorithm 1 does not perform too well if transit times are “highly convex”. For demand $d = 1000$, both algorithms seem to perform equally well; see Figure 2.22.

Figure 2.23 shows the running times of both algorithms measured in logarithmic scale. The demand d is fixed to 100. For $\beta = 1$ (left), the running times of both algorithms are similar; in particular, they remain below one minute. Not surprisingly, for $\beta = 4$, Algorithm 1 is significantly slower than Algorithm 2. Recall that parameter β has a direct impact on the size of the bow graph and thus on the running time of Algorithm 1; compare Figure 2.20. Algorithm 2 seems to be unaffected by the choice of β .

CHAPTER 3

QUICKEST MULTI-COMMODITY FLOWS

3.1 INTRODUCTION

The main focus of this chapter is on the quickest multi-commodity flow problem. An overview of the results known for fixed transit times is given in Chapter 1. In this chapter we present approximation algorithms designed for the more general setting of inflow-dependent transit times. In a first step, we develop a relaxation which reflects flows over time with inflow-dependent transit times more accurately than the model presented in Section 2.2. Based on the new relaxation, we present both, a constant factor approximation algorithm and an FPTAS for the quickest multi-commodity transshipment problem with bounded cost. Since the problem is \mathcal{NP} -hard already for the single source, single sink case, this result is the best we can hope for.

In the single source, single sink case, we are able to approximate quickest flows by considering a corresponding relaxed instance given by the bow graph; see Section 2.3. In the bow graph, transit times are fixed and thus we can apply an exact polynomial time algorithm for computing quickest s - t -flows. In the multi-commodity case, we are facing the following difficulties when following the same approach: Hall, Hippler, and Skutella [27] prove that the multi-commodity flow over time problem is already \mathcal{NP} -hard for fixed transit times. Therefore, we can only compute approximate quickest multi-commodity flows in the bow graph. Furthermore, there does not always exist a good temporally repeated flow solution as the example given at the end of Section 2.3 shows; an instance with k commodities is presented, for which any temporally repeated flow with flow-dependent transit times needs at least k times longer than the optimal solution.

Due to these difficulties, we need to consider a relaxation that is stronger than the one presented in Section 2.2. Using this new relaxation, we can approximate a quickest multi-commodity transshipment with inflow-dependent transit times within arbitrary precision.

However, the constant approximation algorithms presented in Chapter 2 for quickest s - t -flows do not become obsolete. These algorithms solely rely on standard network flow techniques and therefore they are very efficient as the

computational results presented at the end of Chapter 2 confirm. Moreover, they generate solutions that have a simple structure. These two aspects make the algorithms very attractive for application purposes.

Both algorithms presented in this chapter rely on static flow formulations with additional bundle constraints linking the flow values of several arcs. In particular, we need general linear programming techniques to solve these problems. The constant factor approximation computes fairly well-structured solutions; the flow rate on each arc follows a temporally repeated pattern. Since the optimal flow might require a much more complicated time-varying structure, the FPTAS must rely on a model that reflects the parameter time with a higher resolution. Our FPTAS utilizes a time-expansion with a fine discretization of time in order to attain a good performance guarantee. Hence, the trade-off between quality and efficiency of the considered algorithms should not be disregarded.

Recall that in the multi-commodity case we are given a set of commodities $K = \{1, \dots, k\}$; every commodity $i \in K$ is defined by a set of sources and sinks with given supplies and demands. To simplify notation, we restrict to the case of only one source s_i and one sink t_i with given demand $d_i > 0$, for each commodity i . However, our results can be generalized directly to the case of several sources and sinks: Our algorithms rely on static network flow formulations. In particular, a transshipment problem can easily be reduced to a single source, single sink problem by introducing a super source and a super sink; see Section 1.2.

Without loss of generality we assume that no two commodities have the same source node. Otherwise we introduce a new source node s'_i , for each commodity i , and connect it to s_i by an arc with transit time zero and infinite capacity. Then, the objective is to send d_i units of flow from s'_i to t_i .

A large part of this chapter is based on joint work with Martin Skutella and Alex Hall. An extended abstract appeared in [28].

3.2 A STRONGER RELAXATION

In this section we define a relaxation of inflow-dependent transit times that is stronger than the one presented in Section 2.2. Throughout this section we make the following assumption:

Assumption. All transit time functions are given as piecewise constant, nondecreasing, and left-continuous functions¹ $(\tau_a^s)_{a \in A}$.

¹The transit time function of arc a is denoted τ_a^s to stress its step function character.

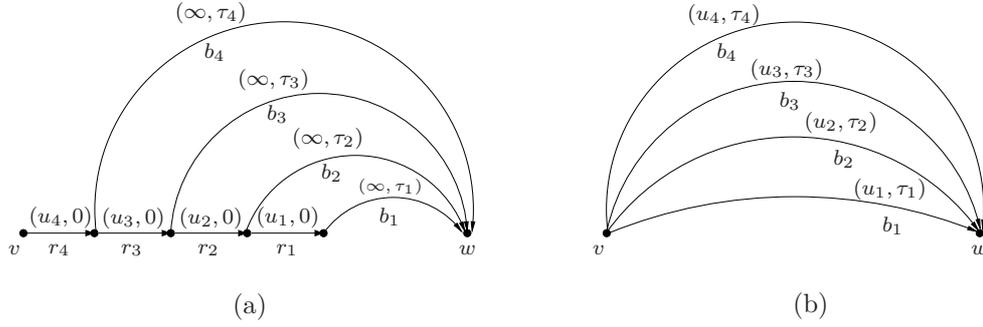


Figure 3.1: Comparison of both bow graph models. Figure (a) shows the expansion of a single arc $a = (v, w)$ as defined in Section 2.2.1; compare Figure 2.1. Figure (b) depicts the simpler expansion used in this chapter.

3.2.1 The New Bow Graph

We begin with the definitions of a bow graph that is very similar to the one defined in Section 2.2.1. To simplify notation, we again refer to it as G^B . Throughout this chapter we only work on this simpler bow graph.

The bow graph, denoted $G^B = (V^B, A^B)$, is defined on the same node set as G , i.e., $V^B := V$, and is obtained by creating several copies of an arc, one for each possible transit time on the arc. Every arc $e \in A^B$ has a capacity u_e , a constant transit time $\tau_e \in \mathbb{R}^+$, and costs $c_{e,i}$, $i \in K$.

For the definition, let us consider a specific arc $a \in A$. We assume that the transit time function τ_a^s is given by breakpoints $0 = u_0 < u_1 < \dots < u_\ell = u_a$ and corresponding transit times $\tau_1 < \dots < \tau_\ell$. Flow entering at rate $x \in (u_{j-1}, u_j]$ needs τ_j time units to traverse arc a . In the bow graph, arc a is replaced by ℓ copies b_1, \dots, b_ℓ of a . These *bow arcs* represent all possible transit times of arc a . Namely, the transit time of arc b_j is given by τ_j , its capacity is chosen as u_j , $j = 1, \dots, \ell$. We denote the set of bow arcs corresponding to a by A_a^B and refer to A_a^B as the *expansion* of arc a . The cost coefficients of every arc $e \in A_a^B$ are identical to those of a , i.e., $c_{e,i} := c_{a,i}$, for $i \in K$. For every arc $e \in A_a^B$, let $a(e)$ denote the original arc a .

The main difference to the bow graph introduced in Section 2.2.1 can be seen in Figure 3.1: in the new model, we omit the regulating arcs which, in the old model, limit the amount of flow entering the bow arcs. In particular, all bow arcs representing the same original arc “share” capacity. In the new model, capacities are directly assigned to the bow arcs. They no longer share capacities. Moreover, we include arc costs in the new model.

3.2.2 Relaxation Property of the New Model

We now discuss the relationship between flows over time in the bow graph G^B and flows over time with inflow-dependent transit times in G . Any flow over time f with inflow-dependent transit times $(\tau_a^s)_{a \in A}$ in G with time horizon T and cost C can be interpreted as a flow over time f^B (with constant transit times) in G^B with same time horizon T and same cost C : If, in the original graph G , flow is entering arc $a \in A$ at time θ with flow rate $f_a(\theta)$, then, in the bow graph, this flow is sent onto the bow arc $e \in A_a^B$ representing transit time $\tau_a^s(f_a(\theta))$. Costs are preserved, since the cost of every arc $e \in A_a^B$ is identical to the cost of arc a .

However, an arbitrary flow over time f^B in G^B does not correspond to a flow over time f with inflow-dependent transit times $(\tau_a^s)_{a \in A}$ in G . In addition, we have to require the following property: For every original arc $a \in A$ and at every point in time θ , the flow f^B sends flow into at most one bow arc $e \in A_a^B$. This property ensures that flow units entering arc a at the same point in time θ travel through a at the same speed. As in Section 2.2, we call a flow over time *inflow-preserving* if it fulfills this property.

Using the same terminology as in Section 2.2.2, we let $\mathcal{F}(T)$ denote the set of flows over time in G with inflow-dependent transit times $(\tau_a^s)_{a \in A}$ and time horizon T . Similarly, let $\mathcal{F}^B(T)$ denote the set of flows over time in G^B (with constant transit times) and time horizon T . Let $\iota : \mathcal{F}(T) \rightarrow \mathcal{F}^B(T)$ be the embedding that maps a flow over time $f \in \mathcal{F}(T)$ to the corresponding inflow-preserving flow over time f^B in G^B .

Observation 3.1. Let f be a flow over time with inflow-dependent transit times $(\tau_a^s)_{a \in A}$ in G that satisfies the multi-commodity demands within time T at cost C . Then $\iota(f)$ defines an inflow-preserving flow over time (with constant transit times) in G^B which satisfies the multi-commodity demands within time T at cost C .

Not every inflow-preserving flow over time in G^B lies in the image of ι . An inflow-preserving flow over time can travel through the expansion of arc a via a bow arc with transit time larger than prescribed by the transit time function. As in Section 2.2.2, we define a mapping π that projects any inflow-preserving flows over time in $\mathcal{F}^B(T)$ onto the set $\mathcal{F}(T)$.

Let f^B be an inflow-preserving flow over time in $\mathcal{F}^B(T)$ with cost C . We define a flow over time f with inflow-dependent transit times in G as follows. Consider an arc $a \in A$ and let b_1, \dots, b_ℓ be the set of bow arcs of arc a . For

any point in time $\theta \in [0, T)$, we set

$$f_a(\theta) := \sum_{j=1}^{\ell} f_{b_j}^B(\theta). \quad (3.1)$$

Claim 3.2. The flow f defines a feasible multi-commodity flow over time with inflow-dependent transit times $(\tau_a^s)_{a \in A}$ in G with time horizon T and cost C .

Proof. Consider a fixed point in time θ . There is a unique bow arc $b^a \in A_a^B$ with $f_a(\theta) = f_{b^a}^B(\theta)$ because f^B is inflow-preserving. Since f^B satisfies the capacity constraints, it follows that $f_a(\theta) = f_{b^a}^B(\theta) \leq u_{b^a} \leq u_a$. Hence, the flow f satisfies the capacity constraints as well.

Next, we show that f satisfies the flow conservation constraints. Since storage of flow at intermediate nodes is allowed, it suffices to prove that flow in f travels through arc $a \in A$ not slower than flow in f^B and therefore reaches $\text{head}(a)$ on time. Flow in f entering arc a at time θ needs $\tau_a^s(f_a(\theta))$ time to reach $\text{head}(a)$. Flow in f^B entering the expansion of arc a at time θ needs τ_{b^a} time to reach $\text{head}(a)$. Since the function τ_a^s is nondecreasing, it follows that $\tau_a^s(f_a(\theta)) = \tau_a^s(f_{b^a}^B(\theta)) \leq \tau_a^s(u_{b^a})$. By definition of transit times in the bow graph, $\tau_a^s(u_{b^a}) = \tau_{b^a}$. We conclude that $\tau_a^s(f_a(\theta)) \leq \tau_{b^a}$ and thus flow in f travels not slower than flow in f^B .

Obviously, the cost of f is equal to the cost of f^B , because the cost coefficient of every bow arc in A_a^B is identical to that of arc a . \square

Let π denote the projection that maps an inflow-preserving multi-commodity flow over time f^B in $\mathcal{F}^B(T)$ to the corresponding flow f in $\mathcal{F}(T)$; see Claim 3.2. Notice that $\pi \circ \iota : \mathcal{F}(T) \rightarrow \mathcal{F}(T)$ is the identity map.

Observation 3.3. Let f be an inflow-preserving multi-commodity flow over time in G^B that satisfies the multi-commodity demands within time T at cost C . Then $\pi(f)$ defines a flow over time with inflow-dependent transit times $(\tau_a^s)_{a \in A}$ in G which satisfies the multi-commodity demands within time T at cost C .

The set of inflow-preserving flows over time is not convex. In particular, it is difficult to compute inflow-preserving flows directly. Therefore, we also consider a relaxed notion which can be interpreted as a convexification of inflow-preserving flows: For any bow arc $e \in A^B$, let $\lambda_e(\theta) := f_e^B(\theta)/u_e$ denote the *per capacity inflow rate* into arc e .

Definition 3.4. A flow over time f^B with time horizon T in G^B is called *weakly inflow-preserving* if $\sum_{e \in A_a^B} \lambda_e(\theta) \leq 1$ for all $a \in A$ and $\theta \in [0, T)$.

We let $\mathcal{F}_\lambda^B(T)$ denote the set of weakly inflow-preserving flows over time in G^B with time horizon T . The following observation summarizes the relaxation property of $\mathcal{F}_\lambda^B(T)$.

Observation 3.5. Let f be a multi-commodity flow over time with inflow-dependent transit times $(\tau_a^s)_{a \in A}$ in G that satisfies the multi-commodity demands within time T at cost C . Then $\iota(f)$ defines a (weakly) inflow-preserving multi-commodity flow over time in G^B which satisfies the multi-commodity demands within time T at cost C .

Remark 3.6. In Section 2.2 we have worked with a slightly different bow graph; see Figure 3.1 (a) on page 89. It is not difficult to observe that the set of weakly inflow-preserving flows over time $\mathcal{F}_\lambda^B(T)$ can be embedded into the set of flows over time in the bow graph defined as in Figure 3.1 (a).

To see this, consider a weakly inflow-preserving flow over time f in the expansion shown in Figure 3.1 (b). We define a flow over time \tilde{f} in the expansion depicted in Figure 3.1 (a) in the canonical way: If at time θ flow in f is entering bow arc b_i at rate $f_{b_i}(\theta)$, send this flow through the regulating arc r_i into the corresponding bow arc b_i . The flow \tilde{f} defines a feasible flow over time: The flow rate into arc r_i at time θ is given by

$$\tilde{f}_{r_i}(\theta) = \sum_{j=1}^i f_{b_j}(\theta) = \sum_{j=1}^i \lambda_{b_j}(\theta) u_{b_j} \leq u_{b_i},$$

where u_{b_i} is the capacity of bow arc b_i in the bow graph defined as in Figure 3.1 (b). By definition, u_{b_i} is equal to the capacity u_{r_i} of regulating arc r_i . We conclude that $\tilde{f}_{r_i}(\theta) \leq u_{r_i}$ and thus \tilde{f} defines a feasible flow over time.

Therefore, the model of weakly inflow-preserving flows over time indeed constitutes a relaxation which is at least as strong as the relaxation presented in Section 2.2.

The basic idea of the approximation algorithms presented in this chapter is to compute *weakly* inflow-preserving flows over time in an appropriate bow graph and to turn them into inflow-preserving flows over time afterwards. Applying the projection π to such an inflow-preserving flow over time, we derive a solution to the original problem. The following lemma and its corollary make this approach work. Consider the expansion of a single arc $a \in A$ to bow arcs $A_a^B = \{b_1, \dots, b_\ell\}$.

Lemma 3.7. Let f^B be a weakly inflow-preserving multi-commodity flow over time with time horizon T in A_a^B and $\delta > 0$. Then, f^B can be turned into an inflow-preserving multi-commodity flow over time \hat{f}^B in A_a^B such that every (infinitesimal) unit of flow in \hat{f}^B reaches $\text{head}(a)$ with a delay of at most δ .

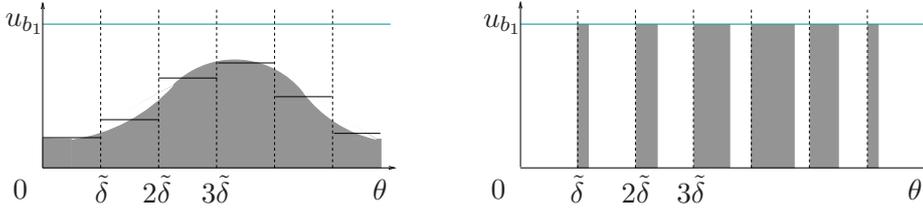


Figure 3.2: Original flow rate on bow arc b_1 and modified flow rate produced by buffering in $\text{tail}(b_1)$.

Proof. For every bow arc b_i , $i = 1, \dots, \ell$, we set up a buffer B_i in $\text{tail}(a)$ for temporary storage of flow. The buffer B_i is collecting all flow in f^B which is about to be shipped through bow arc b_i . It can output this flow in a first-in-first-out manner, i.e., flow particles must enter and leave the buffer in the same order. Buffer B_i has only two output modes. Either it is *closed*, then no flow is leaving the buffer, or it is *open* and flow is leaving the buffer at constant rate u_{b_i} , immediately entering arc b_i . In our modified solution \hat{f}^B , at every point in time at most one of the buffers B_i , $i = 1, \dots, \ell$, will be open. This guarantees that \hat{f}^B is inflow-preserving.

As above, let $\lambda_e(\theta) := f_e^B(\theta)/u_e$ be the per capacity inflow rate of f^B on arc $e \in A_a^B$ at time θ . We partition the time horizon into intervals of length $\tilde{\delta}$, where $\tilde{\delta} := \delta/2$. Let $\lambda_{e,j}$ be the average per capacity inflow rate on arc $e \in A_a^B$ during time interval $[(j-1)\tilde{\delta}, j\tilde{\delta})$, i.e.,

$$\lambda_{e,j} := \frac{1}{\tilde{\delta}} \int_{(j-1)\tilde{\delta}}^{j\tilde{\delta}} \lambda_e(\theta) d\theta,$$

$j = 1, \dots, \lceil T/\tilde{\delta} \rceil$. We define the modified flow \hat{f}^B as follows: During the first $\tilde{\delta}$ -round, all buffers are closed. During each following $\tilde{\delta}$ -round, we open the buffers in a ‘round robin’ fashion. More precisely, during time interval $[j\tilde{\delta}, (j+1)\tilde{\delta})$, we first open buffer B_1 for $\lambda_{b_1,j}\tilde{\delta}$ time, then buffer B_2 for $\lambda_{b_2,j}\tilde{\delta}$ time, and so on. Since f^B is weakly inflow-preserving, $\sum_{i=1}^{\ell} \lambda_{b_i,j} \leq 1$ holds and the last buffer is closed again before the end of this $\tilde{\delta}$ -round. Figure 3.2 illustrates how the buffer changes the original inflow rate of bow arc b_1 .

We show that the buffers are never empty while they are open. Consider bow arc b_i . During interval $[(j-1)\tilde{\delta}, j\tilde{\delta})$, the flow f^B sends $\tilde{\delta}\lambda_{b_i,j}u_{b_i}$ units of flow into bow arc b_i . This is exactly the amount of flow that the corresponding buffer B_i is sending out during the succeeding interval $[j\tilde{\delta}, (j+1)\tilde{\delta})$. Hence buffer B_i is never empty while it is open and, in particular, every unit of flow is delayed for at most $2\tilde{\delta} = \delta$ time. \square

For $\delta > 0$, we call a multi-commodity flow over time f^B in G^B δ -resting if, for every commodity i and for every node $v \neq s_i$, all flow arriving at v is stored there for at least δ time units before it moves on. A weakly inflow-preserving multi-commodity flow over time f^B in G^B which is δ -resting can easily be interpreted as an inflow-preserving flow over time \hat{f}^B : Consider a single arc $a \in A$ and its expansion A_a^B . Applying Lemma 3.7, the multi-commodity flow over time f^B restricted to A_a^B can be modified to an inflow-preserving multi-commodity flow over time such that every unit of flow is delayed by at most δ . The resting property of f^B makes up for this delay and ensures that every such flow unit can continue its way on time. Applying the projection map π , the flow \hat{f}^B can be turned into a flow over time f in G with inflow-dependent transit times $(\tau_a^s)_{a \in A}$.

Corollary 3.8. Let f^B be a weakly inflow-preserving multi-commodity flow over time in G^B with time horizon T which is δ -resting. Then, f^B can be turned into a multi-commodity flow over time f in G with inflow-dependent transit times $(\tau_a^s)_{a \in A}$ and time horizon T satisfying the same demands as f^B at the same cost as f^B . Moreover, the flow over time f_i , $i \in K$, is given by piecewise constant functions $(f_{a,i})_{a \in A}$ such that the number of breakpoints of $f_{a,i}$ can be bounded by $2|A_a^B| \lceil T/\delta \rceil$.

Proof. The transformation is described above. We claim that it preserves cost. Throughout the transformation of f^B no flow is rerouted. We only make use of storage at nodes; see Lemma 3.7. Moreover, the projection π preserves cost; see Observation 3.3. Thus, the claim follows. \square

3.3 A CONSTANT FACTOR APPROXIMATION FOR QUICKEST FLOWS

In this section we present a $(2 + \varepsilon)$ -approximation algorithm for the quickest multi-commodity flow problem with inflow-dependent transit times. The algorithm consists of the following three main steps. First, the original transit times $(\tau_a)_{a \in A}$ are replaced by lower step functions $(\tau_a^s)_{a \in A}$ and the corresponding bow graph G^B is constructed. Then, an appropriately modified version of the $(2 + \varepsilon)$ -approximation algorithm presented in [13] is applied yielding a weakly inflow-preserving flow over time in G^B . Finally, the output is turned into a feasible solution to the original problem.

3.3.1 Quickest Weakly Inflow-Preserving Flows

Fleischer and Skutella [13] propose a $(2 + \varepsilon)$ -approximation algorithm for the quickest multi-commodity flow problem with bounded cost and constant

transit times. The method is based on an approximate length-bounded static flow computation. The same approach can be applied to the problem of finding a quickest weakly inflow-preserving multi-commodity flow over time with bounded cost in the bow graph. Here, the goal is to determine a weakly inflow-preserving multi-commodity flow over time in G^B which satisfies all demands within minimum time T at a cost which is bounded by C .

Let f^B be an optimal solution to this problem with minimum time horizon T . As suggested in [13], we consider the static multi-commodity flow x^B in G^B which results from averaging the flow f^B on every arc $e \in A^B$ over the time interval $[0, T)$, i.e.,

$$x_{e,i}^B := \frac{1}{T} \int_0^T f_{e,i}^B(\theta) d\theta.$$

As proven in [13], this static flow

- (i) satisfies a fraction of $1/T$ of the demands covered by the flow over time f^B ,
- (ii) has cost $c(x^B) = c(f^B)/T$, and
- (iii) is T -length-bounded.

The latter property means that the flow of every commodity $i \in K$ can be decomposed into a sum of flows on s_i - t_i -paths such that the length $\tau(P) := \sum_{e \in P} \tau_e$ of any such path P is at most T . To see this, notice that, since the flow over time f^B finishes by time T , every (infinitesimal) unit of flow in f describes a path in G^B of length at most T . Taking all such paths yields a T -length-bounded path decomposition of x^B .

Since f^B is weakly inflow-preserving, so is x^B , i.e., its *per capacity flow values* $\lambda_e := x_e^B/u_e$, $e \in A^B$, satisfy

- (iv) $\sum_{e \in A_a^B} \lambda_e \leq 1$ for every arc $a \in A$.

An arbitrary static flow \tilde{x}^B in G^B meeting requirements (i) – (iv) can be turned into a weakly inflow-preserving flow over time g^B in G^B meeting the same demands at the same cost as f^B within time $2T$: Send flow into every s_i - t_i -path P given by the length-bounded path decomposition of x at the corresponding flow rate x_P for exactly T time units; wait for at most another T time units until all flow has arrived at its destination. Since $g_e^B(\theta)/u_e$ is always upper-bounded by x_e/u_e , it follows from property (iv) that g^B is weakly inflow-preserving.

Provided that we know the optimal time horizon of a quickest weakly inflow-preserving flow, we can compute a 2-approximate solution by solving the static flow problem defined by requirements (i) – (iv).

Unfortunately, computing T -length-bounded flows is \mathcal{NP} -hard. This follows from an easy reduction from the \mathcal{NP} -complete problem PARTITION. Yet, as discussed in [13], the T -length-bounded multi-commodity flow problem can be approximated within arbitrary precision in polynomial time by slightly relaxing the length bound T . It is easy to generalize this observation to length-bounded, weakly inflow-preserving flows. More precisely, given a time horizon T such that there exists a T -length-bounded static multi-commodity flow x in G^B which is weakly inflow-preserving, then, a $(1 + \varepsilon)T$ -length-bounded static flow x' which is weakly inflow-preserving, satisfies the same demands, and has cost $c(x') \leq c(x)$ can be computed in time polynomial in the input size and $1/\varepsilon$. Turning x' into a flow over time as described above thus yields a $(2 + \varepsilon)$ -approximate solution. In the following discussion, we detail how to compute x' . The discussion is a straightforward generalization of the arguments used in [13].

Let \mathcal{P}_i^T be the set of all s_i - t_i -paths in G^B whose transit times are bounded from above by T . Finding a static flow satisfying (i) – (iv) is equivalent to solving the following linear program:

The primal LP:

$$\begin{aligned}
\min \quad & \sum_{i \in K} \sum_{P \in \mathcal{P}_i^T} c_i(P) x_P \\
\text{s.t.} \quad & \sum_{P \in \mathcal{P}_i^T} x_P \geq d_i/T && \text{for all } i \in K, \\
& \sum_{e \in A_a^B} \frac{1}{u_e} \left(\sum_{i \in K} \sum_{\substack{P \in \mathcal{P}_i^T: \\ e \in P}} x_P \right) \leq 1 && \text{for all } a \in A, \\
& x_P \geq 0 && \text{for all } i \in K, P \in \mathcal{P}_i^T.
\end{aligned}$$

Notice that the flow variable x_P associated with path $P \in \mathcal{P}_i^T$ only contributes to the flow of commodity i , since we assume that no two commodities have the same source node. Since the number of paths in \mathcal{P}_i^T and thus the number of variables in this linear program can be exponentially large, we consider its dual.

The dual LP:

$$\begin{aligned}
\max \quad & \sum_{i \in K} (d_i/T) z_i - \sum_{a \in A} p_a \\
\text{s.t.} \quad & \sum_{e \in P} (p_{a(e)}/u_e + c_{e,i}) \geq z_i && \text{for all } i \in K, P \in \mathcal{P}_i^T, \\
& z_i, p_a \geq 0 && \text{for all } i \in K, e \in A^B, a \in A.
\end{aligned}$$

The separation problem of the dual can be formulated as $|K|$ length-bounded shortest path problems: for every $i \in K$, find a shortest s_i - t_i -path P with respect to the arc weights $y_e + p_{a(e)}/u_e + c_{e,i}$ whose length τ_P is at most T , i.e., $P \in \mathcal{P}_i^T$. The length-bounded shortest path problem is \mathcal{NP} -hard; see Garey and Johnson [23]. However, it can be solved approximately in the following sense; for any $\varepsilon > 0$, one can find in time polynomial in the size of the underlying graph and $1/\varepsilon$ an s_i - t_i -path P with $\tau_P \leq (1 + \varepsilon)T$ whose length with respect to the arc weights $y_e + p_{a(e)}/u_e + c_{e,i}$ is not larger than the length of a shortest path in \mathcal{P}_i^T [30, 46].

From the equivalence of optimization and separation [26] it follows that we can solve the dual linear program approximately: Assume we want to check the feasibility of the current solution. If it is not feasible, then there exists a path $P \in \mathcal{P}_i^T$ whose length with respect to the current arc weights $y_e + p_{a(e)}/u_e + c_{e,i}$ is smaller than the current value z_i . Using the approximation algorithm for the length-bounded shortest path problem, we can compute in polynomial time a path \tilde{P} whose transit time τ_P is bounded by $(1 + \varepsilon)T$ and whose length is smaller than z_i . In particular, we can separate the current solution by adding the constraint corresponding to \tilde{P} . That way, we might even separate feasible solutions since we also generate cuts corresponding to paths with transit time $T < \tau_P \leq (1 + \varepsilon)T$. Eventually, we optimally solve a modified dual linear program which arises from the dual program above by adding those constraints which we generated with our approximate separation routine and which correspond to some paths of length at most $(1 + \varepsilon)T$. From this dual solution we get a primal solution which uses extra variables corresponding to the generated paths of length at most $(1 + \varepsilon)T$. The next lemma summarizes the above.

Lemma 3.9. Given a time horizon T such that there exists a weakly inflow-preserving multi-commodity flow over time with time horizon T and cost at most C . Then, for every $\varepsilon > 0$, a weakly inflow-preserving multi-commodity flow over time with time horizon at most $(2 + \varepsilon)T$ and cost at most C can be computed in time polynomial in the input size and $1/\varepsilon$.

We point out that the above method relies on the ellipsoid method since the equivalence of optimization and separation [26] is based on the ellipsoid method. In particular, this method is only of limited relevance for solving practical problems. Alternatively, we suggest to apply the revised simplex method with column generation, which is known to be very efficient for solving practical problems. Notice, however, that it is not clear whether the revised simplex method terminates after a polynomial number of steps.

We give a brief description of this method; more details can be found, e.g., in [55]. The revised simplex method starts with a basic feasible solution to

the primal. In each iteration, the current basic solution is updated, i.e., a non-basic variable is selected which will enter the basis and a basic variable is selected that will leave the basis. In our case, the update of the basis essentially amounts to computing a solution to the length-bounded shortest path problem described above. Namely, the variable corresponding to a length-bounded shortest path is a candidate for entering the basis. Important is that we only generate a primal variable when it is about to enter the basis. The latter property is the main advantage of this method; throughout the entire procedure, we only keep a small number of variables (columns).

Lower bounds. If all transit time functions τ_a are constant, the $(2 + \varepsilon)$ -approximation algorithm in Lemma 3.9 and the one presented in [13] basically coincide. In [13], an example is given which shows that the performance guarantee of both algorithms is not better than 2, more precisely, for every $k \in \mathbb{N}$, a k -commodity problem is defined for which the algorithm has performance ratio exactly $(2k - 1)/k$.

The following instance shows that even in the single source, single sink case the approximation ratio of the discussed algorithm is not better than $4/3$. The example consists of a single arc $a = (s, t)$. The transit time of arc a is set to 0 if flow is entering at rate $x_a \leq 1$, and it is set to 1 if flow is entering at rate $x_a \in (1, 2]$. We want to send 2 units of flow from s to t as quickly as possible. A quickest weakly inflow-preserving flow finishes within $T = 3/2$ simply by sending flow at rate 2 during the interval $[0, 1/2]$, and at rate 1 during the interval $[1/2, 3/2]$. Note that this flow is even inflow-preserving.

A weakly inflow-preserving flow over time f^B which is generated from a path decomposition of a static flow as described above needs at least 2 time units. To see this, consider the corresponding bow graph G^B consisting of two parallel arcs e_1 and e_2 , where e_1 has transit time 0 and capacity 1, and e_2 has transit time 1 and capacity 2. Let λ_i be the per capacity flow rate of f^B on e_i . Then, for $T \geq 1$, the flow f^B manages to send $\lambda_1 T + 2\lambda_2(T - 1)$ flow units from s to t within time T . It is easily checked that f^B needs at least time $T = 2$ to satisfy the demand.

We point out that this lower bound can be further increased to $e^{1/e}$ (≈ 1.445) by discretizing the instance presented to prove Claim 2.39.

3.3.2 The Algorithm

So far, we have presented an algorithm to compute a $(2 + \varepsilon)$ -approximate solution to the quickest multi-commodity flow problem with bounded cost in the relaxed model of weakly inflow-preserving flows over time. Such a solution has a simple structure, namely it is generated from a path decomposition of

a static flow in the bow graph. We will use this property to turn such a flow into a solution to the original problem. Throughout this modification we will make sure that the time horizon only increases by a small factor.

Assume in the following that we have constructed the bow graph G^B according to step functions fulfilling the requirements stated in Observation 2.18. We will later specify the parameters $\delta, \eta > 0$ such that the size of the resulting bow graph is polynomial in the input size and $1/\varepsilon$.

Let f^B be a weakly inflow-preserving (multi-commodity) flow over time with time horizon T^B in G^B , which is generated from a static flow x^B as described in the last section. In particular, x^B is weakly inflow-preserving and has a length-bounded path decomposition. Let \mathcal{P}_i denote the set of s_i - t_i -paths from the length-bounded path decomposition of x^B and $\mathcal{P} := \cup_{i=1}^k \mathcal{P}_i$.

Lemma 3.10. The flow over time f^B can be turned into a flow over time f with inflow-dependent transit times $(\tau_a)_{a \in A}$ in G satisfying the same demands at the same cost as f^B within time T , where T is bounded from above by $(1 + \eta)T^B + 2n\delta$.

Proof. We increase transit times in G^B in order to emulate the original transit times $(\tau_a)_{a \in A}$. For every arc $e \in A^B$, let $\tilde{\tau}_e := (1 + \eta)\tau_e + \delta$ be the new transit time along e . Note that this corresponds to constructing the bow graph according to step functions $(\tilde{\tau}_a^s)_{a \in A}$, where $\tilde{\tau}_a^s(x) := (1 + \eta)\tau_a^s(x) + \delta$ for every $x \in [0, u_a]$. Consider a path $P \in \mathcal{P}$. The flow f^B sends flow at constant rate x_P into P for a certain time period. Before increasing transit times, flow traveling along P needed $\tau(P) := \sum_{e \in P} \tau_e$ time to reach its destination. After the increase, this time goes up to $\tilde{\tau}(P) := \sum_{e \in P} \tilde{\tau}_e \leq (1 + \eta)\tau(P) + n\delta$. Since $\tau(P)$ is bounded from above by T^B , the transit time of every unit of flow increases by at most $\eta T^B + n\delta$.

We repeat this procedure, but this time we increase the transit time of every arc $e \in A^B$ by another additive factor of δ . This way, we obtain a weakly inflow-preserving flow over time \hat{f}^B in the bow graph constructed with respect to transit times $(\tilde{\tau}_a^s)_{a \in A}$ which is δ -resting and whose time horizon is bounded by $(1 + \eta)T^B + 2n\delta$. Notice that throughout these modifications no flow is rerouted. We only make use of storage in nodes. Therefore, the cost of f^B remains unchanged, i.e., $c(f^B) = c(\hat{f}^B)$. Applying Corollary 3.8, this yields a flow over time f with inflow-dependent transit times $(\tilde{\tau}_a^s)_{a \in A}$ in G . Observation 1.19 implies that f can be interpreted as a flow over time with inflow-dependent transit times $(\tau_a)_{a \in A}$ in G which concludes the proof. \square

We are now ready to state the main result of this section.

Theorem 3.11. Assume that each transit time function $(\tau_a)_{a \in A}$ is a nonnegative, nondecreasing, left-continuous function given by an oracle as described

Algorithm 4: Approximating quickest inflow-dependent flow.

Input: Directed graph $G = (V, A)$ with positive capacities u_a , costs $c_{a,i}$, and nonnegative, nondecreasing, left-continuous transit time functions τ_a on the arcs, source-sink node pairs $(s_i, t_i) \in V \times V$, flow demands d_i , a budget C , parameter $\varepsilon > 0$.

Output: Multi-commodity flow over time f with inflow-dependent transit times in G satisfying demands d_i at cost C within time $(2 + \varepsilon)\bar{T}$.

- 1 Compute lower bound L on \bar{T} (see proof of Theorem 3.11), set $\eta := \varepsilon/8$ and $\delta := \varepsilon L/(12n)$;
 - 2 Replace transit time functions τ_a by step functions τ_a^s (see Observation 2.18);
 - 3 Construct bow graph G^B with respect to $(\tau_a^s)_{a \in A}$;
 - 4 Using geometric mean binary search, compute (temporally repeated) weakly inflow-preserving multi-commodity flow over time f^B with time horizon $T \leq (2 + \varepsilon/4)\bar{T}$ and cost $c(f^B) \leq C$ in G^B (see Lemma 3.9);
 - 5 Turn f^B into δ -resting flow by increasing the time horizon to at most $(2 + \varepsilon)\bar{T}$ (see Lemma 3.10);
 - 6 Turn f^B into flow over time f with inflow-dependent transit times in G (see Corollary 3.8);
-

in Remark 1.18. Then there exists a polynomial time algorithm for the quickest multi-commodity flow problem with inflow-dependent transit times and bounded cost that, for any $\varepsilon > 0$, finds a solution of the same cost as optimal with time horizon at most $2 + \varepsilon$ times the optimal time horizon \bar{T} .

Proof. The algorithm is summarized above. We can compute in polynomial time a lower bound L on \bar{T} such that $L \leq \bar{T} \leq p(n)kL$, for some polynomial p . Namely, it is stated in Lemma 2.25 that for every commodity i , a lower bound L_i on the optimal time horizon T_i for sending commodity i such that $L_i \leq T_i \leq p(n)L_i$ can be computed in polynomial time. Setting $L := \max_i L_i$ yields the desired bound. Notice that Lemma 2.25 assumes that every transit time function τ_a is given by an oracle as specified in Remark 1.18.

We fix η to $\varepsilon/8$ and δ to $\varepsilon L/(12n)$. For every arc $a \in A$, we pick lower step functions according to Observation 2.18. As already observed in [43], the number of breakpoints of τ_a^s is then in $\mathcal{O}(\log(nk/\varepsilon)/\varepsilon)$ and thus polynomially bounded. The latter is a direct consequence of Observation 2.18 (ii) and the fact that without loss of generality we can set the capacity of every

arc $a \in A$ to $u'_a := \max\{x \in [0, u_a] \mid \tau_a(x) \leq p(n)kL\}$. We then construct the bow graph G^B with respect to these step functions. Because of the relaxation property of G^B (see Observation 3.5), the time horizon \overline{T}^B of a quickest weakly inflow-preserving flow in G^B is a lower bound on \overline{T} . If $\overline{T}^B \leq L$, then using Lemma 3.9 with $T = L$ we can compute a weakly inflow-preserving multi-commodity flow over time with time horizon at most $(2 + \varepsilon/4)L \leq (2 + \varepsilon/4)\overline{T}$. Otherwise $L \leq \overline{T}^B \leq p(n)kL$ holds. Using geometric mean binary search together with Lemma 3.9, we can compute a weakly inflow-preserving multi-commodity flow over time with time horizon T such that $T \leq (2 + \varepsilon/4)\overline{T}^B \leq (2 + \varepsilon/4)\overline{T}$.

Applying Lemma 3.10, this flow over time can be turned into a flow over time f with inflow-dependent transit times in G . Its time horizon is bounded by $(1 + \eta)(2 + \varepsilon/4)\overline{T} + 2n\delta = (1 + \varepsilon/8)(2 + \varepsilon/4)\overline{T} + \varepsilon/6L \leq (2 + \varepsilon)\overline{T}$. Recall that f_i is given by piecewise constant functions $(f_{a,i})_{a \in A}$. Corollary 3.8 implies that the number of breakpoints of each such function is indeed polynomial in the input size and $1/\varepsilon$. \square

3.4 FPTAS

In this section we present a fully polynomial time approximation scheme (FPTAS) for the quickest multi-commodity flow problem with inflow-dependent transit times and bounded cost. We use ideas similar to the ones employed in [14] for the problem with fixed transit times. The FPTAS is based on a static weakly inflow-preserving flow computation in a condensed time-expanded bow graph.

3.4.1 Condensed Time-Expanded Graphs

It is shown in Section 1.3.3 that flow over time problems with constant transit times translate to static flow problems in a time-expanded graph. Hence, many flow over time problems can be solved by applying algorithmic techniques developed for static network flows in the time-expanded graph. However, one has to pay for this simplification of the considered flow problem in terms of an enormous increase in the size of the underlying network. In particular, the size of the time-expanded graph is only pseudo-polynomial in the input size. Nevertheless, Fleischer and Skutella [13] show that the time-expanded network can be condensed to polynomial size while keeping control over the optimal time horizon of a flow solution. In the following, we present their idea.

Assume that all transit times are multiples of some large number $\Delta > 0$.

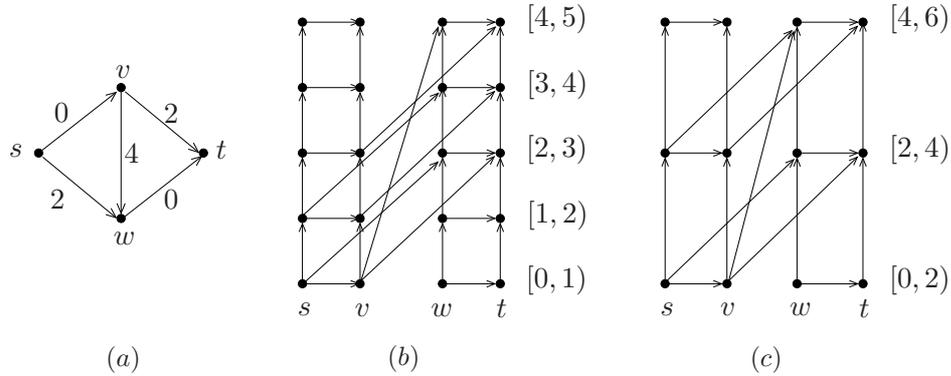


Figure 3.3: Figure (b) displays the T -time-expansion of the instance shown in (a) for time horizon $T = 5$. Figure (c) depicts the Δ -condensed time-expanded network for $\Delta = 2$.

Instead of using the T -time-expanded graph, one can rescale time and use a Δ -condensed time-expanded graph $G(T)/\Delta$ that contains only $\lceil T/\Delta \rceil$ copies of V , denoted $V(\rho\Delta)$, $\rho = 0, \dots, \lceil T/\Delta \rceil - 1$. For every arc $a = (v, w) \in A$ and every point in time $\theta \in \{\rho\Delta \mid \rho = 0, \dots, (\lceil T/\Delta \rceil - 1) - \tau_a/\Delta\}$, there is an arc $a(\theta)$ from $v(\theta)$ to $w(\theta + \tau_a)$. Since in this setting every arc corresponds to a time interval of length Δ , capacities are multiplied by Δ . In addition, there is an infinite capacity *holdover arc* from $v(\theta)$ to $v(\theta + \Delta)$, for all $v \in V$ and $\theta \in \{\rho\Delta \mid \rho = 0, \dots, \lceil T/\Delta \rceil - 2\}$, which models the possibility to hold flow at node v . The cost of every arc in the time-expanded graph is equal to the cost of the original arc. An example is shown in Figure 3.3.

Any static flow in this condensed time-expanded graph corresponds to a flow over time of equal cost in the original graph with time horizon bounded by $T + \Delta$: divide the static flow on an arc with tail in $V_{\rho\Delta}$ by Δ and send flow at this rate for Δ units of time starting at time $\rho\Delta$. Conversely, given a flow over time with time horizon T , a static flow in the condensed time-expanded graph of equal cost can be obtained by mapping the whole flow on arc $a \in A$ during time interval $[\rho\Delta, (\rho + 1)\Delta)$, $\rho = 0, \dots, \lceil T/\Delta \rceil - 1$, to the copy $a(\rho\Delta)$. Capacity constraints are fulfilled since the total flow starting on arc a in interval $[\rho\Delta, (\rho + 1)\Delta)$, $\rho = 0, \dots, \lceil T/\Delta \rceil - 1$, is bounded by Δu_a . The following lemma taken from [13] summarizes the above.

Lemma 3.12. Suppose that all transit times are multiples of Δ . Then, any flow over time that completes by time T corresponds to a static flow of equal value and cost in $G(T)/\Delta$, and any static flow in $G(T)/\Delta$ corresponds to a flow over time of equal value and cost that completes by time $T + \Delta$.

3.4.2 The Algorithm

In this section we describe a fully polynomial time approximation scheme for the quickest multi-commodity flow problem. The algorithm works on an appropriately condensed time-expanded graph. To state the algorithm and prove its correctness, we first need to define the underlying bow graphs.

The Underlying Graphs. Assume we are given a graph $G = (V, A)$ with transit time functions $(\tau_a)_{a \in A}$ on the arcs, a time horizon T , and $\Delta > 0$. We define the following three bow graphs: The first bow graph $G^\downarrow = (V^\downarrow, A^\downarrow)$ is constructed with respect to lower step functions $(\tau_a^\downarrow)_{a \in A}$, which arise from the original transit time functions $(\tau_a)_{a \in A}$ by rounding down to the nearest multiples of Δ , i.e., $\tau_a^\downarrow(x) := \lfloor \tau_a(x)/\Delta \rfloor \Delta$, for $a \in A$, $x \in [0, u_a]$. We denote the transit time of an arc $e \in A^\downarrow$ by τ_e^\downarrow . The second bow graph $G^\uparrow = (V^\uparrow, A^\uparrow)$ denotes the Δ -lengthened bow graph $G^\uparrow = (V^\uparrow, A^\uparrow)$ which is constructed from G^\downarrow by increasing the transit time function of each arc by Δ . The corresponding transit time step functions are given by $\tau_a^\uparrow(x) := \tau_a^\downarrow(x) + \Delta$, for $a \in A$, $x \in [0, u_a]$. We denote the transit time of an arc $e \in A^\uparrow$ by τ_e^\uparrow . Thirdly, let $G^{\uparrow\uparrow} = (V^{\uparrow\uparrow}, A^{\uparrow\uparrow})$ be the 2Δ -lengthened bow graph, which arises from G^\downarrow by lengthening the transit time of each arc by 2Δ . The corresponding transit time step functions are given by $\tau_a^{\uparrow\uparrow}(x) := \tau_a^\downarrow(x) + 2\Delta$, for $a \in A$, $x \in [0, u_a]$. We denote the transit time of an arc $e \in A^{\uparrow\uparrow}$ by $\tau_e^{\uparrow\uparrow}$.

Let $G^F = (V^F, A^F)$ be the Δ -condensed time-expansion of $G^{\uparrow\uparrow}$ with time horizon T , i.e., $G^F := G^{\uparrow\uparrow}(T)/\Delta$. The exact choice of T is given in the paragraph below. We refer to G^F also as the Δ -condensed fan graph; compare Section 2.3.4. Each arc $a = (v, w) \in A$ is represented in the bow graph $G^{\uparrow\uparrow}$ by its expansion $A_a^{\uparrow\uparrow}$. Thus, the fan graph contains, for each point in time $\theta \in \{\rho\Delta \mid \rho = 0, \dots, \lceil T/\Delta \rceil - 1\}$, a “fan” of arcs $A_a^F(\theta) := \{e(\theta) : e \in A_a^{\uparrow\uparrow}, \theta + \tau_e \leq (\lceil T/\Delta \rceil - 1)\Delta\}$, where $e(\theta) := (v(\theta), w(\theta + \tau_e))$.

For a static flow x in G^F , we define $\lambda_e := x_e/u_e$ to be the *per capacity inflow value* on arc $e \in A^F$. With these definitions, the concept of (weakly) inflow-preserving flows directly carries over to static flows in G^F : The static flow x is *weakly inflow-preserving* if

$$\sum_{e \in A_a^F(\theta)} \lambda_e \leq 1, \quad (3.2)$$

for all $a \in A$ and $\theta \in \{\rho\Delta \mid \rho = 0, \dots, \lceil T/\Delta \rceil - 1\}$.

Specifying T and Δ . Let \bar{T} denote the time horizon of a quickest multi-commodity flow with inflow-dependent transit times and bounded cost in G and let L be a lower bound on \bar{T} with $L \in \mathcal{O}(\bar{T})$. Given $\varepsilon > 0$, we set

$\varepsilon' := \varepsilon/8$. Without loss of generality, we can assume in the following calculations that $\varepsilon < 1$ and therefore $\varepsilon' < 1/8$. Let T' be an estimate of \bar{T} with $\bar{T} \leq T' \leq (1 + \varepsilon')\bar{T}$. We set $\Delta := \varepsilon'^2 L/n$ and $T := (1 + \varepsilon/2)T'$.

With this choice of parameters, we can prove the following lemma.

Lemma 3.13. A weakly inflow-preserving static multi-commodity flow x^F in G^F can be turned into a multi-commodity flow over time f with inflow-dependent transit times $(\tau_a)_{a \in A}$ in G satisfying the same demands as x^F at the same cost as x^F within time horizon bounded by $(1 + \varepsilon)\bar{T}$.

Proof. It follows from Lemma 3.12 that x^F corresponds to a flow over time f' in $G^{\uparrow\uparrow}$ satisfying the same demands at the same cost as x^F with time horizon bounded by $T + \Delta$. By choice of T , we have that $T = (1 + \varepsilon/2)T' \leq (1 + \varepsilon/2)(1 + \varepsilon/8)\bar{T} \leq (1 + 3/4\varepsilon)\bar{T}$. By choice of Δ , we have that $\Delta = \varepsilon'^2 L/n \leq 1/8\varepsilon\bar{T}$. We conclude that the time horizon of f' can be upper-bounded by $T + \Delta \leq (1 + \varepsilon)\bar{T}$. Since x^F is weakly inflow-preserving, so is f' . By shortening all arcs of $G^{\uparrow\uparrow}$ by Δ , the flow over time f' can be interpreted as a weakly inflow-preserving flow over time in G^{\uparrow} which is Δ -resting. Applying Corollary 3.8, we derive a flow over time f in G with inflow-dependent transit times $(\tau_a^{\uparrow})_{a \in A}$ and time horizon at most $(1 + \varepsilon)\bar{T}$. Since $\tau_a^{\uparrow}(x) \geq \tau_a(x)$, for all $a \in A$, $x \in [0, u_a]$, the statement of the lemma follows from Observation 1.19. \square

From this lemma we can immediately derive an algorithm. Using binary search, we find the estimate T of \bar{T} . In each search step, we check whether $G^F = G^{\uparrow\uparrow}(T)/\Delta$ contains a weakly inflow-preserving multi-commodity flow satisfying all demands at given cost and update the value of T . The crucial part is to show that, for this choice of T , such a flow exists in G^F . The algorithm is summarized on the facing page.

We now show that the algorithm can be implemented to run in polynomial time. The correctness of the algorithm will be shown subsequently. The lower bound L in Step 1 can be computed in polynomial time using the constant factor approximation algorithm presented in Section 3.3. The estimate T' can thus be found within $\mathcal{O}(\log(1/\varepsilon))$ geometric mean binary search steps.

In each search step, we need to compute a weakly inflow-preserving multi-commodity flow in a condensed fan graph G^F . This problem can be formulated as a linear program by adding a bundle constraint of type (3.2) for each fan in G^F to a standard network flow formulation. It remains to bound the size of G^F . By definition, the fan graph contains $\lceil T/\Delta \rceil \in \mathcal{O}(n/\varepsilon^2)$ time layers. Hence, the number of nodes lies in $\mathcal{O}(n^2/\varepsilon^2)$. Every arc $a \in A$ is represented in each time layer by a fan. Since a fan in G^F contains at most $\lceil T/\Delta \rceil \in \mathcal{O}(n/\varepsilon^2)$ arcs—one for each time layer—the number of arcs in G^F

Algorithm 5: FPTAS for quickest inflow-dependent flow.

Input: Directed graph $G = (V, A)$ with positive capacities u_a , costs c_a , and nonnegative, nondecreasing, left-continuous transit time functions τ_a on the arcs, source-sink node pairs $(s_i, t_i) \in V \times V$, flow demands d_i , a budget C , $\varepsilon > 0$.

Output: Multi-commodity flow over time f with inflow-dependent transit times in G satisfying demands d_i at cost C within time $(1 + \varepsilon)\bar{T}$.

- 1 Compute lower bound L and upper bound U on \bar{T} with $L \in \mathcal{O}(U)$ (see Theorem 3.11), set $\varepsilon' := \varepsilon/8$ and $\Delta := \varepsilon'^2 L/n$;
 - 2 Using geometric mean binary search on $[L, U]$, determine T' with $\bar{T} \leq T' \leq (1 + \varepsilon')\bar{T}$ such that, for $T := (1 + \varepsilon/2)T'$, $G^F = G^{\uparrow\uparrow}(T)/\Delta$ contains a weakly inflow-preserving static multi-commodity flow x' satisfying all demands at cost C ;
 - 3 Interpret x' as flow over time f' in $G^{\uparrow\uparrow}$;
 - 4 Interpret f' as a Δ -resting flow over time in G^\uparrow ;
 - 5 Turn f' into a flow over time f with inflow-dependent transit times $(\tau_a)_{a \in A}$ in G (see Corollary 3.8);
-

lies in $\mathcal{O}(mn^2/\varepsilon^4)$. We conclude that G^F is of polynomial size. Consequently, a weakly inflow-preserving multi-commodity flow in G^F can be computed in polynomial time, for instance, by applying the ellipsoid method; for more details on this method see for example [26].

The running time of Step 3 and 4 is obviously polynomial. It follows from Corollary 3.8 that the algorithm outputs a solution which is given by piecewise constant functions $f_{a,i}$, $a \in A, i \in K$, where the number of breakpoints of $f_{a,i}$ is bounded by $2|A_a^\uparrow| \lceil T/\Delta \rceil$. Since we can assume that A_a^\uparrow only contains arcs with transit time not larger than T , we can bound $|A_a^\uparrow|$ by T/Δ . Hence, the number of breakpoints of $f_{a,i}$ lies in $\mathcal{O}(n^2/\varepsilon^4)$.

It remains to show that the algorithm performs correctly, i.e., we need to discuss the choice of T in the algorithm and prove that a static multi-commodity flow as described in Step 2 exists. To this end, we transform a quickest flow in G with inflow-dependent transit times into a weakly inflow-preserving static multi-commodity flow in G^F . The transformation is done in several steps which are illustrated in Figure 3.4: Step ① in the transformation immediately follows from Observations 1.19 and 3.5; a flow over time with inflow-dependent transit times $(\tau_a)_{a \in A}$ in G naturally induces a flow over time with inflow-dependent transit times $(\tau_a^\downarrow)_{a \in A}$ in G which, in turn, corresponds to an inflow-preserving flow over time in G^\downarrow . Likewise, transformation ③

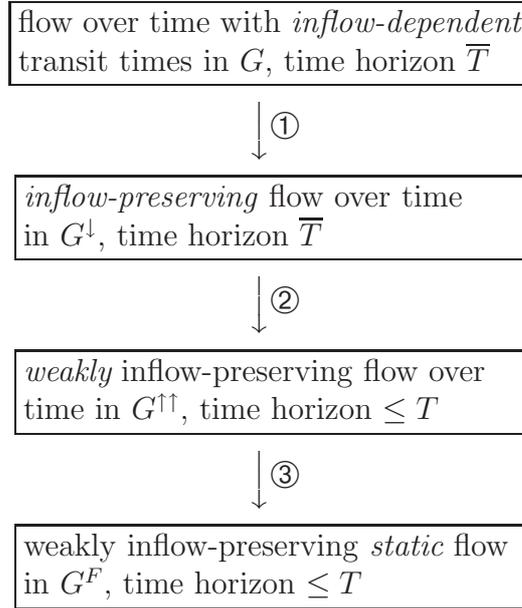


Figure 3.4: Diagram depicting the transformation steps needed to prove Lemma 3.14; a quickest flow with inflow-dependent transit times is turned into a weakly inflow-preserving static flow in G^F .

was already used to prove Lemma 3.12; simply map the total amount of flow entering arc $e \in A^{\uparrow\uparrow}$ in the interval $[\theta, \theta + \Delta)$ to $e(\theta) \in A^F$, for $\theta \in \{\rho\Delta \mid \rho = 0, \dots, \lceil T/\Delta \rceil - 1\}$. It can easily be checked that these transformations preserve cost and the property of being weakly inflow-preserving.

Step ② is the most interesting but also the most intricate one. Increasing transit times in the bow graph will in general destroy feasibility of the considered flow over time. Similarly to [14], we will carefully average flow to derive an “almost feasible” flow, then subsequently reduce the flow value to obtain feasibility, and finally increase the time horizon to meet demands. We can adopt this method since the transit times in bow graphs G^\downarrow and $G^{\uparrow\uparrow}$ are constant. However, in contrast to [14], our flows must have the additional property of being weakly inflow-preserving. Hence, in the following more detailed discussion of transformation ②, we will stress why this property is preserved.

Lemma 3.14. A weakly inflow-preserving multi-commodity flow over time f in G^\downarrow with time horizon \bar{T} can be transformed into a weakly inflow-preserving flow over time in $G^{\uparrow\uparrow}$ with time horizon at most T satisfying the same demands as f at the same cost as f .

On the way to proving the lemma we need to show several intermediate results. Consider a flow over time f in G^\downarrow as claimed in the lemma. Fleischer and Skutella [14] introduce a technique for “smoothing” f . The intuition is the following. Consider two paths P_1 and P_2 in G^\downarrow sharing an arc e in G^\downarrow . Assume that f is sending flow into P_i , $i \in \{1, 2\}$, at a very high rate for a short period of time. Figuratively, both paths carry a big “package” of flow. We assume that these packages travel through arc e one after another. Now we increase the transit time of every arc in G^\downarrow by an equal amount. That way, we also increase the transit times of paths P_1 and P_2 . After this modification, the two packages might collide when traveling through e . This happens if the number of arcs preceding e in P_1 is not equal to the number of arcs preceding e in P_2 . In that case, the package traveling along P_1 experiences a different delay than the package traveling along P_2 . To avoid such collisions, we preprocess each path flow. That is, we smooth each path flow in order to avoid extreme changes in the flow rate.

Smoothing f . In f , every infinitesimal unit of flow describes a simple² path P in G^\downarrow and a certain delay configuration; if P is given by nodes (v_0, v_1, \dots, v_q) , then a vector of nonnegative delays $\delta = (\delta_1, \dots, \delta_{q-1})$ specifies the amount of time for which the infinitesimal flow unit is stored in each node. More precisely, the flow unit rests in node v_j for δ_j units of time before it continues towards node v_{j+1} . We call a path P together with a delay vector δ a *path with delay* and denote it by P^δ .

Since all arc lengths in G^\downarrow are multiples of Δ , we can assume that each δ_i , $i = 1, \dots, q-1$, is a multiple of Δ : First, we apply Lemma 3.12 to turn f into a static flow in the Δ -condensed time expansion of G^\downarrow . Then, we again apply Lemma 3.12 to turn the static flow back into a flow over time in G^\downarrow . After this transformation, f is still weakly inflow-preserving and has the same cost.

Since all hold-over arcs in the Δ -condensed time-expansion connect time layers of distance Δ , the statement follows. Notice that we have to pay for that transformation with an increase of the time horizon by an additive factor of Δ . Hence, the following statement holds after the transformation.

Observation 3.15. The time horizon of the flow over time f in G^\downarrow is upper-bounded from above by $\bar{T} + \Delta$.

In particular, the flow f induces only finitely many paths with delay P^δ . Therefore, there exists a (finite) decomposition of the flow over time f into flows over time f_{P^δ} on paths with delay P^δ . Consider a path P^δ of the path

²Notice that cycles can be avoided by storing flow at intermediate nodes.

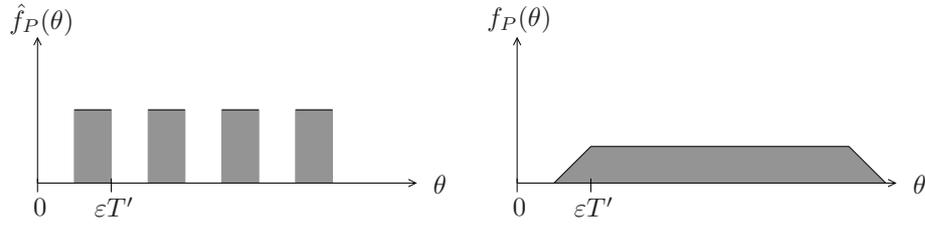


Figure 3.5: The original path flow over time f_P and the smoothed path flow over time \hat{f}_P .

decomposition of f , i.e., P^δ is given by a path $P = (v_0, v_1, \dots, v_q)$ in G^\downarrow and a delay vector δ . For $e := (v_l, v_{l+1})$, we define

$$\tau^\downarrow(P^\delta, e) := \sum_{j=1}^{\ell} (\tau_{(v_{j-1}, v_j)}^\downarrow + \delta_j) \quad (3.3)$$

to be the *transit time with delay* of the subpath (v_0, v_1, \dots, v_l) in G^\downarrow . Similarly, let $\tau^{\uparrow\uparrow}(P^\delta, e)$ denote the transit time with delay of this subpath with respect to the increased transit times in $G^{\uparrow\uparrow}$. We denote the transit time with delay of the entire path P^δ by $\tau^\downarrow(P^\delta)$ and $\tau^{\uparrow\uparrow}(P^\delta)$, respectively.

In particular, flow in f entering P^δ at time θ enters arc e at time $\theta + \tau^\downarrow(P^\delta, e)$. Therefore, the total flow entering arc e at time θ can be computed as follows:

$$f_e(\theta) = \sum_{P^\delta: e \in P} f_{P^\delta}(\theta - \tau^\downarrow(P^\delta, e)). \quad (3.4)$$

We average the flow along each path P^δ and, thereby, define a new flow \hat{f} . For every $\theta \in [0, \bar{T} + \Delta + \varepsilon' T']$, we set

$$\hat{f}_{P^\delta}(\theta) := \frac{1}{\varepsilon T'} \int_{\theta - \varepsilon' T'}^{\theta} f_{P^\delta}(\xi) d\xi. \quad (3.5)$$

As in [14], we call \hat{f} *smoothed*. Notice that, if ε is small, then \hat{f}_{P^δ} approximates the original path flow over time f_{P^δ} very accurately. An example of a smoothed flow rate function is depicted in Figure 3.5.

Since \hat{f} is defined via a path decomposition, we can interpret \hat{f} as a flow over time in the 2Δ -lengthened bow graph $G^{\uparrow\uparrow}$. In particular, flow in \hat{f} entering P^δ at time θ now enters arc e at time $\theta + \tau^{\uparrow\uparrow}(P^\delta, e)$. Thus, the total flow entering arc e at time θ can be computed as follows:

$$\hat{f}_e(\theta) = \sum_{P^\delta: e \in P} \hat{f}_{P^\delta}(\theta - \tau^{\uparrow\uparrow}(P^\delta, e)). \quad (3.6)$$

The smoothed flow satisfies all demands and has the same cost as f because the total amount of flow sent along every path with delay P^δ is not changed. Moreover, it satisfies the flow conservation constraints since it is defined through flows on paths.

However, the flow over time \hat{f} need not be feasible in $G^{\uparrow\uparrow}$; in fact, capacity constraints and the weakly inflow-preserving property might be violated. Before we address this problem, we derive an upper bound on the time horizon \hat{T} of \hat{f} in $G^{\uparrow\uparrow}$.

Claim 3.16. The time horizon \hat{T} of the flow over time \hat{f} in $G^{\uparrow\uparrow}$ is bounded from above by $(1 + 3/2\varepsilon')T'$.

Proof. The time horizon of f in G^\downarrow is bounded by $\bar{T} + \Delta$; see Observation 3.15. Smoothing f increases the time horizon by another additive factor of $\varepsilon' T'$: simply note that flow in \hat{f}_{P^δ} is entering path P^δ at most $\varepsilon' T'$ time units later than in f_{P^δ} ; see (3.5). Therefore, we can upper-bound the time horizon of \hat{f} in G^\downarrow by $\bar{T} + \Delta + \varepsilon' T' \leq (1 + 5/4\varepsilon')T'$, where the latter inequality holds because $\bar{T} \leq T'$ and $\Delta = \varepsilon'^2 L/n \leq \varepsilon'/4 T'$.

Next, we interpret \hat{f} as a flow over time in the 2Δ -lengthened bow graph $G^{\uparrow\uparrow}$. Consider a path P^δ of the path decomposition of f , i.e., P^δ is given by a path $P = (v_0, v_1, \dots, v_q)$ in G^\downarrow and a delay vector δ . Since every path P^δ is simple, it contains at most $n - 1$ arcs. Therefore, we can bound the transit time with delay $\tau^{\uparrow\uparrow}(P^\delta)$ of path P^δ in $G^{\uparrow\uparrow}$ in terms of the transit time with delay $\tau^\downarrow(P^\delta)$ in G^\downarrow :

$$\begin{aligned} \tau^\downarrow(P^\delta) &\leq \tau^{\uparrow\uparrow}(P^\delta) \\ &\leq \tau^\downarrow(P^\delta) + 2(n - 1)\Delta \\ &\leq \tau^\downarrow(P^\delta) + 2\varepsilon'^2 T', \end{aligned} \tag{3.7}$$

where the last inequality holds because $\Delta = \varepsilon'^2 L/n$. Consequently, we can bound the time horizon of \hat{f} in $G^{\uparrow\uparrow}$ by $(1 + 5/4\varepsilon')T' + 2\varepsilon'^2 L \leq (1 + 5/4\varepsilon')T' + \varepsilon'/4 L \leq (1 + 3/2\varepsilon')T'$. This concludes the proof of the claim. \square

Later we need the following claim. It follows immediately from the calculations in (3.7) by replacing P^δ by a subpath of P^δ .

Claim 3.17. For every arc $e \in P^\delta$, the following holds:

$$\tau^\downarrow(P^\delta, e) \leq \tau^{\uparrow\uparrow}(P^\delta, e) \leq \tau^\downarrow(P^\delta, e) + 2\varepsilon'^2 T'. \tag{3.8}$$

Rescaling \hat{f} . We have mentioned already above that the flow over time \hat{f} need neither be feasible nor weakly inflow-preserving. However, it is proven

in [14] that the flow over time \hat{f} is almost feasible, i.e., for every arc $e \in A^{\uparrow\uparrow}$ and every $\theta \in [0, \hat{T})$, the flow rate $\hat{f}_e(\theta)$ is not larger than $(1 + 2\varepsilon')u_e$. In a similar fashion, we will show next that the flow over time \hat{f} is almost weakly inflow-preserving. The following rather technical claims bound the sum of the per capacity inflow rates $\hat{\lambda}_e(\theta)$ of \hat{f} in each expansion $A_a^{\uparrow\uparrow}$, $a \in A$. First, we bound $\hat{f}_e(\theta)$ in terms of f .

Claim 3.18. For every $e \in A^{\uparrow\uparrow}$ and $\theta \in [0, \hat{T})$,

$$\hat{f}_e(\theta) \leq \frac{1}{\varepsilon' T'} \sum_{P^\delta: e \in P} \int_{\theta - 2\varepsilon'^2 T' - \varepsilon' T'}^{\theta} f_{P^\delta}(\xi - \tau^\downarrow(P^\delta, e)) \, d\xi.$$

Proof.

$$\begin{aligned} \hat{f}_e(\theta) &= \sum_{P^\delta: e \in P} \hat{f}_{P^\delta}(\theta - \tau^{\uparrow\uparrow}(P^\delta, e)) && \text{by (3.6),} \\ &= \frac{1}{\varepsilon' T'} \sum_{P^\delta: e \in P} \int_{\theta - \tau^{\uparrow\uparrow}(P^\delta, e) - \varepsilon' T'}^{\theta - \tau^{\uparrow\uparrow}(P^\delta, e)} f_{P^\delta}(\xi) \, d\xi && \text{by (3.5),} \\ &\leq \frac{1}{\varepsilon' T'} \sum_{P^\delta: e \in P} \int_{\theta - \tau^\downarrow(P^\delta, e) - 2\varepsilon'^2 T' - \varepsilon' T'}^{\theta - \tau^\downarrow(P^\delta, e)} f_{P^\delta}(\xi) \, d\xi && \text{by (3.8),} \\ &= \frac{1}{\varepsilon' T'} \sum_{P^\delta: e \in P} \int_{\theta - 2\varepsilon'^2 T' - \varepsilon' T'}^{\theta} f_{P^\delta}(\xi - \tau^\downarrow(P^\delta, e)) \, d\xi. \end{aligned}$$

□

Using the above claim, we prove that \hat{f} is almost weakly inflow-preserving in $G^{\uparrow\uparrow}$.

Claim 3.19. For every $a \in A$ and $\theta \in [0, \hat{T})$,

$$\sum_{e \in A_a^{\uparrow\uparrow}} \hat{\lambda}_e(\theta) \leq 1 + 2\varepsilon'.$$

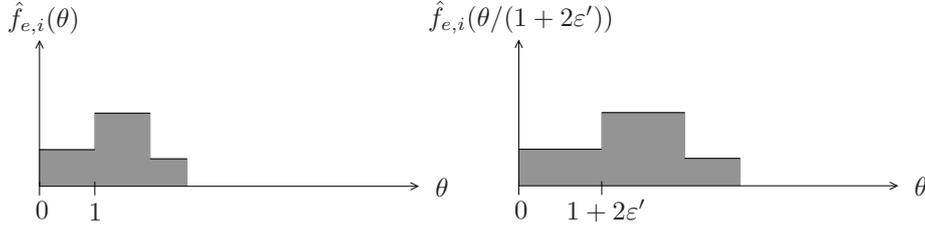


Figure 3.6: The original flow rate $\hat{f}_{e,i}(\theta)$ and the stretched flow rate $\hat{f}_{e,i}(\theta/(1+2\epsilon'))$.

Proof.

$$\begin{aligned}
\sum_{e \in A_a^{\uparrow\uparrow}} \hat{\lambda}_e(\theta) &= \sum_{e \in A_a^{\uparrow\uparrow}} \hat{f}_e(\theta)/u_e \\
&\leq \sum_{e \in A_a^{\uparrow\uparrow}} \left(\frac{1}{\epsilon' T'} \sum_{P^\delta: e \in P} \int_{\theta - 2\epsilon'^2 T' - \epsilon' T'}^{\theta} f_{P^\delta}(\xi - \tau^\downarrow(P^\delta, e)) d\xi \right) / u_e \\
&= \frac{1}{\epsilon' T'} \int_{\theta - 2\epsilon'^2 T' - \epsilon' T'}^{\theta} \left(\sum_{e \in A_a^{\uparrow\uparrow}} \frac{1}{u_e} \sum_{P^\delta: e \in P} f_{P^\delta}(\xi - \tau^\downarrow(P^\delta, e)) \right) d\xi \\
&= \frac{1}{\epsilon' T'} \int_{\theta - 2\epsilon'^2 T' - \epsilon' T'}^{\theta} \sum_{e \in A_e^{\uparrow\uparrow}} f_e(\xi)/u_e d\xi \quad \text{by (3.4),} \\
&\leq \frac{1}{\epsilon' T'} \int_{\theta - 2\epsilon'^2 T' - \epsilon' T'}^{\theta} 1 d\xi \\
&= \frac{2\epsilon'^2 T' + \epsilon' T'}{\epsilon' T'} = 1 + 2\epsilon',
\end{aligned}$$

where the first inequality follows from Claim 3.18 and where the last inequality holds because f is weakly inflow-preserving. This concludes the proof of the claim. \square

Dividing \hat{f}_e by $(1+2\epsilon')$ on every arc $e \in A^{\uparrow\uparrow}$ yields a feasible weakly inflow-preserving flow over time in $G^{\uparrow\uparrow}$ that satisfies a fraction of $1/(1+2\epsilon')$ of all demands and has cost at most $c(\hat{f})/(1+2\epsilon') = c(f)/(1+2\epsilon') \leq C/(1+2\epsilon')$. A subsequent “stretching” of \hat{f} ensures that all demands are met.

Stretching \hat{f} . We expand the time horizon \hat{T} by a multiplicative factor of $(1+2\epsilon')$ and increase the transit time $\tau_e^{\uparrow\uparrow}$ of arc e to $\tilde{\tau}_e := (1+2\epsilon')\tau_e^{\uparrow\uparrow}$, for every arc $e \in A^{\uparrow\uparrow}$. Then, we reparameterize \hat{f} accordingly by considering $\hat{f}_{e,i}(\theta/(1+2\epsilon'))$, $e \in A^{\uparrow\uparrow}$, $\theta \in [0, (1+2\epsilon')\hat{T}]$ instead. Figure 3.6 depicts an example of a stretched flow rate function. The new flow satisfies flow conservation

constraints:

$$\begin{aligned}
& \sum_{e \in \delta^+(v)} \int_0^\xi \hat{f}_{e,i}(\theta/(1+2\varepsilon')) d\theta - \sum_{e \in \delta^-(v)} \int_{\tilde{\tau}_e}^\xi \hat{f}_{e,i}(\theta/(1+2\varepsilon') - \tilde{\tau}_e) d\theta \\
&= (1+2\varepsilon') \sum_{e \in \delta^+(v)} \int_0^{\frac{\xi}{1+2\varepsilon'}} \hat{f}_{e,i}(\theta) d\theta - (1+2\varepsilon') \sum_{e \in \delta^-(v)} \int_0^{\frac{\xi - \tilde{\tau}_e}{1+2\varepsilon'}} \hat{f}_{e,i}(\theta) d\theta \\
&= (1+2\varepsilon') \left(\sum_{e \in \delta^+(v)} \int_0^{\frac{\xi}{1+2\varepsilon'}} \hat{f}_{e,i}(\theta) d\theta - \sum_{e \in \delta^-(v)} \int_0^{\frac{\xi}{1+2\varepsilon'}} \hat{f}_{e,i}(\theta - \tau_e^{\uparrow}) d\theta \right) \\
&\leq 0,
\end{aligned}$$

for all $v \in V^{\uparrow\uparrow} \setminus \{s_i\}$, $i \in K$, where the last inequality follows because \hat{f} satisfies flow conservation constraints. Similarly, it can be checked that the value of the new flow with respect to commodity $i \in K$ is equal to d_i and its cost is bounded by C . Moreover, the new flow is still weakly inflow-preserving. By Claim 3.16, its time horizon can be bounded by $(1+2\varepsilon')\hat{T} \leq (1+2\varepsilon')(1+3/2\varepsilon')T' \leq (1+4\varepsilon')T' \leq (1+\varepsilon/2)T' = T$. This concludes the proof of Lemma 3.14. We can now state the main result of this section.

Theorem 3.20. Assume that each transit time function $(\tau_a)_{a \in A}$ is a nonnegative, nondecreasing, left-continuous function given by an oracle as described in Remark 1.18. Then there is an FPTAS for the quickest multi-commodity flow problem with inflow-dependent transit times and bounded cost.

Quality of the Relaxation. In Section 3.2 we have introduced the notion of weakly inflow-preserving flows over time as a relaxation to flows over time with inflow-dependent transit times. It was shown in this section that this relaxation leads to an FPTAS for the quickest multi-commodity flow problem with bounded cost. This result provides evidence of the quality of this relaxation.

Theorem 3.21. Let \bar{T} denote the time horizon of a quickest multi-commodity flow with inflow-dependent transit times $(\tau_a)_{a \in A}$ in G .

- (i) For every $\varepsilon > 0$, there exists a bow graph G^B of size polynomial in the input size and $1/\varepsilon$ such that the time horizon \bar{T}^B of a quickest weakly inflow-preserving multi-commodity flow in G^B satisfies $\bar{T}^B \leq \bar{T} \leq (1+\varepsilon)\bar{T}^B$.
- (ii) Let all transit time functions be step functions and let G^B be the corresponding bow graph. Then the time horizon \bar{T}^B of a quickest weakly inflow-preserving multi-commodity flow f^B in G^B is equal to \bar{T} .

Proof. We begin with the proof of (ii). It follows from Observation 3.5 that $\bar{T}^B \leq \bar{T}$, so it remains to prove $\bar{T} \leq \bar{T}^B$. To see this, it suffices to show that for any $\varepsilon > 0$, there exists a flow over time f with inflow-dependent transit times in G with time horizon bounded from above by $(1 + \varepsilon)\bar{T}^B$. This follows from the result presented in Lemma 3.14: Consider a weakly inflow-preserving flow over time f^B with time horizon \bar{T}^B in G^B . Then, f^B can be interpreted as a weakly inflow-preserving flow over time in G^\downarrow , where G^\downarrow is the lower bow graph as defined at the beginning of Section 3.4.2. Lemma 3.14 implies that f^B can be turned into a weakly inflow-preserving flow over time in $G^{\uparrow\uparrow}$ with time horizon bounded by $(1 + \varepsilon)\bar{T}^B$. We interpret this flow as a weakly inflow-preserving flow over time in G^\uparrow which is Δ -resting. Applying Corollary 3.8 proves part (ii).

Next, we consider (i). Pick as bow graph G^B the lower bow graph G^\downarrow as defined at the beginning of Section 3.4.2. Because of the relaxation property of the bow graph $\bar{T}^B \leq \bar{T}$. Hence it remains to show $\bar{T} \leq (1 + \varepsilon)\bar{T}^B$. Let $T^{\uparrow\uparrow}$ denote the optimal time horizon of a weakly inflow-preserving flow over time in $G^{\uparrow\uparrow}$. It follows from the second part of this theorem that $\bar{T} \leq T^{\uparrow\uparrow}$. Moreover, Lemma 3.14 implies that $T^{\uparrow\uparrow} \leq (1 + \varepsilon)\bar{T}^B$. This concludes the proof. \square

3.5 COMPLEXITY

We close this chapter with a summary of the main complexity results known for flows over time with constant transit times and with flow-dependent transit times. In Table 3.1 four problem settings are considered: s - t -flow, transshipment, minimum cost s - t -flow, and multi-commodity flow. We focus on those entries in the table which are presented in this thesis.

Consider the first row in the table which discusses constant transit times. The algorithm of Ford and Fulkerson, which solves the maximum s - t -flow over time problem with constant transit times, is presented in Section 1.3.5. The reduction of the problem to a static minimum cost s - t -flow problem can also be found there. Moreover, the quickest s - t -flow problem is polynomial time solvable; see Section 1.3.5. For integral data, all four problem versions can be translated to static flow problems in the corresponding time-expanded graph; see Section 1.3.4. In particular, for integral time horizon T , all problems are solvable in pseudo-polynomial time since the respective static network flow problems are polynomial time solvable.

Consider the second row in the table which discusses inflow-dependent transit times. The strong \mathcal{NP} -hardness of the quickest s - t -flow problem in

Table 3.1: The complexity of flows over time with constant transit times in comparison to the corresponding flow problems with inflow-dependent and load-dependent transit times, respectively. The entry "poly" ("pseudo-poly") means that the problem can be solved exactly in polynomial (pseudo-polynomial) time.

transit times	s - t -flow	transshipment	min-cost flow	multi-commodity flow
constant	poly [18] (\simeq static min cost flow)	poly [34] (\simeq submod. function)	pseudo-poly, \mathcal{NP} -hard [42], FPTAS [14]	pseudo-poly, \mathcal{NP} -hard [27], FPTAS [14]
inflow-dependent	strongly \mathcal{NP} -hard, FPTAS [28]			
load-dependent	\mathcal{APX} -hard [44] ($2 + \varepsilon$)-approx. [44]			

the setting of inflow-dependent transit times is proven in Section 2.4 by a reduction from the \mathcal{NP} -complete problem 3-PARTITION. The FPTAS, which is even applicable to the quickest multi-commodity transshipment problem with costs, is discussed in Section 3.4. It relies on a linear programming formulation on a condensed time-expanded graph. Constant factor approximation algorithms for the quickest s - t -flow problem in the setting of inflow-dependent transit times are not explicitly mentioned in the table. They can be found in Section 2.3.

The third row contains results known for the model of load-dependent transit times. Computing quickest s - t -flows is already \mathcal{APX} -hard; in particular, there does not exist a polynomial time approximation scheme for the quickest s - t -flow problem, unless $\mathcal{P} = \mathcal{NP}$. The $(2 + \varepsilon)$ -approximation algorithm of Köhler and Skutella [44], which solves the quickest s - t flow problem in the setting of load-dependent transit times, is presented in Section 2.3.5. It is also shown there that the algorithm can be applied to other, more general models of flow-dependent transit times; in particular, it can be used to approximate the quickest s - t -flow problem in the setting of inflow-dependent transit times.

CHAPTER 4

THE FIRST IN, FIRST OUT PROPERTY

4.1 INTRODUCTION

The model of flows over time with inflow-dependent transit times is only a rough approximation of the actual behavior of flow in real-life applications. For instance, flows over time with inflow-dependent transit times do in general not obey the *first in, first out (FIFO) property* on an arc. The FIFO property states that traffic that enters a road at a particular time exits the facility before traffic which enters in later periods. Translated to time-varying flows, the FIFO condition requires that flow particles are entering and leaving an arc in the same order. In the model of inflow-dependent transit times, a violation of the FIFO property can arise if the inflow rate of an arc drops down quickly. This causes a strong decrease in the transit time of that arc such that some flow units might pass other flow units which are ahead on the arc traveling at a slower speed.

The FIFO property is not only relevant from a practical point of view, but also from a theoretical one: A generalization of the shortest path problem is the dynamic shortest path problem. Here, it is assumed that the length of an arc is not fixed but time-varying. Kaufman and Smith [40] prove that essentially any labeling algorithm such as Dijkstra’s method [12] can be used to compute a shortest path provided that the given time-dependent length functions satisfy the FIFO property. Hence, in that case a shortest path can be computed in the original network and an explicit time-expansion is avoided.

Although FIFO violations can emerge in the setting of inflow-dependent transit times, this phenomenon does not occur in the solutions generated by the constant factor approximation algorithms presented in Section 2.3. Namely, the algorithm outputs a temporally repeated flow; in such a flow, the transit time of every arc is fixed and, hence, flow units can never pass each other. Unfortunately, the solutions produced by the FPTAS presented in Section 3.4 do not satisfy the first in, first out property. Since in such a solution the inflow rate—and thus the transit time—on an arc is changing in a “round robin” fashion, the FIFO property will in general be violated.

Flows over time with load-dependent transit times always satisfy the first in, first out condition. In this model, flow units traveling on the same arc at the same time experience the same speed. Thus, the first in, first out property is an intrinsic characteristic of the model. From a practical standpoint, the latter feature favors the model of load-dependent transit times over the model of inflow-dependent transit times. However, theoretical results give evidence that the model of load-dependent transit times is not too promising from a mathematical programming standpoint: Köhler and Skutella [44] prove that computing a quickest s - t -flow in this model is \mathcal{APX} -hard. More precisely, they prove that there does not exist an approximation algorithm with performance guarantee better than $37/36$, unless $\mathcal{P} = \mathcal{NP}$. Moreover, at this point no approximation algorithm is known that computes good multi-commodity flow solutions.

In this chapter we analyze a much simpler model of flow-dependent transit times with regard to the FIFO property. In Chapter 2 we have introduced the fan graph. We have seen that the solution space defined by static flows on the fan graph can be seen as a relaxation of the space of flows over time with inflow-dependent transit times. The advantage of this model is that the algorithmic machinery developed for static network flows can be applied to this generalized time-expanded network. As will be explained in Section 4.3, the first in, first out property does not necessarily hold in the fan graph model. Hence, a natural question to ask is whether one can enforce the FIFO property, for instance by imposing additional constraints. The scope of this chapter is to prove that finding a static multi-commodity flow in the fan graph that satisfies the first in, first out property is \mathcal{NP} -hard. For the single source, single sink case, we will present some simple approximation results.

Carey and Subrahmanian [9] study static flows in a fan graph similar to the one presented in Chapter 2 with a slightly different interpretation of arc transit times. They address the problem of ensuring the first in, first out property in a flow solution and propose to introduce additional nonconvex constraints. These constraints can be formulated as linear constraints with zero-one integer variables. They note that this approach makes the mathematical program much less tractable and alternatively suggest to add only few such constraints initially and repeatedly add new or drop old constraints. However, this iterative approach is a heuristic method, and it is not clear that it converges towards an optimal solution.

Remark 4.1. Recall that the fan graph is defined as the time-expansion of the bow graph. In this thesis we have considered different types of bow graphs; see Remark 2.32 and Figure 2.11 on page 59. In this chapter we

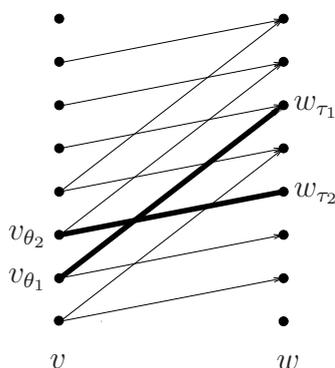


Figure 4.1: Example of a fan graph over a single arc (v, w) ; a FIFO violation occurs if two crossing arcs in the fan graph carry flow.

concentrate on the fan graph which is the time-expansion of the bow graph presented in Figure 2.11 (c). Notice however that the results presented in this chapter hold for all three types of fan graphs.

4.2 FIRST IN, FIRST OUT VIOLATIONS IN THE FAN GRAPH

In this section we give a formal definition of the first in, first out property for static flows in the fan graph. After that, we will prove that finding a static multi-commodity flow in the fan graph that in addition satisfies the first in, first out property is \mathcal{NP} -hard.

We are considering the following setting. Let $G = (V, A)$ be a directed graph, in which each arc $a \in A$ has a nonnegative capacity u_a and a piecewise constant transit time function¹ $\tau_a^s : [0, u_a] \rightarrow \mathbb{N}$. For a fixed integral time horizon T , let $G^F = (V^F, A^F)$ denote the fan graph with time horizon T constructed with respect to $(\tau_a^s)_{a \in A}$. In particular, V^F contains T copies of V , denoted $V^F(0), \dots, V^F(T-1)$. For every arc $a = (v, w) \in A$ and every point in time $0 \leq \theta \leq T-1$, there is a whole “fan” of arcs leaving $v(\theta)$; one arc for each possible transit time on arc a . Consider the example in Figure 4.1 which displays the fan graph over a single arc (v, w) . In the example, each fan contains two arcs; the first representing transit time one, the second representing transit time four.

For a static flow in the fan graph, we can formalize the first in, first out property as follows.

Definition 4.2. Let x be a static flow in G^F . The flow x violates the first

¹The transit time function is denoted τ_a^s to stress its step function character.

in, first out property if there exist flow-carrying arcs $e_i = (v_{\theta_i}, w_{\xi_i}) \in A^F$, $i \in \{1, 2\}$, representing the same arc $(v, w) \in A$, such that $\theta_1 < \theta_2$ and $\xi_1 > \xi_2$.

A FIFO violation of x has an intuitive interpretation in the layout of the fan graph depicted in Figure 4.1; the respective arcs are crossing each other.

4.3 COMPLEXITY

We are considering the problem of deciding whether, for a given time horizon T , the fan graph contains a multi-commodity flow for certain demands satisfying the first in, first out property. We prove that this problem is strongly \mathcal{NP} -hard. The result follows by a reduction from the well-known \mathcal{NP} -complete problem 3-SATISFIABILITY.

First, we need to specify the exact input. We are given a directed graph $G = (V, A)$, where each arc $a \in A$ has a nonnegative integral capacity u_a and a piecewise constant transit time function $\tau_a^s : [0, u_a] \rightarrow \mathbb{N}$. Each function τ_a^s is given by integral breakpoints $0 = u_0 < u_1 < \dots < u_\ell = u_a$ and corresponding integral transit times $\tau_1 < \dots < \tau_\ell$. Moreover, a set of commodities $K = \{1, \dots, k\}$ is specified, where every commodity $i \in K$ is defined by a source-sink pair $(s_i, t_i) \in V \times V$. For each commodity $i \in K$, an integral demand $d_i \geq 0$ is given. Finally, an integral time horizon T is given.

By definition, a static multi-commodity flow x in G^F satisfies the demand of commodity i if it sends d_i units from $s_i(0)$ to $t_i(T - 1)$ in G^F ; see Section 1.3.3. We are considering the following decision problem.

Problem 4.3. Let $T \in \mathbb{N}$ be given. Decide whether there exists a static multi-commodity flow in G^F that satisfies all demands and obeys the first in, first out property.

Theorem 4.4. Problem 4.3 is strongly \mathcal{NP} -complete.

We give a reduction from the strongly \mathcal{NP} -complete problem 3-SATISFIABILITY; see, e.g., Garey and Johnson [23].

Problem 4.5 (3-SATISFIABILITY (3-SAT)). Given a set of m clauses C_1, \dots, C_m on a set of n variables x_1, \dots, x_n such that each C_i is the conjunction of three literals of different² variables. Decide whether there exists a truth assignment of the variables that satisfies all clauses.

²Without loss of generality, one can assume that the variables of each clause are different: If $C = (x \vee \bar{x} \vee y)$, drop C . If $C = (x \vee x \vee y)$, add an auxiliary variable z and replace C by clauses $C' := (x \vee y \vee z)$ and $C'' := (x \vee y \vee \bar{z})$.

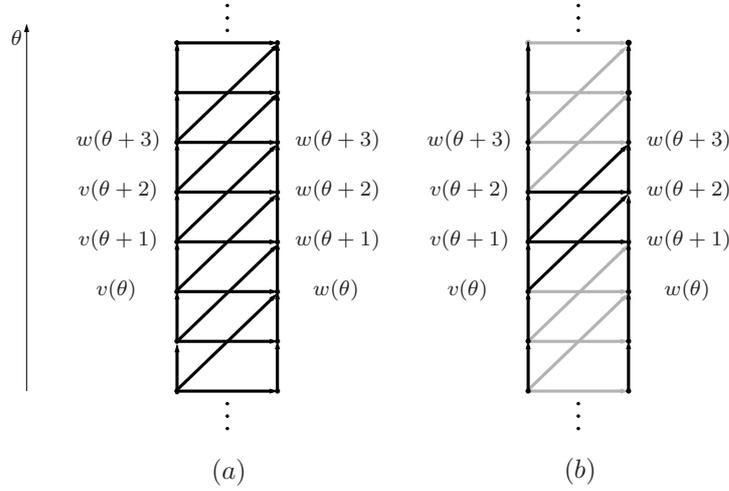


Figure 4.2: Figure (a) displays the full expansion of arc (v, w) in the fan graph. By introducing extra commodities, the fan graph can be thinned out to yield the sparser graph shown in (b).

To prove Theorem 4.4, we pursue the following strategy. Given an instance of 3-SAT with m clauses and n variables, we define an instance of Problem 4.3 with input size polynomial in m and n . We show that for $T := 8m$ the answer to Problem 4.3 is “yes” if and only if the 3-SAT instance is satisfiable. An important ingredient to the proof is the FIFO gadget.

The FIFO gadget. Consider an original arc $a = (v, w)$ with capacity $u_a := 5$ and transit time function τ_a^s defined as follows:

$$\tau_a^s(x) := \begin{cases} 0 & \text{if } 0 \leq x \leq 1, \\ 2 & \text{if } 1 < x \leq 5. \end{cases}$$

This arc is represented in the fan graph as shown in Figure 4.2 (a). By definition, every arc $(v(\theta), w(\theta))$, $\theta = 0, \dots, T - 1$, has capacity 1, and every arc $(v(\theta), w(\theta + 2))$, $\theta = 0, \dots, T - 3$, has capacity 5.

Consider a point in time $0 \leq \theta < T - 2$. We apply the following trick to thin out the fan graph before time θ and after time $\theta + 2$. For every arc $(v(\xi), w(\xi))$, $\xi \notin \{\theta + 1, \theta + 2\}$, we introduce a commodity with demand 1. In total, we need another $T - 2$ commodities. More precisely, in the original graph we introduce a new source node which we connect to node v via an arc that has capacity 1 and transit time ξ . Similarly, we introduce a new sink node to which we connect w by an arc that has capacity 1 and constant transit time $T - 1 - \xi$; see Figure 4.3. In order to satisfy the demand of this

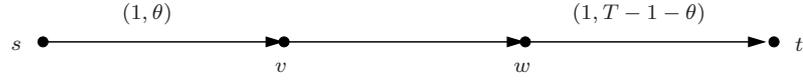


Figure 4.3: Introducing an extra commodity (s, t) guaranties that arc (v, w) is blocked at time θ .

new commodity by time $T - 1$, any static flow in the corresponding fan graph must use arc $(v(\xi), w(\xi))$ at full capacity.

Since we are only interested in solutions obeying the first in, first out property, the arcs of type $(v(\xi), w(\xi + 2))$, $\xi \notin \{\theta, \theta + 1\}$, cannot carry flow due to the auxiliary commodities (compare Figure 4.2 (b)). Hence, these arcs are blocked as well. In particular, in the fan graph, no flow can travel through arc a before time θ and after time $\theta + 2$. We say that the FIFO gadget is *open at time* θ , i.e., only the arcs $(v(\theta), w(\theta + 2))$, $(v(\theta + 1), w(\theta + 1))$, $(v(\theta + 1), w(\theta + 3))$, and $(v(\theta + 2), w(\theta + 2))$ can carry flow. These arcs are highlighted in Figure 4.2 (b).

We use the sparser fan graph to force zero, one decisions. Assume that we want to send $3 \leq d \leq 5$ units of flow from v to w and assume the gadget is open at time θ . If arc $(v(\theta + 1), w(\theta + 1))$ does not carry flow, we say that the FIFO gadget is *empty at time* θ . In this case, all d units of flow can be routed along arc $(v(\theta), w(\theta + 2))$ and can reach node w at time $\theta + 2$. Otherwise the FIFO gadget is *blocked at time* θ , i.e., arc $(v(\theta + 1), w(\theta + 1))$ carries flow and a first in, first out conflict can occur. In that case only arcs $(v(\theta + 1), w(\theta + 1))$, $(v(\theta + 1), w(\theta + 3))$, and $(v(\theta + 2), w(\theta + 2))$ are allowed to carry flow. Since the capacity of arc $(v(\theta + 1), w(\theta + 1))$ is 1, we need to assign at least $d - 1 \geq 2$ units of flow to arcs $(v(\theta + 1), w(\theta + 3))$ and $(v(\theta + 2), w(\theta + 2))$. Again, flow can only be assigned to one of the two arcs because of the first in, first out rule. Since the “faster” arc $(v(\theta + 2), w(\theta + 2))$ can carry at most 1 unit of flow, all remaining flow must be routed along $(v(\theta + 1), w(\theta + 3))$. Consequently, at least $d - 1$ units of flow arrive in w at time $\theta + 3$.

Summarizing, two cases can occur. Either the FIFO gadget is empty at time θ , then all d units of flow can reach w until time $\theta + 2$. Or the FIFO gadget is blocked at time θ , then at least $d - 1$ units of flow arrive in w at time exactly $\theta + 3$.

The underlying graph G . We are ready to define the underlying graph G of the fan graph G^F that encodes a given 3-SAT instance. The graph G only contains arcs with constant transit times and FIFO gadgets.

We start with the representation of clauses in graph G . For every clause C in $\{C_1, \dots, C_m\}$, we introduce a commodity that is given by a source-sink

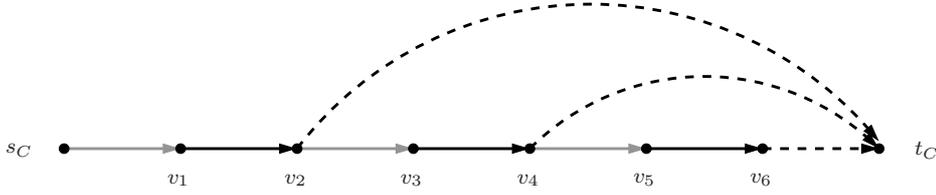


Figure 4.4: Representation of a clause C as a commodity pair (s_C, t_C) which is connected by a path with two extra arcs attached to it. The path contains exactly three FIFO gadgets represented by black arcs; one FIFO gadget for each variable contained in the clause.

pair (s_C, t_C) . The demand of each *clause commodity* is 5. We connect the source to the sink by a path $P := (s_C, v_1, \dots, v_6, t_C)$ and attach two extra arcs (v_2, t_C) and (v_4, t_C) to this path as depicted in Figure 4.4.

The arcs $f_1 := (v_1, v_2)$, $f_2 := (v_3, v_4)$, and $f_3 := (v_5, v_6)$ (indicated by black arcs in Figure 4.4) represent FIFO gadgets. The path contains one FIFO gadget for each variable in the clause. Assume that clause C contains variables x_{i_1} , x_{i_2} , and x_{i_3} with $i_1 < i_2 < i_3$. Then, we say that variable x_{i_1} is in first *position*, variable x_{i_2} is in second position, and variable x_{i_3} is in third position of clause C . Moreover, we define the *index* of f_j to be i_j , $j \in \{1, 2, 3\}$. Notice that the FIFO-gadgets of clause C are located on path P in order of increasing index. Globally speaking, each FIFO gadget in G represents a clause-variable incidence, where the index of the variable defines the index of the FIFO gadget.

Every FIFO gadget has a successor arc (indicated by dashed arcs in the figure) leading directly to the sink. In particular, flow coming from the source s_{C_j} must use one of the arcs (v_i, t_{C_j}) , $i \in \{2, 4, 6\}$, to reach the sink t_{C_j} . All arcs except the FIFO gadgets have capacity 5.

We continue with the representation of variables in G . We introduce one commodity pair (s_{x_i}, t_{x_i}) for each variable x_1, \dots, x_n . Each of these commodities has demand 1. The source s_{x_i} is connected to the sink t_{x_i} by a TRUE-path and a FALSE-path. Next, we explain how these paths are interwoven with the paths connecting commodities.

Consider the example in Figure 4.5. In the example, we are given four clauses C_1, \dots, C_4 represented by vertical paths with two arcs attached to it as explained in the above paragraph. The first clause contains literal x_i in second position; hence, we direct the FALSE-path through the second FIFO gadget on the path connecting commodity pair (s_{C_1}, t_{C_1}) . Note that, by definition, the index of this FIFO gadget is i . Clause C_3 contains literal x_i in third position; thus, the FALSE-path leads through the third FIFO gadget on the path connecting commodity pair (s_{C_3}, t_{C_3}) . Again, by definition, the

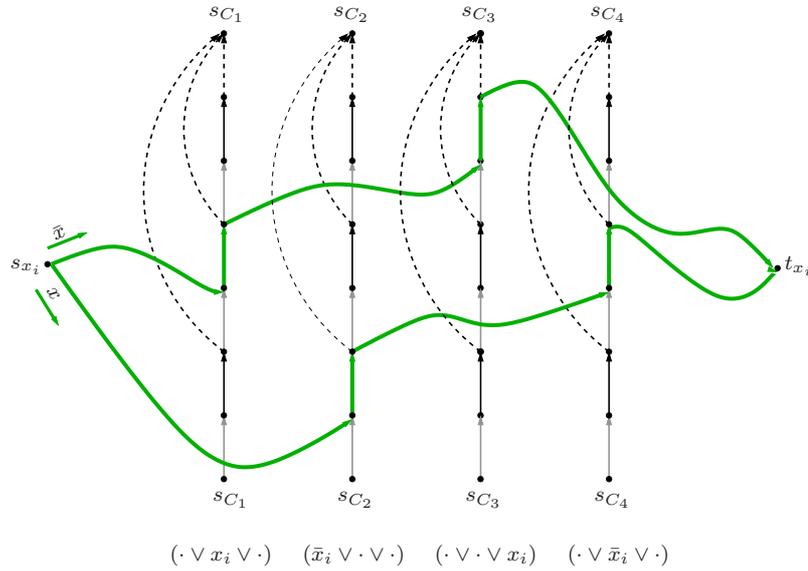


Figure 4.5: Example of a graph representing a 3-SAT-instance consisting of four clauses C_1, \dots, C_4 . Each clause is represented by a commodity pair (s_{C_i}, t_{C_i}) , $i = 1, \dots, 4$. Variable x appears in each of the commodities either as literal x_i or \bar{x}_i . Commodity pair (s_{x_i}, t_{x_i}) represents variable x_i . The source s_{x_i} is connected to the sink t_{x_i} via a TRUE- and a FALSE-path. The TRUE-path visits one FIFO gadget of each clause with literal \bar{x}_i ; analogously, the FALSE-path visits one FIFO gadget of each clause with literal x_i .

index of this FIFO gadget is i . That way, the FALSE-path traverses exactly one of the three³ gadgets of each clause with literal x_i . The position of literal x_i in the clause determines which of the three FIFO gadgets is chosen. Moreover, the FALSE-path only visits FIFO gadgets with index i .

Similarly, the TRUE-path visits exactly one of the three FIFO gadgets of each clause with literal \bar{x}_i , namely those with index i . In the example, clauses C_2 and C_4 both contain literal \bar{x}_i . Thus, the TRUE-path visits the corresponding FIFO gadgets on the paths connecting commodity pair (s_{C_2}, t_{C_2}) and commodity pair (s_{C_4}, t_{C_4}) . All non-gadget arcs on both the TRUE- and the FALSE-path have capacity 1.

Claim 4.6. For every variable x_i , $i = 1, \dots, n$, there are exactly two paths in G leading from the source node s_{x_i} to the sink node t_{x_i} . These paths are given by the TRUE- and FALSE-path of variable x_i .

Proof. First of all note that the TRUE- and the FALSE-paths of variable x_i

³By assumption, each clause has three different variables.

only share the source node s_{x_i} and the sink node t_{x_i} . This holds because each clause contains a variable either negated or un-negated.

Consider an arbitrary s_{x_i} - t_{x_i} -path P in G . We show that it coincides either with the TRUE- or with the FALSE-path. Without loss of generality, assume that the path P and the TRUE-path share the first arc. Assume that, on its way to the sink node t_{x_i} , path P traverses FIFO gadgets f_1, \dots, f_ℓ . By construction, the indices of f_1, \dots, f_ℓ form a nondecreasing sequence; simply recall that we have ordered the FIFO gadgets of each clause by increasing index. Since the TRUE- and the FALSE-path only traverse FIFO gadgets representing variable x_i , their sequences are given by (i, \dots, i) . Then, the sequence of path P must be identical to i , as well: Since the sequence is nondecreasing, it suffices to show that the first and the last entry of the sequence are equal to i . Since path P shares the first FIFO gadget of the TRUE-path, the first entry is equal to i . On the other hand, path P either contains the last FIFO gadget on the TRUE-path or it contains the last FIFO gadget on the FALSE-path. Hence, the last entry in the sequence is equal to i . It is easy to see that the sequence uniquely determines the path. This concludes the proof. \square

Encoding a 3-SAT instance. It follows from Claim 4.6 that for each variable x_i , $i = 1, \dots, n$, there are two ways to route flow from the source s_{x_i} to the sink t_{x_i} . If, in a solution, flow is sent along the TRUE-path, the variable x_i is set to TRUE, otherwise it is set to FALSE. Flow traveling along the TRUE-path blocks out those commodities which contain literal \bar{x} , analogously, flow traveling along the FALSE-path blocks out those commodities which contain literal x . More precisely, let (s_{C_j}, t_{C_j}) , $j \in \{1, \dots, m\}$, be a commodity with literal \bar{x} in position $\alpha \in \{1, 2, 3\}$. We assign transit times in such a manner that flow traveling along the FALSE-path of variable x_i blocks the α^{th} gadget of clause C_j at the moment when flow traveling from s_{C_j} to t_{C_j} is about to traverse this gadget. This will cause a delay of the “clause-flow” by 1. If its total delay is greater than 2, the flow cannot reach the sink t_{C_j} on time.

It remains to fix transit times in G . More precisely, we need to specify the transit times of all non-gadget arcs and the opening times of all FIFO gadgets in G . In the following, let $T = 8m$. For every $j \in \{1, \dots, m\}$, we specify the transit times of arcs lying on (respectively, attached to) the path connecting the source s_{C_j} to the sink t_{C_j} ; see Figure 4.6.

Consider the non-gadget arcs on the path (indicated by gray arcs in the figure). The transit time of the first non-gadget arc on the path is set to $8(j - 1)$. This choice will guarantee that the flow of clause C_j enters the first gadget when the routing of clause C_{j-1} is essentially completed. All other non-gadget arcs on the path have transit time zero.

the transit times of all non-gadget arcs on the path. We pick these in such a way that flow starting in the source node s_{x_i} at time zero blocks each gadget on the TRUE-path at its opening time. In order to choose transit times appropriately, we need to make sure that the opening times form a monotone increasing sequence. This follows from the fact that for $j \in \{1, \dots, m\}$, the opening time of any gadget of clause C_j lies in $[8(j-1), 8(j-1)+6]$. Finally, the transit time of the last arc on the path is chosen such that the transit time of all non-gadget arcs on the path add up to $T-1$.

Bounding the size of G . Every representation of a clause contains three FIFO gadgets each of which involves $T-2 = 8m-2$ auxiliary commodities; see Figure 4.4. Hence, the number of arcs associated with all m clauses lies in $\mathcal{O}(m^2)$. Each representation of a variable consists of two paths of length at most $2m$; this implies that the number of arcs in G lies in $\mathcal{O}(m^2 + mn)$.

The following lemma proves Theorem 4.4. That is, deciding whether G^F contains a feasible static multi-commodity flow that satisfies the first in, first out property is a strongly \mathcal{NP} -complete problem.

Lemma 4.7. There exists a truth-assignment for clauses C_1, \dots, C_m if and only if, the fan graph G^F contains a feasible multi-commodity flow satisfying all demands and obeying the first in, first out property.

Proof. Let $\mathcal{T} : \{x_1, \dots, x_n\} \rightarrow \{\text{TRUE}, \text{FALSE}\}$ be a truth-assignment for clauses C_1, \dots, C_m . Consider variable x_i , $i = 1, \dots, n$. If $\mathcal{T}(x_i) = \text{TRUE}$ (respectively, FALSE), at time zero we send one unit of flow into the TRUE-path (respectively, FALSE-path) of x_i . Since each clause is fulfilled, at least one of its three FIFO gadgets is empty. Hence, every flow unit arrives at its sink not later than $T-1$.

On the other hand, if a feasible routing exists, then we can define a truth-assignment as follows. Consider variable x_i , $i = 1, \dots, n$. In the flow solution, one unit of flow is routed from the source node s_{x_i} to the sink node t_{x_i} . It follows from Claim 4.6 that there are exactly two paths leading from source s_{x_i} to sink t_{x_i} , namely the TRUE- and the FALSE-path. If only the TRUE-path (respectively, only the FALSE-path) carries flow, then we set $\mathcal{T}(x_i) := \text{TRUE}$ (respectively, FALSE). If in the flow solution both paths carry a fraction of the flow, then we can set $\mathcal{T}(x_i)$ either to TRUE or to FALSE. In that case, the truth-assignment is independent of the value of the variable x_i . Since all flow units arrive at their sink on time, all clauses must be fulfilled. \square

In the minimization version of Problem 4.3, we ask for the minimum integral time horizon T such that the fan graph G^F contains a static multi-

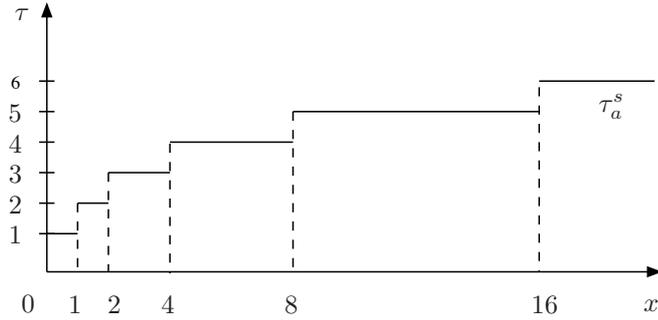


Figure 4.7: Example of a transit time function τ_a^s with geometrically increasing breakpoints.

commodity flow which satisfies the demands and obeys the first in, first out property. Theorem 4.3 yields the following non-approximability result.

Corollary 4.8. There exists no FPTAS for the optimization version of Problem 4.3, unless $\mathcal{P} = \mathcal{NP}$.

Proof. Since we only consider integral time horizons, the statement follows from a well-known fact on strongly \mathcal{NP} -complete problems; see, e.g., [4]. \square

4.4 GEOMETRICALLY INCREASING CAPACITIES

Deciding whether, for a given time horizon T , the fan graph contains a multi-commodity flow satisfying the first in, first out property is a strongly \mathcal{NP} -complete problem; see Theorem 4.4. In this section we consider the single source, single sink version of the problem. We study the special case that each arc $a \in A$ has infinite capacity u_a and a piecewise constant transit time function τ_a^s where the breakpoints of τ_a^s form a geometrically increasing sequence.

More precisely, for each arc $a \in G$, we are given integral values $q_a \geq 2$ and $p_a \geq 0$. The breakpoints of τ_a^s are given by the sequence $(q_a^i)_{i \in \mathbb{N}}$ where $\tau_a^s(x) := p_a$, for $x \in (0, 1]$, and $\tau_a^s(x) := i + 1 + p_a$, for $x \in (q_a^i, q_a^{i+1}]$, $i \in \mathbb{N}$. In Figure 4.7, an example of such a transit time function is shown with $q_a := 2$ and $p_a := 1$. For a given integral time horizon T , let $G^F := (V^F, A^F)$ be the fan graph with time horizon T with respect to the above defined transit time functions $(\tau_a^s)_{a \in A}$. For every arc $a = (v, w) \in G$, let A_a^F denote those arcs in A^F associated with a , i.e., $A_a^F := \{(v(\theta), w(\xi)) \mid \theta = 0, \dots, T - 1, \xi = \theta + p_a, \dots, T - 1\}$. By definition, the capacity u_e of

arc $e := (v(\theta), w(\xi))$ is $q_a^{\xi - p_a - \theta}$. In G , we have specified a source node s and a sink node t . Recall that, by definition, a static s - t -flow in G^F sends flow from $s(0)$ to $t(T - 1)$ in G^F .

Problem 4.9. Let $T \in \mathbb{N}$ be given. In G^F , determine a maximum s - t -flow among all s - t -flows that satisfy the first in, first out property.

In the following, we prove that a simple greedy strategy can be applied to remove FIFO violations in a given static s - t -flow x in G^F ; the new s - t -flow satisfies the FIFO condition and its value is at least half the value of x . Employing this method, we can compute a 2-approximate solution to Problem 4.9. Since the fan graph is of pseudo-polynomial size, the running time of this algorithm is pseudo-polynomial in the input size.

Remark 4.10. In this section, we assume that the transit time function τ_a^s of arc $a \in A$ is defined implicitly by integral values q_a and p_a . In particular, we consider q_a and p_a as input of the algorithm. If, for every arc $a \in A$, a list of the breakpoints $(q_a^i)_{i \leq T}$ is given as input, then T is polynomial in the input size and hence the size of the fan graph is polynomial in the input size. In this case, the algorithm described below is even a polynomial-time algorithm.

Lemma 4.11. Let x be a static s - t -flow in G^F of value d . Then, there exists a static s - t -flow \tilde{x} in G^F of value at least $d/2$ which satisfies the first in, first out property. Moreover, the flow \tilde{x} can be computed in pseudo-polynomial time.

Proof. We describe the algorithm that turns a given s - t -flow x in G^F into a flow \tilde{x} which satisfies the FIFO condition.

In each step of the algorithm, we process the expansion A_a^F of a single arc $a = (v, w) \in A$. We reassign flow such that, at the end of that step, no FIFO violation occurs in A_a^F . The modification preserves the total outflow $x(\delta^+(v(\theta))) := \sum_{\xi=\theta+p_a}^{T-1} x_{(v(\theta), w(\xi))}$ of every node $v(\theta)$, $\theta = 0, \dots, T - 1$, and it preserves the total inflow $x(\delta^-(w(\theta))) := \sum_{\xi=0}^{\theta-p_a} x_{(v(\xi), w(\theta))}$ of every node $w(\theta)$, $\tau = 0, \dots, T - 1$. After the reassignment, arc capacities might be violated. We prove that every arc in A_a^F carries at most twice as much flow as the arc capacity allows. Hence, dividing the flow on every arc by two yields a feasible static flow in G^F which satisfies the first in, first out property.

We continue with a detailed description of the subroutine which removes FIFO violations in A_a^F . A compact version is given on the following page.

Associated with each node $v(\theta)$, $\theta = 0, \dots, T - 1$, is a label $\ell(v(\theta))$ which measures the current supply of node $v(\theta)$ at each step of the algorithm; at the beginning the label $\ell(v(\theta))$ is set to $x(\delta^+(v(\theta)))$. Similarly each

Subroutine 6: Remove FIFO violations in A_a^F

Input: Expansion A_a^F of arc $a \in A$, a feasible static flow x in A_a^F .

Output: A static flow \tilde{x} in A_a^F that satisfies FIFO and with $\tilde{x}_e \leq 2u_e$,
for all $e \in A_a^F$.

```

1 for  $\theta = 0, \dots, T - 1$  do
     $\ell(v(\theta)) \leftarrow x(\delta^+(v(\theta)))$ ;
     $\ell(w(\theta)) \leftarrow x(\delta^-(w(\theta)))$ ;
endfor
2 for  $\theta = 0, \dots, T - 1$  do
    for  $\xi = \theta + p_a, \dots, T - 1$  do
         $\tilde{x}_{(v(\theta), w(\xi))} \leftarrow \min\{\ell(v(\theta)), \ell(w(\xi))\}$ ;
         $\ell(v(\theta)) \leftarrow \ell(v(\theta)) - \min\{\ell(v(\theta)), \ell(w(\xi))\}$ ;
         $\ell(w(\xi)) \leftarrow \ell(w(\xi)) - \min\{\ell(v(\theta)), \ell(w(\xi))\}$ ;
    endfor
endfor

```

node $w(\xi)$, $\xi = 0, \dots, T - 1$, carries a label $\ell(w(\xi))$ which stores the current demand of node $w(\xi)$ at each step of the algorithm; initially, the label $\ell(w(\xi))$ is set to $x^-(w(\xi))$.

We process the nodes $v(\theta)$, $\theta = 0, \dots, T - 1$, in order of increasing θ . When processing node $v(\theta)$, we assign its current supply greedily to the arcs $(v(\theta), w(\xi))$, $\xi = \theta + p_a, \dots, T - 1$. More precisely, for $\xi = \theta + p_a, \dots, T - 1$, the flow value of arc $(v(\theta), w(\xi))$ is set to the minimum of the current supply of node $v(\theta)$ and the current demand of node $w(\xi)$, i.e., $\tilde{x}_{(v(\theta), w(\xi))} := \min\{\ell(v(\theta)), \ell(w(\xi))\}$. Afterwards the current supply of node $v(\theta)$ and the current demand of node $w(\xi)$ are decreased correspondingly.

Obviously, after this procedure $\tilde{x}(\delta^+(v(\theta))) = x(\delta^+(v(\theta)))$ holds and similarly $\tilde{x}(\delta^-(w(\theta))) = x(\delta^-(w(\theta)))$, $\theta = 0, \dots, T - 1$. Moreover, we claim that no FIFO violation occurs in A_a^F . By contradiction, assume there exist flow-carrying arcs $e_i = (v_{\theta_i}, w_{\xi_i}) \in A_a^F$, $i \in \{1, 2\}$, such that $\theta_1 < \theta_2$ and $\xi_1 > \xi_2$. When processing node v_{θ_1} , the algorithm first visits node w_{ξ_2} and then node w_{ξ_1} . After visiting node w_{ξ_2} , either the label of node v_{θ_1} must be zero or the label of node w_{ξ_2} must be zero. Since afterwards the algorithm assigns flow to arc e_1 and to arc e_2 , neither the label of node v_{θ_1} nor the label of node w_{ξ_2} can be zero. This contradiction proves the statement.

It remains to show that the capacity of every arc $e \in A_a^F$ is violated by at most a factor of two. Consider an arc $e := (v(\theta), w(\xi))$, $\theta = 0, \dots, T - 1$, $\xi = \theta + p_a, \dots, T - 1$. By definition, its capacity is $u_e := q_a^{\xi - p_a - \theta}$. We prove that \tilde{x}_e is not larger than $2u_e$.

When processing arc e , the algorithm assigns at most $\ell(w(\xi))$ units of

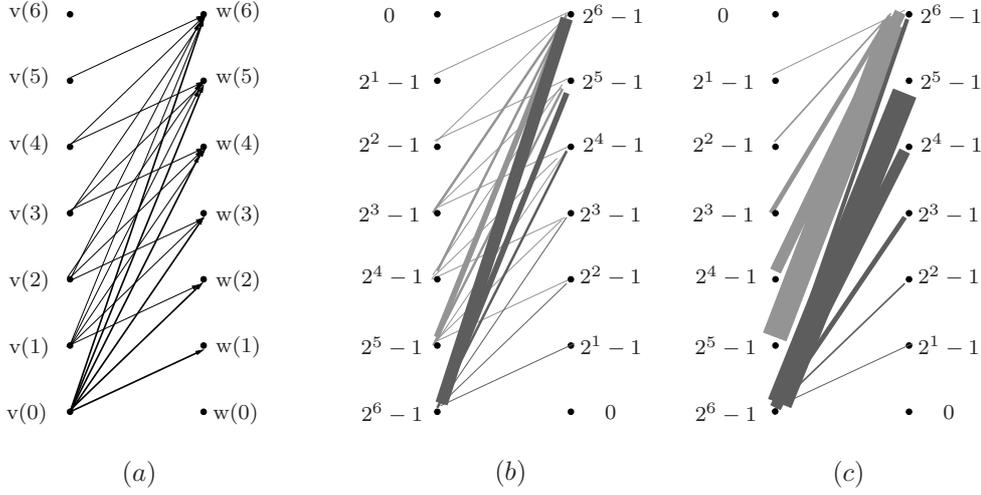


Figure 4.8: Example of how Subroutine 6 removes FIFO violations. Figure (a) displays the expansion of a single arc. The capacities of the arcs in a fan are geometrically increasing with respect to basis $q_a = 2$. The flow in (b) fills every arc up to its capacity and thus violates the first in, first out property wherever two flow-carrying arcs cross each other. The flow in (c) is the output of Subroutine 6. All FIFO violations are resolved by greedily rerouting flow from bottom to top.

flow to arc e , where $\ell(w(\xi))$ is the current demand of node $w(\xi)$. On the other hand, the current demand of node $w(\xi)$ is not larger than the total capacity of all arcs $(v(\theta'), w(\xi))$, $\theta' = \theta, \dots, \xi - p_a$, which, by definition, is

$$\sum_{\theta'=\theta}^{\xi-p_a} q_a^{\xi-p_a-\theta'} = \sum_{\theta'=0}^{\xi-p_a-\theta} q_a^{\theta'} = \frac{q_a^{\xi-p_a+1-\theta} - 1}{q_a - 1} \leq \frac{q_a}{q_a - 1} q_a^{\xi-p_a-\theta} \leq 2u_e.$$

This concludes the proof of the lemma. \square

The example given in Figure 4.8 demonstrates how the algorithm removes FIFO violations in A_a^F . In the example, parameter p_a is set to 1 and q_a is set to 2, thus the capacity of arc $(v(\theta), w(\xi))$ is $2^{\xi-1-\theta}$.

In Figure 4.8 (a), the expansion A_a^F is depicted. In (b), an example of a flow in A_a^F is shown. The flow leaving $v(0)$ is highlighted simply to improve visibility. Every node is labeled by its initial supply and demand. In the example, the flow fills every arc up to its capacity and thus several FIFO violations occur.

In Figure (c), all FIFO violations are resolved by satisfying the supplies and demands greedily from bottom to top as done by the algorithm. Notice that arc $(v(0), w(\xi))$, $\xi = 2, \dots, T - 2$, carries $2^\xi - 1$ units of flow while its

Algorithm 7: Approximating maximum s - t -flow with FIFO constraints

Input: Fan graph G^F , a source-sink pair $(s, t) \in V \times V$.

Output: A static s - t -flow x in G^F satisfying demand $\bar{d}/2$ and obeying the first in, first out property.

- 1 Compute static s - t -flow x in G^F of value \bar{d} ;
 - 2 **for** $a \in A$ **do**
 Remove FIFO violations in A_a^F (call Subroutine 6);
endfor
 - 3 **for** $e \in A^F$ **do**
 $x_e \leftarrow x_e/2$;
endfor
-

capacity is $2^{\xi-1}$. This shows that, for $q_a = 2$, the analysis of the algorithm is tight.

Theorem 4.12. Let \bar{d} denote the maximum value of a static s - t -flow in G^F . Then there exists a static s - t -flow of value $\bar{d}/2$ in G^F which satisfies the first in, first out property. Moreover, it can be computed in pseudo-polynomial time.

Proof. First compute a maximum static s - t -flow in G^F . Then apply the algorithm described in Lemma 4.11. \square

The main steps of the algorithm can be found above.

Remark 4.13. The described algorithm is also used in a different context. In the *Hitchcock problem* we ask for a minimum cost static flow on a bipartite graph. A feasible solution to this problem can be found using the above method. In the literature this method is referred to as the *north-west rule*; for more details see, e.g., [62].

Corollary 4.14. Given demand d , let \bar{T} denote the minimum time horizon such that the corresponding fan graph G^F contains a static s - t -flow x of value d . Similarly, let \bar{T}' denote the minimum time horizon such that the corresponding fan graph $G^{F'}$ contains a static s - t -flow x of value d satisfying the first in, first out property. Then $\bar{T}' \leq 2\bar{T}$.

Proof. By Theorem 4.12, there exists a static s - t -flow in G^F of value $d/2$ which satisfies the FIFO condition. Consider this flow in the enlarged fan graph $G^{F'}$. After time $\bar{T} - 1$, the network is empty, hence we can repeat the flow starting at time \bar{T} . Obviously, that way we have doubled the flow value. This proves the statement of the theorem. \square

BIBLIOGRAPHY

- [1] R. K. AHUJA, T. L. MAGNANTI, AND J. B. ORLIN, *Network Flows. Theory, Algorithms, and Applications*, Prentice Hall, Englewood Cliffs, NJ, 1993. [5, 8, 65]
- [2] R. K. AHUJA AND J. B. ORLIN, *A capacity scaling algorithm for the constrained maximum flow problem*, *Networks* **25** (1995), pp. 89–98. [63]
- [3] J. E. ARONSON, *A survey of dynamic network flows*, *Annals of Operations Research* **20** (1989), pp. 1–66. [6, 16]
- [4] G. AUSIELLO, P. CRESCENZI, G. GAMBOSI, V. KANN, A. MARCHETTI-SPACCAMELA, AND M. PROTASI, *Complexity and Approximation*, Springer, Berlin, 1999. [126]
- [5] J. H. BOOKBINDER AND S. P. SETHI, *The dynamic transportation problem: A survey*, *Naval Research Logistics Quarterly* **27** (1980). [2]
- [6] R. E. BURKARD, K. DLASKA, AND B. KLINZ, *The quickest flow problem*, *ZOR — Methods and Models of Operations Research* **37** (1993), pp. 31–58. [19, 43, 51, 70, 78]
- [7] R. G. BUSACKER AND P. J. GOWEN, *A procedure for determining a family of minimum-cost network flow patterns*, ORO Technical Paper 15, Operational Research Office, Johns Hopkins University, Baltimore, 1961. [79]
- [8] M. CAREY, *Optimal time-varying flows on congested networks*, *Operations Research* **35** (1987), pp. 58–69. [31, 32]
- [9] M. CAREY AND E. SUBRAHMANIAN, *An approach to modelling time-varying flows on congested networks*, *Transportation Research B* **34** (2000), pp. 157–183. [5, 33, 59, 116]
- [10] H. CHEN AND C. HSUEH, *A model and an algorithm for the dynamic user-optimal route choice problem*, *Transportation Research B* **32** (1998), pp. 219–234. [34]
- [11] W. J. COOK, W. H. CUNNINGHAM, W. R. PULLEYBLANK, AND A. SCHRIJVER, *Combinatorial Optimization*, John Wiley & Sons, New York, 1998. [8]
- [12] E. DIJKSTRA, *A note on two problems in connexion with graphs*, *Numerische Mathematik* **1** (1959), pp. 269–271. [115]

-
- [13] L. FLEISCHER AND M. SKUTELLA, *The quickest multicommodity flow problem*, in Integer Programming and Combinatorial Optimization, W. J. Cook and A. S. Schulz, eds., Lecture Notes in Computer Science 2337, Springer, Berlin, 2002, pp. 36–53. [5, 15, 21, 22, 94, 95, 96, 98, 101, 102]
- [14] L. FLEISCHER AND M. SKUTELLA, *Minimum cost flows over time without intermediate storage*, in Proceedings of the 14th Annual ACM–SIAM Symposium on Discrete Algorithms, Baltimore, MD, 2003, pp. 66–75. [5, 15, 21, 22, 101, 106, 107, 108, 110, 114]
- [15] L. FLEISCHER AND M. SKUTELLA, *Quickest flows over time* (2003). Submitted. [5, 6, 16]
- [16] L. K. FLEISCHER, *Faster algorithms for the quickest transshipment problem*, SIAM Journal on Optimization **12** (2001), pp. 18–35. [10]
- [17] L. K. FLEISCHER AND E. TARDOS, *Efficient continuous-time dynamic network flow algorithms*, Operations Research Letters **23** (1998), pp. 71–80. [16, 18, 19]
- [18] L. R. FORD AND D. R. FULKERSON, *Constructing maximal dynamic flows from static flows*, Operations Research **6** (1958), pp. 419–433. [3, 7, 9, 13, 14, 17, 19, 64, 114]
- [19] L. R. FORD AND D. R. FULKERSON, *Flows in Networks*, Princeton University Press, Princeton, NJ, 1962. [3, 7, 9, 13, 14, 17, 64]
- [20] T. L. FRIESZ, D. BERNSTEIN, T. E. SMITH, R. L. TOBIN, AND B. W. WIE, *A variational inequality formulation of the dynamic network user equilibrium problem*, Operations Research **41** (1993), pp. 179–191. [34]
- [21] T. L. FRIESZ, J. LUQUE, R. L. TOBIN, AND B. WIE, *Dynamic network traffic assignment considered as a continuous time optimal control problem*, Operations Research **37** (1989), pp. 893–901. [31, 33]
- [22] D. GALE, *Transient flows in networks*, Michigan Mathematical Journal **6** (1959), pp. 59–63. [20]
- [23] M. R. GAREY AND D. S. JOHNSON, *Computers and Intractability: A Guide to the Theory of NP–Completeness*, Freeman, San Francisco, 1979. [6, 20, 97, 118]
- [24] N. GARTNER, C. J. MESSER, AND A. K. RATHI, *Traffic flow theory: A state-of-the-art report*. <http://www.tfrc.gov/its/tft/tft.htm>. [23]
- [25] A. V. GOLDBERG AND R. E. TARJAN, *Finding minimum-cost circulations by canceling negative cycles*, Journal of the ACM **36** (1989), pp. 873–886. [18]

- [26] M. GRÖTSCHEL, L. LOVÁSZ, AND A. SCHRIJVER, *Geometric Algorithms and Combinatorial Optimization*, Algorithms and Combinatorics 2, Springer, Berlin, 1988. [5, 97, 105]
- [27] A. HALL, S. HIPPLER, AND M. SKUTELLA, *Multicommodity flows over time: Efficient algorithms and complexity*, in Automata, Languages and Programming, J. C. M. Baeten, J. K. Lenstra, J. Parrow, and G. J. Woeginger, eds., Lecture Notes in Computer Science 2719, Springer, Berlin, 2003, pp. 397–409. [22, 23, 87, 114]
- [28] A. HALL, K. LANGKAU, AND M. SKUTELLA, *An FPTAS for quickest multi-commodity flows with inflow-dependent transit times*, in Approximation, Randomization, and Combinatorial Optimization, S. Arora, K. Jansen, J. D. P. Rolim, and A. Sahai, eds., Lecture Notes in Computer Science 2764, Springer, Berlin, 2003, pp. 71–82. [35, 88, 114]
- [29] H. W. HAMACHER AND S. A. TJANDRA, *Mathematical modelling of evacuation problems: A state of art*, tech. report, Fraunhofer Institut Techno- und Wirtschaftsmathematik, 2001. [2]
- [30] R. HASSIN, *Approximation schemes for the restricted shortest path problem*, Mathematics of Operations Research **17** (1992), pp. 36–42. [97]
- [31] D. S. HOCHBAUM AND J. G. SHANTHIKUMAR, *Convex separable optimization is not much harder than linear optimization*, Journal of the Association for Computing Machinery **37** (1990), pp. 843–862. [65]
- [32] D. S. HOCHBAUM (ED.), *Approximation Algorithms for NP-Hard Problems*, Thomson, 1996. [6, 17]
- [33] B. HOPPE, *Efficient dynamic network flow algorithms*, PhD thesis, Cornell University, 1995. [6, 16]
- [34] B. HOPPE AND E. TARDOS, *The quickest transshipment problem*, Mathematics of Operations Research **25** (2000), pp. 36–62. [20, 21, 114]
- [35] M. IRI, *A new method of solving transportation-network problems*, Journal of the Operations Research Society of Japan **3** (1960), pp. 27–87. [79]
- [36] O. JAHN, R. H. MÖHRING, A. S. SCHULZ, AND N. E. STIER MOSES, *System optimal routing of traffic flows with user constraints in networks with congestion*, tech. report, TU Berlin, 2002. [24]
- [37] B. JANSON, *Dynamic traffic assignment for urban road networks*, Transportation Research B **25** (1991), pp. 143–161. [32]
- [38] R. JAYAKRISHNAN, W. K. TSAI, AND A. CHEN, *A dynamic traffic assignment model with traffic-flow relationships*, Transportation Research C **3** (1995), pp. 51–72. [32]

- [39] D. KAUFMAN, J. NONIS, AND R. SMITH, *A mixed integer linear programming model for dynamic route guidance*, *Transportation Research B* **32** (1998), pp. 431–440. [2, 5, 33]
- [40] D. KAUFMAN AND R. SMITH, *Minimum travel time paths in dynamic networks with application to intelligent vehicle/highway systems*, Tech. Report 90-34, Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, Michigan 48109, November 1990. [115]
- [41] D. E. KAUFMAN, R. L. SMITH, AND K. E. WUNDERLICH, *User-equilibrium properties of fixed points in dynamic traffic assignment*, *Transportation Research C* **6** (1998). [32]
- [42] B. KLINZ AND G. J. WOEGINGER, *Minimum cost dynamic flows: The series-parallel case*, in *Integer Programming and Combinatorial Optimization*, E. Balas and J. Clausen, eds., *Lecture Notes in Computer Science* 920, Springer, Berlin, 1995, pp. 329–343. [20, 21, 114]
- [43] E. KÖHLER, K. LANGKAU, AND M. SKUTELLA, *Time-expanded graphs with flow-dependent transit times*, in *Algorithms — ESA '02*, R. Möhring and R. Raman, eds., *Lecture Notes in Computer Science* 2461, Springer, Berlin, 2002, pp. 599–611. [35, 100]
- [44] E. KÖHLER AND M. SKUTELLA, *Flows over time with load-dependent transit times*, in *Proceedings of the 13th Annual ACM–SIAM Symposium on Discrete Algorithms*, San Francisco, CA, 2002, pp. 174–183. [4, 27, 28, 60, 61, 62, 63, 84, 114, 116]
- [45] B. KORTE AND J. VYGEN, *Combinatorial Optimization*, Springer, Berlin, 2002. [5, 8, 50]
- [46] D. H. LORENZ AND D. RAZ, *A simple efficient approximation scheme for the restricted shortest path problem*, *Operations Research Letters* **28** (2001), pp. 213–219. [97]
- [47] N. MEGIDDO, *Combinatorial optimization with rational objective functions*, *Mathematics of Operations Research* **4** (1979), pp. 414–424. [19]
- [48] D. K. MERCHANT AND G. L. NEMHAUSER, *A model and an algorithm for the dynamic traffic assignment problems*, *Transportation Science* **12** (1978), pp. 183–199. [31, 32]
- [49] D. K. MERCHANT AND G. L. NEMHAUSER, *Optimality conditions for a dynamic traffic assignment model*, *Transportation Science* **12** (1978), pp. 200–207. [31]
- [50] E. MINIEKA, *Maximal, lexicographic, and dynamic network flows*, *Operations Research* **21** (1973), pp. 517–527. [20]

-
- [51] N. C. MYERS, *A new and useful template technique: Traits*, C++ Report (1995), pp. 32–35. [75]
- [52] G. L. NEMHAUSER AND L. A. WOLSEY, *Integer and Combinatorial Optimization*, John Wiley & Sons, New York, 1988. [5]
- [53] J. B. ORLIN, *A faster strongly polynomial minimum cost flow algorithm*, Operations Research **41** (1993), pp. 338–350. [51]
- [54] C. H. PAPADIMITRIOU, *Computational Complexity*, Addison-Wesley, 1994. [25]
- [55] C. H. PAPADIMITRIOU AND K. STEIGLITZ, *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall Inc., Englewood Cliffs, N.J., 1982. [5, 97]
- [56] A. B. PHILPOTT, *Continuous-time flows in networks*, Mathematics of Operations Research **15** (1990), pp. 640–661. [20]
- [57] W. B. POWELL, P. JAILLET, AND A. ODoni, *Stochastic and dynamic networks and routing*, in Network Routing, M. O. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser, eds., Handbooks in Operations Research and Management Science 8, North-Holland, Amsterdam, The Netherlands, 1995, ch. 3, pp. 141–295. [6, 16]
- [58] B. RAN AND D. E. BOYCE, *Dynamic Urban Transportation Network Models: Theory and Implications for Intelligent Vehicle-Highway Systems*, Lecture Notes in Economics and Mathematical Systems 417, Springer, Berlin, 1994. [34]
- [59] B. RAN AND D. E. BOYCE, *Modelling Dynamic Transportation Networks*, Springer, Berlin, 1996. [34]
- [60] T. ROUGHGARDEN AND É. TARDOS, *How bad is selfish routing?*, Journal of the ACM **49** (2002), pp. 236–259. [24]
- [61] A. SCHRIJVER, *Theory of Linear and Integer Programming*, John Wiley & Sons, Chichester, 1986. [6]
- [62] A. SCHRIJVER, *Combinatorial Optimization*, Springer, Berlin, Heidelberg, 2003. [5, 19, 51, 130]
- [63] Y. SHEFFI, *Urban Transportation Networks*, Prentice-Hall, New Jersey, 1985. [23, 24, 76]
- [64] V. V. VAZIRANI, *Approximation Algorithms*, Springer, Berlin, 2001. [6, 17]
- [65] W. L. WILKINSON, *An algorithm for universal maximal dynamic flows in a network*, Operations Research **19** (1971), pp. 1602–1612. [20]

- [66] N. ZADEH, *A bad network problem for the simplex method and other minimum cost flow algorithms*, *Mathematical Programming* **5** (1973), pp. 255–266. [20]

SYMBOL INDEX

A	arc set of G	[7]
G^B	bow graph of G , $G^B = (V^B, A^B)$	[36]
C	budget	[21]
$c_{a,i}$	cost for sending flow of commodity i through arc a	[8]
d_i	demand of commodity i	[8]
$\delta^-(v)$	set of arcs entering v	[8]
$\delta^+(v)$	set of arcs leaving v	[8]
A_a^B	expansion of arc a in the bow graph G^B	[37]
$f_{a,i}(\theta)$	flow rate of commodity i on arc a at time θ	[10]
G^F	fan graph of G , $G^F = (V^F, A^F)$	[103]
G	directed graph with node set V and arc set A , $G = (V, A)$	[7]
$G(T)$	T -time-expanded graph of G	[15]
$G(T)/\Delta$	condensed time-expanded graph of G	[102]
G^\downarrow	lower bow graph, $G^\downarrow = (V^\downarrow, A^\downarrow)$	[103]
G^\uparrow	Δ -lengthened bow graph, $G^\uparrow = (V^\uparrow, A^\uparrow)$	[103]
$G^{\uparrow\uparrow}$	2Δ -lengthened bow graph, $G^{\uparrow\uparrow} = (V^{\uparrow\uparrow}, A^{\uparrow\uparrow})$	[103]
head(a)	head node of arc a	[8]
K	set of commodities, $K = \{1, \dots, k\}$	[8]
P^δ	path with delay	[107]
\mathbb{R}^+	the set of nonnegative reals including zero	[7]
s_i	source node of commodity i	[8]

S_i	set of terminals of commodity i , $S_i = S_i^+ \cup S_i^-$	[8]
S_i^+	set of source nodes of commodity i	[8]
S_i^-	set of sink nodes of commodity i	[8]
T	time horizon	[10]
t_i	sink node of commodity i	[8]
$\text{tail}(a)$	tail node of arc a	[8]
τ_a	transit time function of arc a	[7]
τ_a^s	piecewise constant transit time function of arc a	[7]
$\tau_P(x)$	transit time of path P , $\tau_P(x) := \sum_{a \in P} \tau_a(x_a)$	[8]
τ_P	transit time of path P (fixed transit times), $\tau_P := \sum_{a \in P} \tau_a$	[8]
\bar{T}^B	time horizon of a quickest flow in G^B	[43]
\bar{T}	time horizon of a quickest flow in G	[48]
$\tau(P^\delta)$	transit time with delay of a path with delay P^δ	[108]
\bar{T}^s	time horizon of a quickest flow in G with transit times τ_a^s	[48]
\bar{T}^t	time horizon of a quickest temporally repeated flow in G	[51]
u_a	capacity of arc a	[7]
V	node set of G	[7]
\mathbb{Z}^+	the set of nonnegative integers including zero	[7]

INDEX

- Δ -condensed fan graph, 103
- Δ -lengthened bow graph, 103
- Δ -resting, 94, 99, 104
- 3-PARTITION, 70
- 3-SAT, *see* 3-SATISFIABILITY
- 3-SATISFIABILITY, 118

- approximation algorithm, 17, 42–67, 94–113
- artificial node, *see* bow graph

- bow arc, *see* bow graph
- bow graph, 36, 89
 - artificial node, 37
 - bow arc, 37, 89
 - expansion, 37, 89
 - original node, 37
 - regulating arc, 37, 59

- capacity constraint, 8, 11, 13
- complexity, 70–74, 113–114, 118–126
- computational results, 74–85
- concave transit times, *see* transit times
- condensed time-expansion, 101, 103
- continuous flow over time, *see* flow over time
- convex transit times, *see* transit times
- cost, 8, 11, 13, 20, 26

- Davidson’s function, 77
- demand, 8, 11, 26
- discrete flow over time, *see* flow over time
- duality theory, 96
- dynamic network flow, 10
- dynamic traffic assignment, 1, 30–34

- earliest arrival flow, 19
- ellipsoid method, 97, 105
- evaluation oracle, *see* oracle

- fan graph, 58–60, 103, 115–131
 - Δ -condensed, *see* Δ -condensed fan graph
- feasible, 8, 11, 13, 26
- FIFO, *see* first in, first out property

- first in, first out property, 4, 115–131
- flow, *see* network flow
- flow conservation, 8, 10, 13, 26
- flow over time, 10
 - s - t -flow, 11
 - value, 11
 - continuous, 10, 15, 16
 - discrete, 13, 15
 - multi-commodity, 10, 21, 22, 87–113
 - supply, demand, 11
 - smoothed, 108
 - temporally repeated, 12
 - transshipment, 11
 - with inflow-dependent transit times,
 - see* inflow-dependent transit times
 - with load-dependent transit times, *see* load-dependent transit times
- flow-dependent transit times, 23–30
 - inflow-dependent, *see* inflow-dependent transit times
 - load-dependent, *see* load-dependent
 - temporally repeated flow, 28, 51, 64–70
- FPTAS, *see* fully polynomial approximation scheme
- free-flow travel time, 76
- fully polynomial approximation scheme, 17, 101

- inflow-dependent transit times, 26–27
- inflow-preserving, 38, 90

- latest departure flow, 20
- length-bounded shortest path, 97
- load, 27
- load-dependent transit times, 27–28

- maximum flow over time, 17
- Megiddo’s method of parametric search, 19, 43
- minimum cost flow over time, 20
- multi-commodity, 8, 10, 21, 22, 87–113

- network flow, 7–30
 - over time, *see* flow over time

- static, *see* static network flow
- time-dependent, *see* time-dependent flow
- optimization and separation, 97
- oracle, 25, 48, 50, 51
- original node, *see* bow graph
- path decomposition, 9, 12
- path with delay, 107
- per capacity inflow rate, 91
- per capacity inflow value, 103
- performance guarantee, 17
- performance ratio, 17
- polynomial approximation scheme, 17
- practical capacity, 76
- PTAS, *see* polynomial approximation scheme
- quickest flow, 19
- quickest inflow-dependent flow, 43, 46
- quickest inflow-dependent flow with costs, 87, 98
- quickest multi-commodity flow, 22
- quickest multi-commodity transshipment with costs, 22
- quickest transshipment, 21
- quickest transshipment with costs, 21
- quickest weakly inflow-preserving flow, 94–98
- regulating arc, *see* bow graph
- relaxation, 35–43, 87–94, 112
- revised simplex method, 97
- smoothed flow over time, *see* flow over time
- static convex cost flow, 60
- static network flow, 8
 - s - t -flow, 9
 - value, 9
 - multi-commodity, 8
 - supply, demand, 8
 - transshipment, 8
- storage of flow, 11, 70
- supply, 8, 11, 26
- temporally repeated flow, 12, 28, 46, 51, 64–70
 - value, 12
- time horizon, 10, 13, 26
- time-dependent flow, 25, 60–64
- time-expanded graph, 13, 58
 - holdover arc, 14, 102
- traffic models, 1, 30–34, 76–77
- transit time function, 24–26, 76–77
- transit time with delay, 108
- transit times, 76
 - concave, 52–58
 - convex, 60–67
 - inflow-dependent, *see* inflow-dependent transit times
 - load-dependent, *see* load-dependent transit times
- transshipment, 8, 11
- universally maximal flow, 20
- weakly inflow-preserving, 91, 94, 103

LEBENS LAUF

Katharina Helene Langkau

geboren am 22.3.1975 in Bonn–Bad Godesberg

- 1981 bis 1985 Besuch der Grundschule in Bonn
- 1985 bis 1994 Besuch des Cusanus–Hertz–Gymnasiums in Bonn
- 1994 Abitur
- 1994 bis 2000 Studium der Mathematik an der Rheinischen Friedrich-Wilhelms-Universität Bonn
 - 1996 Vordiplom in Mathematik mit Nebenfach Volkswirtschaftslehre
 - 2000 Diplom in Mathematik mit Nebenfach Volkswirtschaftslehre
- 1997 bis 1998 Tutorin am Mathematischen Institut der Universität Bonn
- 1998 bis 2000 Studentische Mitarbeiterin am Institut für Diskrete Mathematik, Universität Bonn
- Seit 2000 Stipendiatin im Europäischen Graduiertenkolleg “Combinatorics, Geometry, and Computation”