An Introduction to Network Flows Over Time*

Martin Skutella

Abstract

We give an introduction into the fascinating area of flows over time—also called "dynamic flows" in the literature. Starting from the early work of Ford and Fulkerson on maximum flows over time, we cover many exciting results that have been obtained over the last fifty years. One purpose of this manuscript is to serve as a possible basis for teaching network flows over time in an advanced course on combinatorial optimization.

Flow variation over time is an important feature in network flow problems arising in various applications such as road or air traffic control, production systems, communication networks (e. g., the Internet), and financial flows. In such applications, flow values on arcs are not constant but may change over time. Moreover, there is a second temporal dimension in these applications. Usually, flow does not travel instantaneously through a network but requires a certain amount of time to travel through each arc. In particular, when routing decisions are being made in one part of a network, the effects can be seen in other parts of the network only after a certain time delay. Not only the amount of flow to be transmitted but also the time needed for the transmission plays an essential role.

The above mentioned aspects of network flows are not captured by the classic *static* network flow models. This is where *network flows over time* come into play. They include a temporal dimension and therefore provide a more realistic modeling tool for numerous real-world applications. Only few textbooks on combinatorial optimization and network flows, however, mention this topic at all; see, e.g., Ford and Fulkerson [15, Chapter III.9], Ahuja, Magnanti, and Orlin [1, Chapter 19.6], Korte and Vygen [25, Chapter 9.7], and Schrijver [34, Chapter 12.5c].

The following treatment of the topic has been developed for the purpose of teaching in an advanced course on combinatorial optimization. We concentrate on flows over time (also called "dynamic flows" in the literature) with finite *time horizon* and constant capacities and constant *transit times* in a *continuous time model*. For a broader overview of flows over time we refer to the survey papers by Aronson [5], Powell, Jaillet, and Odoni [33], Kotnyek [26], and Lovetskii and Melamed [27]. We also refer to the PhD thesis of Hoppe [20] for an easily accessible and detailed treatment of the topic based on the *discrete time model*.

The paper is organized as follows. In Section 1 we introduce notation and shortly repeat some basic definitions and results from static network flow theory that are of particular importance for our purposes. For a more detailed treatment of those issues we refer to standard textbooks such as, e.g., [1] and [25]. In Section 2 we present a classical result of Ford and Fulkerson on the Maximum Flow Over Time Problem. This problem can be reduced to a static min-cost flow computation. Section 3 is devoted to a special class of flows over time called *earliest arrival flows* which can be used to model evacuation scenarios. We present a classical result that shows how to compute an earliest arrival flow with the Successive Shortest Path Algorithm. In Section 4 we consider flows over time with costs on the arcs, discuss their complexity, and introduce *time-expanded networks*. With the help of these networks, many flow over time problems can

*This work was supported by DFG Research Center MATHEON in Berlin.

be reduced to static flow problems at the cost of a considerable increase in the size of the network. Finally, in Section 5 we discuss multi-commodity flows over time and present a recent approximation result. Pointers to the literature are discussed at the end of each section. We do not claim, however, to give a complete review of all related literature. Several exercises are given in the very end.

1 Basic Notions and Results on Static Network Flows

In this section we compile basic notions and results from the area of static network flows that are needed in the remainder of the paper.

Network. Let $G = (V, E)$ be a network (directed graph) with a *source node* $s \in V$ and a *sink node* $t \in V$. Each arc $e \in E$ has an associated *capacity* u_e and a *transit time* (or *length*) $\tau_e \geq 0$. In the setting with costs, each arc e also has a *cost coefficient* c_e, which determines the cost for sending one unit of flow through the arc. An arc e from node v to node w is sometimes also denoted (v, w); in this case, we write $\text{head}(e) = w$ and $\text{tail}(e) = v$. To avoid confusion, we assume without loss of generality that there is at most one arc between any pair of nodes in G and that there are no loops.

Let $\delta^+(v)$ and $\delta^-(v)$ denote the set of arcs $e \in E$ leaving node v ($\text{tail}(e) = v$) and entering node v ($\text{head}(e) = v$), respectively. We sometimes also write $\delta^+_E(v)$ and $\delta^-_E(v)$ to emphasize the underlying set of arcs E. For a subset of nodes $X \subseteq V$, let

$$\delta^+(X) \;:=\; \{(v, w) \in E \mid v \in X \;\wedge\; w \in V \setminus X\} \;.$$

For technical reasons we assume that there is an s-v-path and a v-t-path in G for every node $v \in V$. Notice that nodes violating this condition (and their incident arcs) are useless when it comes to sending flow from s to t and can therefore be deleted.

Network flow. A (static) *network flow* x assigns a non-negative flow value x_e to each arc $e \in E$. The flow x is *feasible* if it obeys the *capacity constraints*: $x_e \leq u_e$ for each $e \in E$. An s-t-flow x satisfies *flow conservation* at each node $v \in V \setminus \{s, t\}$:

$$\sum_{e \in \delta^-(v)} x_e - \sum_{e \in \delta^+(v)} x_e = 0 \;. \tag{1}$$

The *value* $|x|$ *of an* s-t-*flow* x is

$$|x| \;:=\; \sum_{e \in \delta^-(t)} x_e - \sum_{e \in \delta^+(t)} x_e \;=\; \sum_{e \in \delta^+(s)} x_e - \sum_{e \in \delta^-(s)} x_e \;.$$

A *circulation* is a flow obeying flow conservation (1) at each node $v \in V$. The *cost* of flow x is

$$c(x) \;:=\; \sum_{e \in E} c_e \cdot x_e \;.$$

Residual network. For an arc $e = (v, w) \in E$ we denote the corresponding *backward arc* (w, v) by $\overleftarrow{e} := (w, v)$. Notice that $e \in E$ implies $\overleftarrow{e} \notin E$ due to our assumption that there is at most one arc between any pair of nodes in G. The *bidirected network* $\overleftrightarrow{G} = (V, \overleftrightarrow{E})$ corresponding to $G = (V, E)$ is defined by $\overleftrightarrow{E} := E \dot\cup \{\overleftarrow{e} \mid e \in E\}$. The transit time of a backward arc $\overleftarrow{e}$ with $e \in E$ is $\tau_{\overleftarrow{e}} := -\tau_e$. In particular, the transit time of a backward arc can be negative.

Given a feasible flow x in G, the *residual capacity of arc* $e \in E$ is $u_e - x_e$ and the *residual capacity of the corresponding backward arc* $\overleftarrow{e}$ is x_e. The *residual network* $G_x = (V, E_x)$ consists of all arcs in $\overleftrightarrow{E}$ with positive residual capacity. For $v, w \in V$ we denote the transit time of a shortest v-w-path in G_x by $\text{dist}_x(v, w)$.

Flow decomposition. Let $\mathcal{P}$ and $\mathcal{C}$ denote the sets of all simple s-t-paths and all simple cycles in G, respectively. For $P \in \mathcal{P} \cup \mathcal{C}$ we write $e \in P$ and $v \in P$ to indicate that arc $e \in E$ and node $v \in V$, respectively, lie on P. The well-known Flow Decomposition Theorem states that any static s-t-flow x has a *flow decomposition* $(x_P)_{P \in \mathcal{P} \cup \mathcal{C}}$ where $x_P \geq 0$ for each $P \in \mathcal{P} \cup \mathcal{C}$ and

$$x_e = \sum_{P \in \mathcal{P} \cup \mathcal{C} : e \in P} x_P \qquad \text{for each } e \in E.$$

Moreover, there always exists a flow decomposition for which the number of *flow-carrying* paths and cycles P with $x_P > 0$ is bounded by the number of arcs $|E|$. A flow decomposition with $x_P = 0$ for all cycles $P \in \mathcal{C}$ is also called *path decomposition*.

The set of s-t-paths in $\overleftrightarrow{G}$ is denoted by $\overleftrightarrow{\mathcal{P}}$. For a path or cycle $P \in \overleftrightarrow{\mathcal{P}} \cup \mathcal{C}$ we set $\tau(P) := \sum_{e \in P} \tau_e$. If node v is contained in an s-t-path $P \in \overleftrightarrow{\mathcal{P}}$, we denote the subpath from s to v by $P_{s,v}$ and the subpath from v to t by $P_{v,t}$.

The collection $(x_P)_{P \in \overleftrightarrow{\mathcal{P}}}$ is a *generalized path decomposition* of a given s-t-flow x if $x_P \geq 0$ for each $P \in \overleftrightarrow{\mathcal{P}}$ and

$$x_e = \sum_{P \in \overleftrightarrow{\mathcal{P}} : e \in P} x_P - \sum_{P \in \overleftrightarrow{\mathcal{P}} : \overleftarrow{e} \in P} x_P \qquad \text{for each } e \in E.$$

We mention that there are s-t-flows that do not have a generalized path decomposition since, in general, cycles are also needed in a decomposition.

Multi-commodity flow. Every commodity $i = 1, \ldots, k$ has a source node $s_i \in V$, a sink node $t_i \in V$, and a demand $d_i \geq 0$. A (static) *multi-commodity flow* x consists of k single-commodity flows x^i, $i = 1, \ldots, k$. We call x *feasible* if it satisfies the capacity constraints $\sum_{i=1}^{k} x^i_e \leq u_e$ for each $e \in E$ and if x^i is an s_i-t_i-flow for each commodity $i = 1, \ldots, k$. A feasible multi-commodity flow x *satisfies demands* $d_1, \ldots, d_k$ if $|x^i| \geq d_i$ for $i = 1, \ldots, k$.

2 Maximum Flows Over Time

We consider flows over time with a fixed time horizon $T \geq 0$.

Definition 2.1 (Flow over time)**.** A *flow over time* f *with time horizon* T consists of a Lebesgue-integrable function $f_e : [0, T) \to \mathbb{R}_{\geq 0}$ for each arc $e \in E$; moreover $f_e(\theta) = 0$ must hold for $\theta \geq T - \tau_e$. To simplify notation, we sometimes consider f_e as a function with domain $\mathbb{R}$; in this case we set $f_e(\theta) := 0$ for all $\theta \notin [0, T)$.

We say that $f_e(\theta)$ is the *rate of flow* (i.e., amount of flow per time unit) entering arc e at time θ. The flow particles entering arc e at its tail at time θ arrive at the head of e exactly τ_e time units later at time $\theta + \tau_e$. In particular, the *outflow rate* at the head of arc e at time θ equals $f_e(\theta - \tau_e)$. Definition 2.1 ensures that all flow has left arc e at time T as $f_e(\theta) = 0$ for $\theta \geq T - \tau_e$.

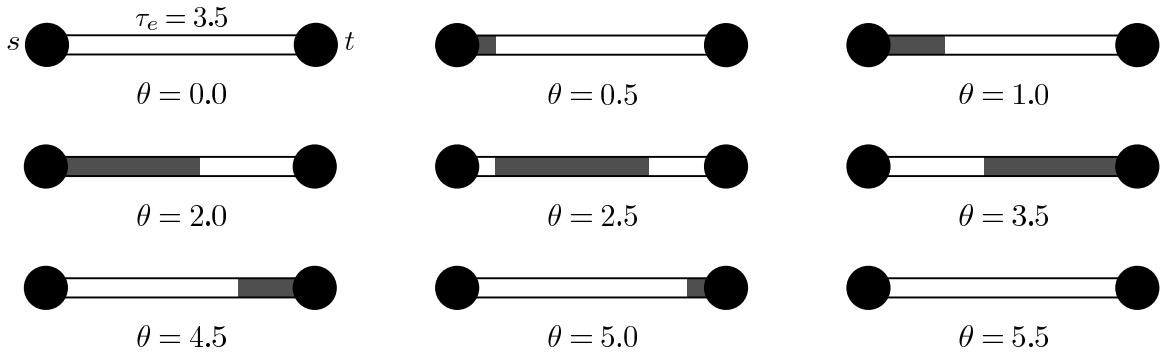

Figure 1: Snapshots of the simple flow over time discussed in Example 2.2 for several points in time. At time 0, the arc $e = (s, t)$ (directed from left to right) is empty and we start to pump flow at rate 1 into the arc. At time 0.5 we have pumped half a unit of flow into the arc, at time 2 there are two flow units and we stop pumping. The first flow particles of the first flow unit reach the head node t at time $\tau_e = 3.5$. At time 4.5 the first flow unit has arrived at node t while the second flow unit is still on the arc. Finally, at time 5.5 the last flow particles have arrived at node t and the arc is empty again.

In order to gain an intuitive understanding of flows over time, one can associate arcs of the network with pipes in a pipeline system for transporting some kind of fluid. The length of each pipeline determines the transit time of the corresponding arc while the width determines its capacity.

Example 2.2. To illustrate the described model of flows over time, we consider the following simple example with two nodes s and t that are connected by an arc $e = (s, t)$ with transit time $\tau_e = 3.5$ (see Figure 1). We can, for example, send two units of flow from s to t as follows: At time 0 we start to pump flow at rate 1 into arc e and continue to do so until time 2. More precisely, we define a flow over time f with time horizon $T := 5.5$ by

$$f_e(\theta) := \begin{cases} 1 & \text{for } \theta \in [0, 2), \\ 0 & \text{otherwise.} \end{cases}$$

The amount of flow that we have sent into arc e is obtained by integrating the flow rate f_e over time:

$$\int_0^T f_e(\theta)\, d\theta = 2 \ .$$

Since the last flow particle enters arc e shortly before time 2, it arrives at node t exactly 3.5 time units later, that is, shortly before time 5.5. More precisely, the outflow rate $f_e(\theta - \tau_e)$ of arc e at its head node t is

$$f_e(\theta - \tau_e) = \begin{cases} 1 & \text{for } \theta \in [3.5, 5.5), \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

As already mentioned above, we work with the so-called *continuous time model* where a flow over time assigns a *flow rate* $f_e(\theta)$ to every point in time θ in the interval $[0, T)$. The original definition of flows over time given by Ford and Fulkerson is based on a so-called *discrete time model*. For integral transit times $(\tau_e)_{e \in E}$ and an integral time horizon T, the flow along an arc e is described by a function $g_e : \{0, 1, 2, \ldots, T\} \to \mathbb{R}_{\geq 0}$ where $g_e(\theta)$ denotes the *amount of flow*

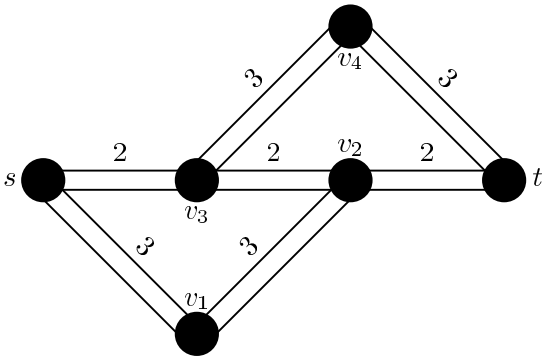

Figure 2: A network consisting of arcs (pipelines) with unit capacities. The arcs are all directed from left to right towards the sink t. The numbers at arcs indicate transit times.

sent at time θ into arc e and arriving at the head node of e at time $\theta + \tau_e$. Both models have their advantages and disadvantages. More details can be found in the references given at the end of this section.

Definition 2.3 (Capacity, excess, flow conservation, s-t-flow over time)**.** Let f be a flow over time with time horizon T.

(a) The flow over time f fulfills the *capacity constraints* (and is called *feasible*) if $f_e(\theta) \leq u_e$ for each $e \in E$ and all $\theta \in [0, T)$.

(b) For $v \in V$, the *excess for node v at time θ* is the net amount of flow that enters node v up to time θ, that is,

$$\mathrm{ex}_f(v, \theta) \;:=\; \sum_{e \in \delta^-(v)} \int_0^{\theta - \tau_e} f_e(\xi)\, d\xi \;-\; \sum_{e \in \delta^+(v)} \int_0^{\theta} f_e(\xi)\, d\xi \;\;.$$

(c) The flow over time f fulfills the *weak flow conservation constraints* if $\mathrm{ex}_f(v, \theta) \geq 0$ for each $v \in V \setminus \{s\}$ and all $\theta \in [0, T)$. Moreover, $\mathrm{ex}_f(v, T) = 0$ must hold for each $v \in V \setminus \{s, t\}$.

(d) A flow over time obeying the weak flow conservation constraints is an *s-t-flow over time*. The *value* of an s-t-flow over time with time horizon T is $|f| := \mathrm{ex}_f(t, T)$.

(e) An s-t-flow over time f fulfills the *strict flow conservation constraints* if $\mathrm{ex}_f(v, \theta) = 0$ for all $v \in V \setminus \{s, t\}$ and $\theta \in [0, T]$. The strict flow conservation constraints say that flow must not be stored at intermediate nodes.

Notice that the *weak* flow conservation constraints allow to store flow at intermediate nodes for some time as long as the flow has left the node again before the time horizon is over. In contrast to this, the *strict* flow conservation constraints ensure that flow entering an intermediate node must leave the node again *immediately*.

In Figure 2 we give an example of a network. An illustration of a feasible s-t-flow over time in this network fulfilling the strict flow conservation constraints is given in Figure 3.

We study the following basic optimization problem for flows over time.

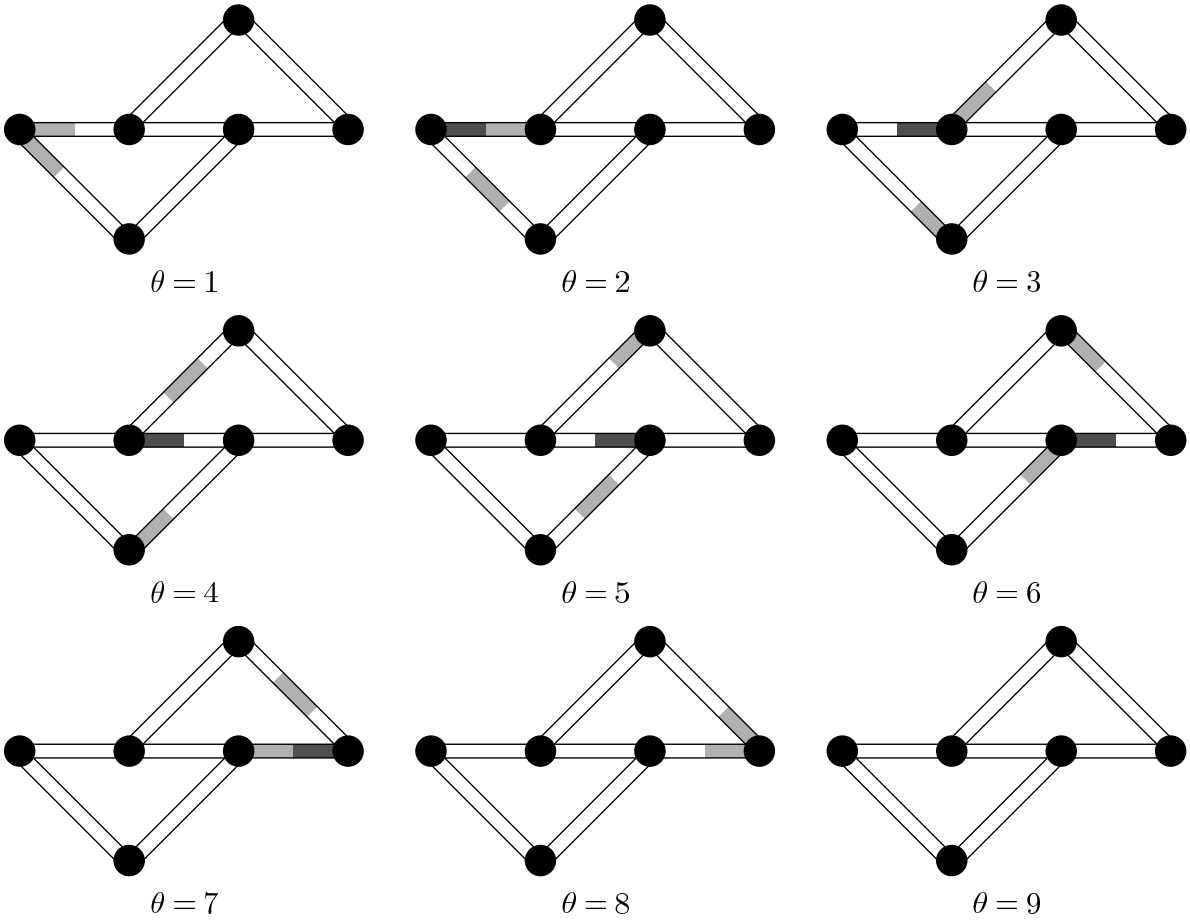

Figure 3: Snapshots of a feasible s-t-flow over time with time horizon $T = 9$ and value 3 in the network depicted in Figure 2. In order to distinguish flow units traveling one after another along an arc, different shadings are used for the flow units.

Maximum Flow Over Time Problem
Given: A network $G = (V, E)$ with capacities and transit times on the arcs, a source node $s \in V$, a sink node $t \in V$, and a time horizon $T \geq 0$.
Task: Find a feasible s-t-flow over time f with time horizon T and maximum value $\|f\|$.

We consider a special class of feasible s-t-flows over time that are induced by static s-t-flows and feature a very simple structure. The intuition behind the following definition is to take a path decomposition (x_P) of a static s-t-flow x. At time 0 the resulting s-t-flow over time starts to send flow on each s-t-path P with flow rate x_P. It keeps sending flow along each path as long as there is enough time left for the flow along the path to arrive at the sink by time T. We first give a formal definition and afterwards, in Observation 2.5, a precise interpretation and intuition.

Definition 2.4 (Temporally repeated flow)**.** Let x be a static s-t-flow with some flow decomposition $(x_P)_{P \in \mathcal{P} \cup \mathcal{C}}$. The corresponding *temporally repeated flow* f with time horizon T is defined

by

$$f_e(\theta) \;:=\; \sum_{P \in \mathcal{P}_e(\theta)} x_P \qquad\qquad \text{for } e=(v,w) \in E,\ \theta \in [0,T), \tag{3}$$

where

$$\mathcal{P}_e(\theta) \;:=\; \{P \in \mathcal{P} : e \in P \wedge \tau(P_{s,v}) \leq \theta \wedge \tau(P_{v,t}) < T-\theta\}\ .$$

Notice that the flow x_P on path $P \in \mathcal{P}$ contributes to the flow rate f_e on arc $e=(v,w) \in P$ within the time interval starting at time $\tau(P_{s,v})$ and ending before time $T-\tau(P_{v,t}) = T-\tau(P)+\tau(P_{s,v})$. Thus, an alternative "path-based" description of the temporally repeated flow f is as follows.

Observation 2.5. The temporally repeated flow f in Definition 2.4 can be obtained as follows: For each path $P \in \mathcal{P}$, send flow at rate x_P into P from the source s during the time interval $[0, T-\tau(P))$ and let the flow progress towards the sink without any delay at intermediate nodes. In particular, all flow reaches the sink by time T.

Notice that the s-t-flow over time depicted in Figure 3 is not temporally repeated since it does not have the structure described in Observation 2.5. This can, for example, be seen as follows. The flow over time in Figure 3 sends flow along the shortest s-t path of length 6. However, it only starts to send flow into this path at time 1 and stops again at time 2 in order to avoid collisions with the remaining two flow units traveling along paths of length 8. In contrast, a temporally repeated flow of value 3 and time horizon 9 on this network sends all 3 units of flow along the shortest path. That is, it starts to send flow at rate 1 into this path at time 0 and stops before time 3 such that the last flow particles arrive at the sink before time 9.

Lemma 2.6. *Let x be a feasible static s-t-flow with flow decomposition $(x_P)_{P \in \mathcal{P} \cup \mathcal{C}}$. Then the corresponding temporally repeated flow f is a feasible s-t-flow over time with time horizon T that satisfies the strict flow conservation constraints.*

Proof. By construction (see Observation 2.5), flow entering an intermediate node leaves this node again immediately. Thus f is an s-t-flow over time with time horizon T that satisfies the strict flow conservation constraints. By (3) the feasibility of x implies

$$f_e(\theta) \;\leq\; \sum_{P \in \mathcal{P}: e \in P} x_P \;\leq\; x_e \;\leq\; u_e \qquad\qquad \text{for } e \in E,\ \theta \in [0,T).$$

Thus f is feasible. □

The next lemma gives an indication how the static s-t-flow x should be chosen to get a temporally repeated flow with large value.

Lemma 2.7. *Let x be a feasible static s-t-flow with flow decomposition $(x_P)_{P \in \mathcal{P} \cup \mathcal{C}}$ such that $x_P = 0$ for all $P \in \mathcal{P}$ with $\tau(P) > T$ and for all $P \in \mathcal{C}$. Then the value of the corresponding temporally repeated flow f is equal to*

$$|f| \;=\; T \cdot |x| - \sum_{e \in E} \tau_e \cdot x_e\ .$$

In particular, the value of f does not depend on the chosen path decomposition of x.

Proof. By Observation 2.5 we get

$$\begin{aligned} |f| &= \sum_{P\in\mathcal{P}} (T-\tau(P))\cdot x_P \\ &= T\cdot\sum_{P\in\mathcal{P}} x_P - \sum_{P\in\mathcal{P}}\sum_{e\in P}\tau_e\cdot x_P \\ &= T\cdot|x| - \sum_{e\in E}\tau_e\cdot\sum_{P\in\mathcal{P}:e\in P} x_P \\ &= T\cdot|x| - \sum_{e\in E}\tau_e\cdot x_e \ . \end{aligned}$$

This concludes the proof. □

We mention the following corollary only for later use in Section 5.

Corollary 2.8. *Let x be a feasible static s-t-flow with flow decomposition $(x_P)_{P\in\mathcal{P}\cup\mathcal{C}}$. Then the value of the corresponding temporally repeated flow f is at least*

$$|f| \ \geq\ T\cdot|x| - \sum_{e\in E}\tau_e\cdot x_e \ .$$

Proof. We modify x by deleting flow on all paths $P\in\mathcal{P}$ with $\tau(P)>T$ and on all cycles $P\in\mathcal{C}$. More precisely, we set

$$\tilde{x}_P \ :=\ \begin{cases} x_P & \text{if } P\in\mathcal{P} \text{ and } \tau(P)\leq T, \\ 0 & \text{otherwise,} \end{cases} \qquad \text{for } P\in\mathcal{P}\cup\mathcal{C}.$$

By Observation 2.5, the temporally repeated flows f and $\tilde{f}$ corresponding to $(x_P)_{P\in\mathcal{P}\cup\mathcal{C}}$ and $(\tilde{x}_P)_{P\in\mathcal{P}\cup\mathcal{C}}$, respectively, are identical. Moreover, since flow along cycles and flow along s-t-paths of length at least T make a non-positive contribution to

$$T\cdot|x| - \sum_{e\in E}\tau_e\cdot x_e \ ,$$

we get by Lemma 2.7

$$|f| \ =\ |\tilde{f}| \ =\ T\cdot|\tilde{x}| - \sum_{e\in E}\tau_e\cdot\tilde{x}_e \ \geq\ T\cdot|x| - \sum_{e\in E}\tau_e\cdot x_e \ .$$

This concludes the proof. □

As a consequence of Lemma 2.7, a maximum temporally repeated flow can be obtained as follows.

FORD-FULKERSON ALGORITHM

Input: A network $G = (V, E)$ with capacities and transit times on the arcs, a source node $s \in V$, a sink node $t \in V$, and a time horizon $T \geq 0$.

Output: A temporally repeated flow with time horizon T.

1. Compute a feasible static s-t-flow x maximizing
$$T \cdot |x| - \sum_{e \in E} \tau_e \cdot x_e \ .$$
2. Compute a flow decomposition $(x_P)_{P \in \mathcal{P} \cup \mathcal{C}}$ of x.
3. Output the corresponding temporally repeated flow f.

We next discuss how to obtain the static s-t-flow in Step 1 of the Ford-Fulkerson Algorithm. We consider an *extended network* $G' = (V, E')$ where the set of arcs E' is obtained by adding the *artificial arc* (t, s) to E with $u_{(t,s)} := \infty$ and $\tau_{(t,s)} := -T$. Any static s-t-flow x in G naturally induces a static circulation in G' by assigning flow value $|x|$ to the artificial arc (t, s). Conversely, any static circulation in G' naturally induces a static s-t-flow x in G whose value $|x|$ is equal to the flow value on the artificial arc (t, s). If we interpret transit times as cost coefficients ($c_e := \tau_e$ for $e \in E'$), the problem of finding a static s-t-flow x maximizing

$$T \cdot |x| - \sum_{e \in E} \tau_e \cdot x_e$$

is equivalent to finding such a flow minimizing the negated objective

$$-T \cdot |x| + \sum_{e \in E} \tau_e \cdot x_e \ = \ c_{(t,s)} \cdot |x| + \sum_{e \in E} c_e \cdot x_e \ .$$

The latter problem is apparently a static minimum cost circulation problem on the extended network G'.

Observation 2.9. The static s-t-flow in Step 1 of the Ford-Fulkerson Algorithm can be obtained from a static minimum cost circulation computation in the extended network $G' = (V, E')$.

As a consequence of this interpretation, we obtain the following insights on the path decomposition computed in Step 2.

Lemma 2.10. *Take any flow decomposition $(x_P)_{P \in \mathcal{P} \cup \mathcal{C}}$ computed in Step 2 of the Ford-Fulkerson Algorithm. Then the following holds for all $P \in \mathcal{P} \cup \mathcal{C}$:*

$$\text{If } x_P > 0 \ , \quad \text{then} \quad \tau(P) \begin{cases} = 0 & \text{if } P \in \mathcal{C}, \\ \leq T & \text{if } P \in \mathcal{P}. \end{cases} \tag{4}$$

In particular, all flow along cycles in x can be canceled without changing the objective function value of x. The resulting s-t-flow x is still feasible and optimal and satisfies the requirements of Lemma 2.7.

Proof. Notice that a path or cycle $P \in \mathcal{P} \cup \mathcal{C}$ violating (4) yields a negative cost cycle in the residual network G'_x. This contradicts the optimality of x. □

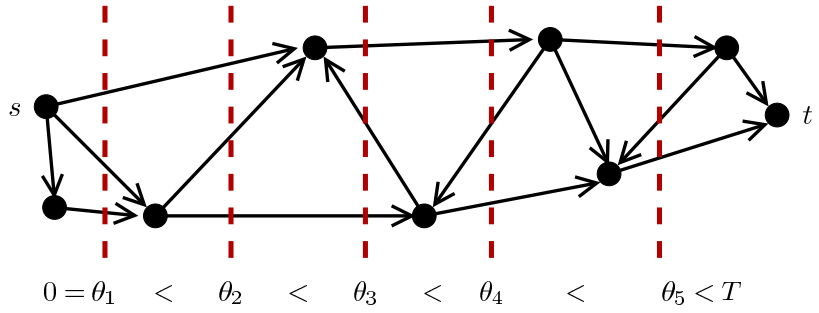

Figure 4: An s-t-cut over time moves through the network over time from the source towards the sink.

Since the static s-t-flow x computed in Step 1 satisfies the requirements of Lemma 2.7, the flow value of the temporally repeated flow equals the optimum objective function value of x.

Corollary 2.11. *Let x be the static s-t-flow computed in Step 1 of the Ford-Fulkerson Algorithm and f the temporally repeated flow obtained in Step 3. Then*

$$|f| \;=\; T \cdot |x| - \sum_{e \in E} \tau_e \cdot x_e \;\; .$$

We conclude from Corollary 2.11 that the s-t-flow over time f computed by the Ford-Fulkerson Algorithm has maximum value among all temporally repeated flows. The following theorem states that it is even a maximum s-t-flow over time.

Theorem 2.12. *The temporally repeated flow computed by the Ford-Fulkerson Algorithm is a maximum s-t-flow over time with time horizon T. The running time of the algorithm is dominated by the static min-cost flow computation in Step 1.*

A proof can be found below. As a result of this theorem we know that computing a maximum s-t-flow over time is at most as difficult as computing a (static) min-cost flow.

The proof of the theorem goes along the same lines as the proof of the corresponding theorem for maximum static s-t-flows. The idea is to come up with an s-t-cut whose capacity is an upper bound on the maximum flow value and matches the value of the computed flow. In the static setting, an s-t-cut is defined by a subset of nodes $X \subseteq V$ with $s \in X$ and $t \notin X$. It consists of the arcs in $\delta^+(X)$ and its capacity is $\sum_{e \in \delta^+(X)} u_e$. For the case of flows over time, however, a more elaborate definition of an s-t-cut and its capacity is required in order to find a tight upper bound on the maximum flow value (see also Exercise 1).

Definition 2.13 (s-t-cut over time)**.** An *s-t-cut over time with time horizon T* is specified by threshold values $\alpha_v \in \mathbb{R}$ for each $v \in V$ with $\alpha_s = 0$ and $\alpha_t \geq T$. The *capacity of an s-t-cut over time* is defined as

$$\sum_{e=(v,w) \in E} \max\{0, \alpha_w - \tau_e - \alpha_v\} \cdot u_e \;\; .$$

We say that node $v \in V$ belongs to the *t-side of the s-t-cut over time* up to time α_v and to the *s-side of the s-t-cut over time* from time α_v on. Thus, we can think of the s-t-cut over time as a sequence of s-t-cuts moving through the network over time towards the sink. An illustration is given in Figure 4.

The definition of the capacity of an s-t-cut over time is motivated as follows: Every flow particle moving from s to t within the time interval $[0, T)$ will eventually cross the considered

s-t-cut, i.e., move from the s-side to the t-side of the cut. A particle moving along some arc $e = (v, w)$ crosses the cut if it enters the arc at node v while v is on the s-side of the cut and leaves the arc at node w while w is on the t-side of the cut. Thus, any particle entering arc e within the *critical time interval* $[\alpha_v, \alpha_w - \tau_e)$ crosses the cut while traveling along this arc. The total amount of flow that can cross the cut on arc e is thus bounded by u_e times the size of the critical time interval. The capacity of the s-t-cut over time is the sum of those values.

This motivates the following result.

Lemma 2.14. *The capacity of an s-t-cut over time with time horizon T is an upper bound on the value of any feasible s-t-flow over time with time horizon T.*

Proof. Let f be a feasible s-t-flow over time with time horizon T and let $(\alpha_v)_{v \in V}$ define an s-t-cut over time with time horizon T. By Definition 2.3 we get

$$|f| \;=\; \mathrm{ex}_f(t, \alpha_t) \;\leq\; \sum_{v \in V} \mathrm{ex}_f(v, \alpha_v) \;=\; \sum_{v \in V} \left(\sum_{e \in \delta^-(v)} \int_0^{\alpha_v - \tau_e} f_e(\theta)\, d\theta - \sum_{e \in \delta^+(v)} \int_0^{\alpha_v} f_e(\theta)\, d\theta \right) \;.$$

For the inequality we also use the fact that $\mathrm{ex}_f(s, 0) = 0$. Exchanging the order of summation in the right hand side term yields

$$\begin{aligned} |f| \;&\leq\; \sum_{e=(v,w) \in E} \left(\int_0^{\alpha_w - \tau_e} f_e(\theta)\, d\theta - \int_0^{\alpha_v} f_e(\theta)\, d\theta \right) \\ &=\; \sum_{e=(v,w) \in E} \int_{\alpha_v}^{\alpha_w - \tau_e} f_e(\theta)\, d\theta \\ &\leq\; \sum_{e=(v,w) \in E} \max\{0, \alpha_w - \tau_e - \alpha_v\} \cdot u_e \;. \end{aligned}$$

This concludes the proof. □

We are now ready to prove Theorem 2.12.

Proof of Theorem 2.12. It follows from Observation 2.9 and Corollary 2.11 that the value of the temporally repeated flow computed by the Ford-Fulkerson Algorithm is equal to the optimum solution value of the following linear programming formulation of a min-cost circulation problem on the extended network $G' = (V, E')$.

$$\begin{aligned} \max\; & T \cdot x_{(t,s)} - \sum_{e \in E} \tau_e \cdot x_e & & \\ \text{s.t.}\; & \sum_{e \in \delta^+_{E'}(v)} x_e - \sum_{e \in \delta^-_{E'}(v)} x_e = 0 & & \text{for all } v \in V, \\ & x_e \leq u_e & & \text{for all } e \in E, \\ & x_e \geq 0 & & \text{for all } e \in E'. \end{aligned}$$

The dual linear program looks as follows.

$$\begin{aligned} \min\; & \sum_{e \in E} u_e y_e & & \\ \text{s.t.}\; & y_e + \alpha_v - \alpha_w \geq -\tau_e & & \text{for all } e = (v, w) \in E, \\ & \alpha_t - \alpha_s \geq T & & \\ & y_e \geq 0 & & \text{for all } e \in E. \end{aligned}$$

Let $(\alpha_v)_{v\in V}, (y_e)_{e\in E}$ be an optimum dual solution. We can assume without loss of generality that $\alpha_s = 0$ (otherwise, replace α_v with $\alpha_v - \alpha_s$ for each $v \in V$). Thus, the values $(\alpha_v)_{v\in V}$ define an s-t-cut over time. Moreover, for an optimum solution it holds that $y_e := \max\{0, \alpha_w - \tau_e - \alpha_v\}$. Therefore the optimum dual solution value equals the capacity of the cut defined by the α_v's. On the other hand, the value of the temporally repeated flow computed by the Ford-Fulkerson Algorithm equals the optimum primal solution value. The result follows from strong linear programming duality. □

Corollary 2.15. *An s-t-cut over time with time horizon T and minimum capacity can be obtained by a static min-cost flow computation.*

Proof. We consider again the pair of linear programs in the proof of Theorem 2.12. It is well-known (and easy to observe using complementary slackness) that an optimum dual solution can be obtained from an optimum primal solution (min-cost circulation) x as follows. For each $v \in V$, let α_v be the length of a shortest s-v-path in the residual network G'_x. □

As another corollary we get the following Max-Flow-Min-Cut Theorem for flows and cuts over time.

Theorem 2.16 (Max-Flow-Min-Cut Theorem)**.** *The maximum value of an s-t-flow over time with time horizon T equals the minimum capacity of an s-t-cut over time with time horizon T.*

Pointers to the Literature

The results on the Maximum Flow Over Time Problem presented above are due to Ford and Fulkerson [14, 15]. More precisely, these results were originally developed for the discrete time model that we already shortly mentioned above. That is, time is discretized into steps of unit length. In each time step, flow can be sent from a node v through an arc (v, w) to the adjacent node w, where it arrives $\tau_{(v,w)}$ time steps later. In particular, the time-dependent flow on an arc is represented by a time-indexed vector in this model. Fleischer and Tardos [13] point out a strong connection between the two models. They show that many results and algorithms which have been developed for the discrete time model can be carried over to the continuous time model. The continuous time version of Theorem 2.12 has first been observed by Anderson and Philpott [4].

The concept of s-t-cuts over time was introduced by Anderson, Nash, and Philpott [3] (see also [2]) for the case of zero transit times and later extended to arbitrary transit times by Philpott [32]. In particular, Lemma 2.14 is due to [3, 32].

A problem closely related to the Maximum Flow Over Time Problem is the Quickest s-t-Flow Problem: Send a given amount of flow from the source to the sink in the shortest possible time. This problem can be solved in polynomial time by incorporating the Ford-Fulkerson Algorithm in a binary search framework; see also [13]. Using Megiddo's method of parametric search [30], Burkard, Dlaska, and Klinz [8] present a faster algorithm which solves the Quickest s-t-Flow Problem in strongly polynomial time.

An interesting generalization of the Quickest s-t-Flow Problem is the Quickest Transshipment Problem: Given a vector of supplies and demands at the nodes, the task is to find a flow over time that satisfies all supplies and demands within minimal time. Unlike the situation for standard (static) network flow problems, this multiple source, multiple sink, single commodity flow over time problem is not equivalent to a maximum s-t-flow over time problem (see Exercise 3). Hoppe and Tardos describe the first polynomial-time algorithm to solve this problem [22, 20]. They introduce a generalized class of temporally repeated flows which can also be compactly encoded

as a collection of paths. However, in contrast to temporally repeated flows, these paths may also contain backward arcs. Therefore, a careful analysis is necessary to show feasibility of the resulting flows over time (see also Section 3). Moreover, the algorithm of Hoppe and Tardos is not practical as it requires a submodular function minimization oracle for a subroutine.

3 Earliest Arrival Flows

In the last section we have considered a given, fixed time horizon T and the problem to send as much flow as possible from source s to sink t by time T. A possible application scenario for this problem is evacuation planning where one wants to get as many people as possible out of an endangered building or area. Since it is usually not clear a priori how long a building can withstand a fire before it collapses or how long a dam can resist a flood before it breaks, the exact time horizon T is not known in such a setting. It is therefore advisable to organize an evacuation such that as much as possible is saved no matter when the catastrophe will actually happen.

Coined in terms of s-t-flows over time, the goal is to find a single s-t-flow over time that simultaneously maximizes the amount of flow reaching the sink t up to any time $\theta \geq 0$.

Definition 3.1 (Earliest arrival flow)**.** A feasible s-t-flow over time f with time horizon T has the *earliest arrival property* and is called *earliest arrival flow* if it maximizes $\mathrm{ex}_f(t,\theta)$ simultaneously for all $\theta \in [0,T]$.

At first sight, the existence of earliest arrival flows is not evident. One can indeed show that the important class of temporally repeated flows does, in general, not contain an earliest arrival flow (see Exercise 4). The situation changes if we consider a slightly more general class of s-t-flows over time.

Definition 3.2 (Generalized temporally repeated flow)**.** Let x be a static s-t-flow with generalized path decomposition $(x_P)_{P\in\overleftrightarrow{\mathcal{P}}}$. The corresponding *generalized temporally repeated flow* f with time horizon T is defined by

$$f_e(\theta) := \sum_{P\in\overleftrightarrow{\mathcal{P}}_e(\theta)} x_P - \sum_{P\in\overleftrightarrow{\mathcal{P}}_{\overleftarrow{e}}(\theta)} x_P \qquad \text{for each } e=(v,w)\in E,\ \theta\in[0,T), \tag{5}$$

where

$$\overleftrightarrow{\mathcal{P}}_e(\theta) := \{P\in\overleftrightarrow{\mathcal{P}} \mid e\in P \wedge \tau(P_{s,v}) \leq \theta \wedge \tau(P_{v,t}) < T-\theta\}$$

and analogously

$$\overleftrightarrow{\mathcal{P}}_{\overleftarrow{e}}(\theta) := \{P\in\overleftrightarrow{\mathcal{P}} \mid \overleftarrow{e}\in P \wedge \tau(P_{s,v}) \leq \theta \wedge \tau(P_{v,t}) < T-\theta\}\ .$$

Notice that a generalized temporally repeated flow f is not necessarily a proper flow over time since $f_e(\theta)$ can, in general, be negative. More precisely, and in analogy to Observation 2.5, f can be interpreted as follows.

Observation 3.3. The generalized temporally repeated flow f in Definition 3.2 can be obtained as follows: For each path $P\in\overleftrightarrow{\mathcal{P}}$, send flow into P at rate x_P during the time interval $[0,T-\tau(P))$ and let the flow progress towards the sink without any delay at intermediate nodes. In particular, if some path P contains a backward arc $\overleftarrow{e}=(w,v)$ with negative transit time $-\tau_e$, flow traveling along arc $\overleftarrow{e}$ goes back in time. A flow particle entering $\overleftarrow{e}$ in node w at time θ arrives in node

v at time $\theta - \tau_e$. In (5) this anomaly is captured by a negative flow rate $f_e(\theta - \tau_e)$ on the corresponding forward arc e.

Under certain circumstances, however, going back in time can be justified. If there is another flow particle at node v that is about to enter arc e at time $\theta - \tau_e$, the two particles traveling along e in opposite (also with respect to time) directions cancel out, that is, they exchange their identity. In (5) this is reflected by the fact that the positive terms in the first sum compensate for a negative contribution of the second sum.

The following example illustrates the intuition behind Observation 3.3.

Example 3.4. Consider again the network depicted in Figure 2. Let x be the feasible s-t-flow that sends one unit of flow across each arc except (v_3, v_2) where the flow value is zero. A generalized path decomposition of x is obtained by sending one unit of flow on the shortest s-t-path $P^1 = s, v_3, v_2, t$ and one unit along the path $P^2 = s, v_1, v_2, v_3, v_4, t$. The corresponding generalized temporally repeated flow f with time horizon $T = 11$ is illustrated in Figure 5. Since $\tau(P^1) = 6$, the flow over time f sends $5 = 11 - 6$ units of flow through path P^1 (dark flow in Figure 5). Moreover, since $\tau(P^2) = 10$, it sends $1 = 11 - 10$ flow units through path P^2 (light flow in Figure 5). As can be seen in Figure 5, f is a feasible s-t-flow over time with time horizon $T = 11$ although the light flow unit goes back in time when traveling through the backward arc (v_2, v_3) with transit time -2. As a result, we see two copies of this flow unit in the network at times $\theta = 5$ and $\theta = 6$. Since the light flow unit and the third dark flow unit cancel out on arc (v_3, v_2), this dark flow unit has disappeared at times $\theta = 5$ and $\theta = 6$. It reappears at time 7 on arc (v_2, t) when the light flow unit has arrived at node v_2.

In the following we interpret transit times also as cost coefficients, that is, we set $c_e := \tau_e$ for each $e \in E$. In particular, $\text{dist}_x(v, w)$ refers to the minimum transit time of a v-w-path in the residual network G_x. Notice that the two paths P^1 and P^2 considered in Example 3.4 are exactly the augmenting paths chosen by the Successive Shortest Path Algorithm. Moreover, it can be easily checked that the resulting s-t-flow over time f is an earliest arrival flow. We show that the augmenting paths chosen by the Successive Shortest Path Algorithm always yield a generalized temporally repeated flow that is feasible and has the earliest arrival property.

EARLIEST ARRIVAL ALGORITHM

Input: A network $G = (V, E)$ with capacities and transit times on the arcs, a source node $s \in V$, a sink node $t \in V$, and a time horizon $T \geq 0$.

Output: A generalized temporally repeated flow with time horizon T.

1. Let $x_P := 0$ for all $P \in \overleftrightarrow{\mathcal{P}}$ and let x denote the static s-t-flow with generalized path decomposition $(x_P)_{P \in \overleftrightarrow{\mathcal{P}}}$;
2. While $\text{dist}_x(s, t) < T$, find a shortest s-t-path P in G_x and increase x_P by the residual capacity of P.
3. Output the generalized temporally repeated flow f with time horizon T corresponding to $(x_P)_{P \in \overleftrightarrow{\mathcal{P}}}$.

We assume that Step 2 of the Earliest Arrival Algorithm terminates after q iterations. For $i = 1, \ldots, q$ let x^i denote the feasible s-t-flow x before iteration i and let P^i be the shortest path found in iteration i. In particular, $(x_{P^i})_{i=1,\ldots,k}$ is a generalized path decomposition of x^{k+1}.

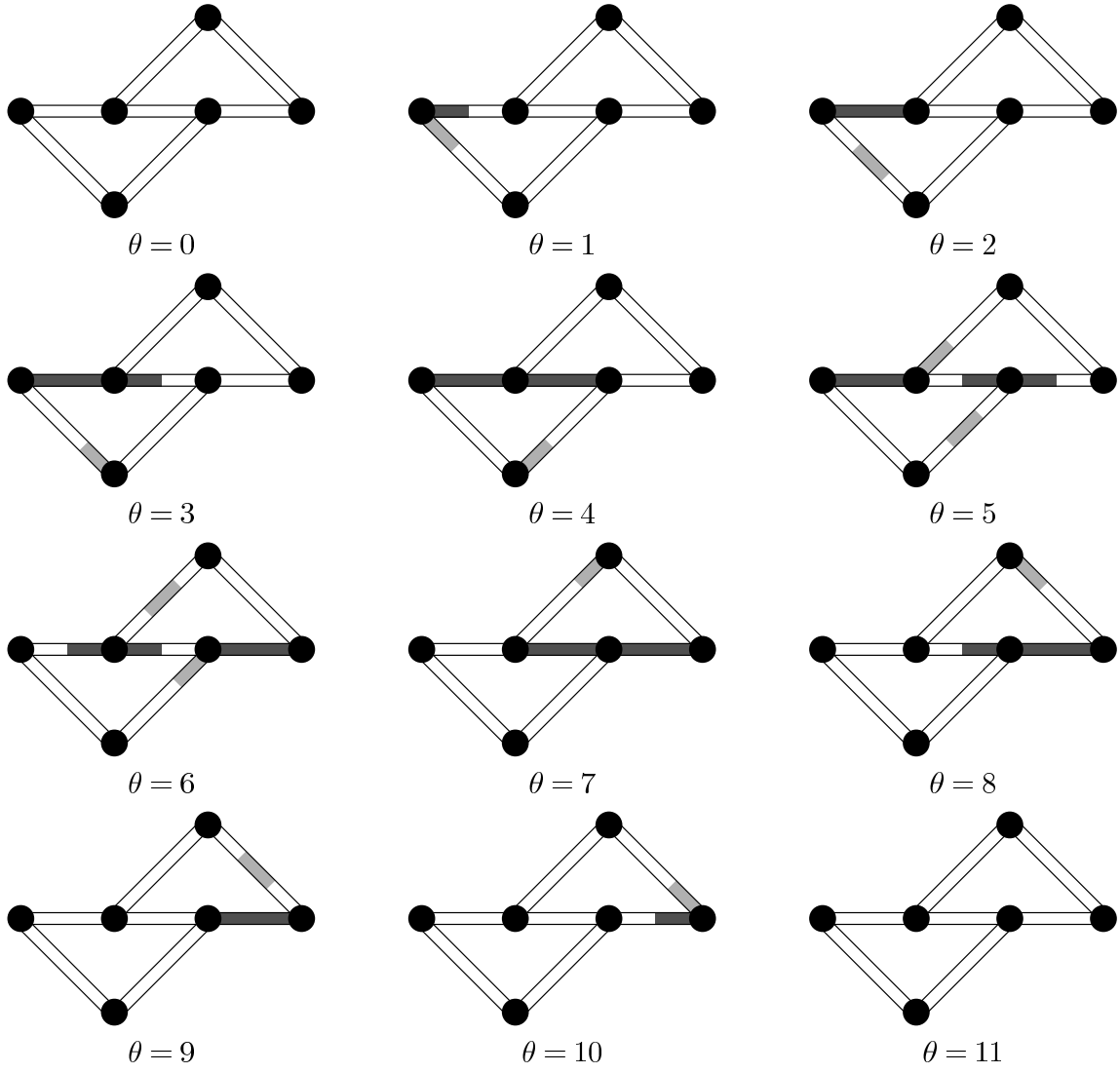

Figure 5: Snapshots of the generalized temporally repeated flow f with time horizon $T = 11$ described in Example 3.4.

Since P^i is a shortest s-t-path in the residual network G_{x^i}, we get for each node $v \in P^i$

$$\tau(P^i_{s,v}) \;=\; \mathrm{dist}_{x^i}(s,v) \qquad \text{and} \qquad \tau(P^i_{v,t}) \;=\; \mathrm{dist}_{x^i}(v,t)\ .$$

In order to show that f is a feasible flow over time, we need the following result from the theory of static network flows.

Lemma 3.5. *The values* $\mathrm{dist}_{x^i}(s,v)$ *and* $\mathrm{dist}_{x^i}(v,t)$ *are monotonically increasing in* i.

Sketch of proof. It is well-known (see, e.g., [1, Chapter 9.7] or [25, Chapter 9.4]) that the flows x^i are min-cost flows in G. A certificate of optimality for x^i is the feasible node potential $\pi_i(v) := \mathrm{dist}_{x^i}(s,v)$ (remember that distances are measured with respect to transit times of arcs). After augmenting flow along the shortest s-t-path P^i in G_{x^i}, the node potential $\pi_i(v)$ is still feasible. This implies that the shortest path distances have not decreased. □

We show that the flow over time computed by the Earliest Arrival Algorithm is feasible.

Lemma 3.6. *The generalized temporally repeated flow f computed by the Earliest Arrival Algorithm is a feasible s-t-flow over time with time horizon T.*

Proof. We first show that f is indeed a feasible flow over time, i.e., $0 \leq f_e(\theta) \leq u_e$ for $e = (v,w) \in E$, $\theta \in [0,T)$. For fixed e and θ, let

$$k \;:=\; \max\{i \mid \mathrm{dist}_{x^i}(s,v) \leq \theta \wedge \mathrm{dist}_{x^i}(v,t) < T - \theta\} \;.$$

By the definition of f in (5) and by Lemma 3.5 we get

$$\begin{aligned} f_e(\theta) \;&=\; \sum_{P \in \overleftrightarrow{\mathcal{P}}_e(\theta)} x_P - \sum_{P \in \overleftrightarrow{\mathcal{P}}_{\bar{e}}(\theta)} x_P \\ &=\; \sum_{i \in \{1,\ldots,k\}: e \in P^i} x_{P^i} - \sum_{i \in \{1,\ldots,k\}: \overleftarrow{e} \in P^i} x_{P^i} \\ &=\; x^{k+1}(e) \;\in\; [0, u_e] \;. \end{aligned}$$

Finally, it follows from Observation 3.3 that f fulfills strict flow conservation and has time horizon T. □

Theorem 3.7. *The generalized temporally repeated flow f computed by the Earliest Arrival Algorithm is an earliest arrival flow with time horizon T.*

Proof. We prove that $\mathrm{ex}_f(t,\theta)$ is maximal for all $\theta \in [0,T]$. For a fixed θ let

$$k \;:=\; \max\{i \mid \mathrm{dist}_{x^i}(s,t) \leq \theta\} \;.$$

Since $\tau(P^i) = \mathrm{dist}_{x^i}(s,t)$ is monotonically increasing in i, Observation 3.3 yields

$$\mathrm{ex}_f(t,\theta) \;=\; \sum_{i=1}^{k} (\theta - \tau(P^i)) \cdot x_{P^i} \;=\; \theta \cdot |x^{k+1}| - \sum_{e \in E} \tau_e \cdot x_e^{k+1} \;.$$

By our choice of k, the s-t-flow x^{k+1} is not only a min-cost s-t-flow in G (with respect to cost coefficients $c_e := \tau_e$ for each $e \in E$) but also induces a min-cost circulation in the extended network that is obtained by adding an artificial arc (t,s) with transit time $\tau_{(t,s)} := -\theta$ (see also Section 2). The artificial arc (t,s) and its backward arc do not induce a negative cycle in the residual graph. The reason is that by our choice of k the length of each flow-carrying s-t-path in x^{k+1} is bounded by θ; moreover, the length of each s-t-path in $G_{x^{k+1}}$ is greater than θ. Therefore x^{k+1} maximizes the objective function

$$\theta \cdot |x| - \sum_{e \in E} \tau_e \cdot x_e$$

and $\mathrm{ex}_f(t,\theta)$ is therefore maximal (see Section 2). □

Notice that the running time of the Earliest Arrival Algorithm is not polynomially bounded in the input size as the Successive Shortest Path Algorithm in Step 2 requires an exponential number of iterations in the worst case (also see references below). As a consequence, the function $\theta \mapsto \mathrm{ex}_f(t,\theta)$ can have exponentially many breakpoints for an earliest arrival flow f. It is therefore unlikely that an algorithm exists that computes f and whose running time is polynomially bounded in the input size.

Pointers to the Literature

Shortly after Ford and Fulkerson introduce flows over time, Gale [16] shows that earliest arrival s-t-flows always exist (they are also called "universally maximum dynamic flows" or "universally quickest flow" in the literature). The Earliest Arrival Algorithm is due to Minieka [31] and Wilkinson [36]. While Gale, Wilkinson, and Minieka all work in the discrete time model, the existence of earliest arrival flows in the continuous time model is first observed by Philpott [32]. Fleischer and Tardos [13] finally discuss the Earliest Arrival Algorithm in the continuous time setting.

Zadeh [37] presents a class of instances for which the Successive Shortest Path Algorithm and thus also the Earliest Arrival Algorithm requires an exponential number of iterations. Those instances also show that the piece-wise linear and convex function $\theta \mapsto \mathrm{ex}_f(t,\theta)$ has exponentially many breakpoints in the worst case.

Hoppe and Tardos [21] present a fully polynomial-time approximation scheme for the earliest arrival s-t-flow problem that is based on a clever scaling trick. For a fixed $\epsilon > 0$, the computed s-t-flow over time f has the following property. For all θ, the excess $\mathrm{ex}_f(t,\theta)$ is at least $(1-\epsilon)$ times the value of a maximum s-t-flow over time with time horizon θ.

In a network with several sources and sinks with given supplies and demands, flows over time having the earliest arrival property do not necessarily exist [12] (see Exercise 6). For the case of several sources with given supplies and a single sink, however, earliest arrival transshipments do always exist. This follows, for example, from the existence of lexicographically maximal flows in time-expanded networks; see, e.g., [31] and [29]. Hajek and Ogier [17] give the first polynomial time algorithm for computing earliest arrival flows in networks with several sources and zero transit times. Fleischer [12] gives an algorithm with improved running time.

Baumann and Skutella [6] give an algorithm that computes earliest arrival flows for the case of several sources and arbitrary transit times and whose running time is polynomially bounded in the input plus output size. Fleischer and Skutella [11] use condensed time-expanded networks to approximate such earliest arrival flows. They give a fully polynomial-time approximation scheme that approximates the time delay as follows: For every time $\theta \geq 0$, the amount of flow that should have reached the sink in an earliest arrival flow by time θ, reaches the sink at latest at time $(1+\epsilon)\theta$. Tjandra [35] shows how to compute earliest arrival transshipments in networks with time dependent supplies and capacities in time polynomial in the time horizon and the total supply at sources.

Earliest arrival flows and transshipments are motivated by applications related to evacuation. In the context of emergency evacuation from buildings, Berlin [7] and Chalmet, Francis, and Saunders [9] study the quickest transshipment problem in networks with multiple sources and a single sink. Jarvis and Ratliff [23] show that three different objectives of this optimization problem can be achieved simultaneously: (i) Minimizing the total time needed to send the supplies of all sources to the sink, (ii) fulfilling the earliest arrival property, and (iii) minimizing the average time for all flow needed to reach the sink. Hamacher and Tufecki [19] study an evacuation problem and propose solutions which further prevent unnecessary movement within a building.

4 Minimum Cost Flows Over Time

In this section we consider flows over time in networks with additional cost coefficients $c_e \geq 0$ on the arcs $e \in E$. As in the case of static flows, c_e is the cost for sending one flow unit along

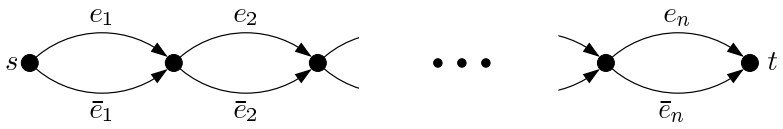

Figure 6: The network obtained from an instance of the Partition Problem in the proof of Theorem 4.1.

arc e. Thus, the cost of a given flow over time f with time horizon T is

$$c(f) \;:=\; \sum_{e\in E} c_e \cdot \int_0^T f_e(\theta)\, d\theta \;. \tag{6}$$

We consider the following minimum cost flow over time problem.

Minimum Cost s-t-Flow Over Time Problem
Given: A network $G=(V,E)$ with capacities, transit times, and costs on the arcs, a source node $s \in V$, a sink node $t \in V$, a time horizon $T \geq 0$, and a demand $d \geq 0$.
Task: Find a feasible s-t-flow over time f with time horizon T, value d, and minimum cost.

Surprisingly, and in contrast to static min-cost flow problems, this problem is already NP-hard.

Theorem 4.1. *The Minimum Cost s-t-Flow Over Time Problem is weakly NP-hard.*

Proof. We consider the corresponding decision problem that asks for a feasible s-t-flow over time f with time horizon T, value d, and cost at most b for some given cost bound b. We reduce the well-known weakly NP-complete Partition Problem to this decision problem.

Partition Problem

Given: $a_1, \ldots, a_n \in \mathbb{Z}_{>0}$ with $A := \sum_{i=1}^n a_i$ even.
Question: Is there a subset $I \subseteq \{1,\ldots,n\}$ with $\sum_{i\in I} a_i = \sum_{i\notin I} a_i$.

We denote the complement of $I \subseteq \{1,\ldots,n\}$ by $\bar{I} := \{1,\ldots,n\} \setminus I$.

Given an instance of the Partition Problem, we build up the network consisting of unit capacity arcs[1] depicted in Figure 6. For $i = 1,\ldots,n$, arc e_i has transit time $\tau_{e_i} := a_i$ and cost $c_{e_i} := 0$, while arc $\bar{e}_i$ has transit time $\tau_{\bar{e}_i} := 0$ and cost $c_{\bar{e}_i} := a_i$. The demand is set to $d := 2$, the time horizon is $T := 1 + A/2$ and the cost bound is $b := A$. We get a natural bijection between subsets $I \subseteq \{1,\ldots,n\}$ and s-t-paths by defining P_I to be the unique s-t-paths with $e_i \in P_I$ if and only if $i \in I$. Notice that P_I and $P_{\bar{I}}$ are arc-disjoint s-t-paths with

$$\tau(P_I) \;=\; \sum_{i\in I} a_i \;, \qquad c(P_I) \;=\; \sum_{i\notin I} a_i \;,$$
$$\tau(P_{\bar{I}}) \;=\; \sum_{i\notin I} a_i \;, \qquad c(P_{\bar{I}}) \;=\; \sum_{i\in I} a_i \;.$$

[1]The network violates our general assumption that there is at most one arc between any pair of nodes. Notice, however, that this can be avoided if we split arcs by introducing intermediate nodes.

We argue that the Partition instance is a yes-instance if and only if there exists a feasible s-t-flow over time with time horizon T, value d, and cost at most b.

If the given Partition instance is a yes-instance with solution I, the two arc-disjoint paths P_I and $P_{\bar{I}}$ have transit time and cost $A/2$ and can thus be used to send two units of flow (one on each path) to the sink within time $T = 1 + A/2$ and cost $b = 2 \cdot A/2$.

We now assume by contradiction that the given instance of the Partition Problem is a no-instance but f is a feasible s-t-flow over time with time horizon T, value d, and cost at most b. Since all flow arrives strictly before time T in the sink, flow can only be sent along s-t-paths with transit time at most $T-1 = A/2$. Moreover, by our assumption, no path with transit time $A/2$ exists such that all flow travels along paths with transit time at most $A/2-1$. Those paths, however, have cost at least $A/2+1$ such that the total cost of f is at least $2(A/2+1) > b$. This contradicts our assumption and thus proves the result. □

On the positive side, there is an algorithm that solves the Minimum Cost s-t-Flow Over Time Problem in pseudo-polynomial time.

Theorem 4.2. *The Minimum Cost s-t-Flow Over Time Problem can be solved in pseudo-polynomial time.*

The proof of this theorem is given below. It relies on the very general concept of *time-expanded networks* which we introduce next.

Definition 4.3 (Time-expanded network)**.** Let $G = (V, E)$ be a network with capacities u, non-negative integral transit times τ, and costs c on the arcs. For a given time horizon $T \in \mathbb{Z}_{>0}$, the corresponding *time-expanded network* $G^T = (V^T, E^T)$ with capacities and costs on the arcs is defined as follows. For each node $v \in V$ we create T copies $v_0, v_1, \ldots, v_{T-1}$, that is,

$$V^T := \{v_\theta \mid v \in V,\ \theta = 0, 1, \ldots, T-1\} .$$

For each arc $e = (v, w) \in E$, there are $T - \tau_e$ copies $e_0, e_1, \ldots, e_{T-1-\tau_e}$ where arc e_θ connects node v_θ to node $w_{\theta+\tau_e}$. Arc e_θ has capacity $u_{e_\theta} := u_e$ and cost $c_{e_\theta} := c_e$. Moreover, E^T contains *holdover arcs* $(v_\theta, v_{\theta+1})$ for $v \in V$ and $\theta = 0, \ldots, T-2$. The capacity of holdover arcs is infinite and they have zero cost. Summarizing, the set of arcs E^T is given by

$$\begin{aligned} E^T := {} & \{e_\theta = (v_\theta, w_{\theta+\tau_e}) \mid e = (v, w) \in E,\ \theta = 0, 1, \ldots, T-1-\tau_e\} \\ & \cup\ \{(v_\theta, v_{\theta+1}) \mid v \in V,\ \theta = 0, 1, \ldots, T-2\} . \end{aligned}$$

An example of a time-expanded network is given in Figure 7. Notice that the size of the time-expanded network G^T is linear in T and therefore only pseudo-polynomial in the input size.

Lemma 4.4. *Let $G = (V, E)$ be a network with capacities u, non-negative integral transit times τ, and costs c on the arcs. For a given time horizon $T \in \mathbb{Z}_{>0}$, a feasible static s_0-t_{T-1}-flow x in G^T yields a feasible s-t-flow over time f in G with time horizon T, cost $c(x) = c(f)$, and value $|f| = |x|$. The reverse direction is also true.*

Proof. Given x, we define f by

$$f_e(\xi) := x_{e_\theta} \qquad \text{for } e \in E,\ \xi \in [\theta, \theta+1),\ \theta = 0, 1, \ldots, T-1-\tau_e.$$

It is straightforward to verify that f obeys capacity and weak flow conservation constraints, has cost $c(f) = c(x)$, and value $|f| = |x|$. Conversely, given f, we define x by

$$x_{e_\theta} := \int_\theta^{\theta+1} f_e(\xi)\, d\xi \qquad \text{for } e \in E,\ \theta = 0, 1, \ldots, T-1-\tau_e.$$

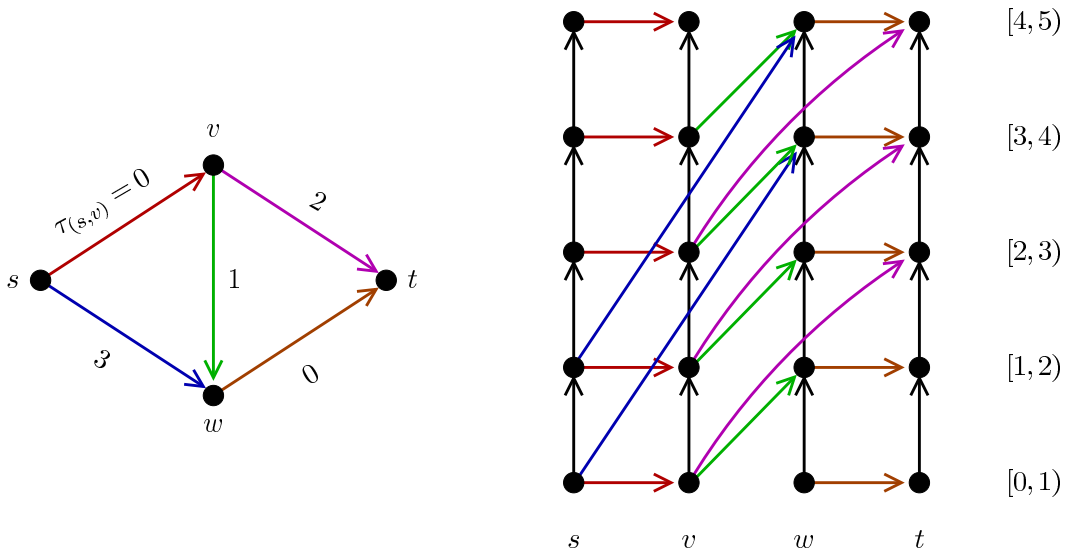

Figure 7: On the left hand side a network G with transit times on the arcs is given. On the right hand side the time-expanded network G^T with time horizon $T = 5$ is depicted. There is one copy of each node for each time interval $[\theta, \theta + 1)$, for $\theta = 0, 1, \ldots, T - 1$.

The flow values on holdover arcs are defined by

$$x_{(v_\theta, v_{\theta+1})} := \sum_{e \in \delta^-(v)} \int_0^{\theta+1-\tau_e} f_e(\xi)\, d\xi - \sum_{e \in \delta^+(v)} \int_0^{\theta+1} f_e(\xi)\, d\xi \ ,$$

for $v \in V \setminus \{s\}$, $\theta = 0, 1, \ldots, T - 2$. For the special case of the source node s, we set

$$x_{(s_\theta, s_{\theta+1})} := \sum_{e \in \delta^+(s)} \int_{\theta+1}^{T} f_e(\xi)\, d\xi - \sum_{e \in \delta^-(s)} \int_{\theta+1-\tau_e}^{T} f_e(\xi)\, d\xi \ ,$$

for $\theta = 0, 1, \ldots, T - 2$. It is again straightforward to verify that x obeys capacity and flow conservation constraints, has cost $c(x) = c(f)$, and value $|x| = |f|$. □

We can finally prove Theorem 4.2.

Proof of Theorem 4.2. As a result of Lemma 4.4, a min-cost s-t-flow over time in G can be obtained in pseudo-polynomial time by computing a min-cost s_0-t_{T-1}-flow x in the time expanded network G^T. This result relies on the assumption that all transit times and the time horizon are integral. In the more general setting with arbitrary rational transit times and time horizon, integrality can be achieved if we scale time by the least common multiple of the denominators of those rational numbers. This concludes the proof. □

Pointers to the Literature

Time-expanded networks have already been introduced by Ford and Fulkerson [14, 15]. Unfortunately, due to the time expansion, the size of the network grows linearly in T which is, in general, exponential in the input size of the problem. This difficulty has already been pointed out by Ford and Fulkerson. On the other hand, the advantage of a time-expanded network is that it turns the problem of determining an optimal flow over time into a static network flow

problem. Moreover, time-expanded networks are very flexible since they can also model time-dependent capacities, costs, transit times etc. This approach is also used in practice to solve flow over time problems. In many cases, however, the size of these networks makes the problem solution prohibitively expensive.

The NP-hardness result in Theorem 4.1 and the presented reduction are due to Klinz and Woeginger [24]. They also show that computing a temporally repeated flow with minimum cost is strongly NP-hard (via a reduction of the strongly NP-complete 3-Partition Problem). Fleischer and Skutella [11] show that there always exists a minimum cost flow over time obeying the strict flow conservation constraints, even for the case of multiple sources and sinks with given supplies and demands, respectively. Moreover, Fleischer and Skutella [11] introduce condensed time expanded networks of polynomial size that lead to fully polyomial-time approximation schemes for several NP-hard flow over time problems including min-cost flows over time. The idea is to partition time into intervals of size $\epsilon^2 \cdot T/n$ and to round transit times accordingly. For the case that arc costs are proportional to transit times, Fleischer and Skutella [10] describe a very simple fully polyomial-time approximation scheme based on capacity scaling for the Minimum Cost s-t-Flow Over Time Problem. They observe that optimal solutions to this problem are flows over time satisfying the earliest arrival and latest departure property (see Exercise 7 for the definition of latest departure flows). Their algorithm runs directly on the original network (i.e., no time expansion).

5 Multi-Commodity Flows Over Time

In the preceding sections we have considered single-commodity flows in the network G with one source node s and one sink node t. In this section we consider the situation where multiple commodities have to be shipped through a common network G and thus have to share the capacities of arcs. Each commodity i has a source node $s_i \in V$ and a sink node $t_i \in V$.

Definition 5.1 (Multi-commodity flow over time)**.** A *multi-commodity flow over time f with k commodities* and time horizon T is a collection of k single-commodity flows over time f^i, $i = 1, \ldots, k$, with time horizon T. We call f *feasible* if it satisfies the capacity constraints

$$\sum_{i=1}^{k} f_e^i(\theta) \;\leq\; u_e \qquad\qquad \text{for } e \in E,\, \theta \in [0, T),$$

and f^i is an s_i-t_i-flow over time for each commodity $i = 1, \ldots, k$. A feasible multi-commodity flow over time f *satisfies given demands* $d_1, \ldots, d_k$ if $|f^i| \geq d_i$, for $i = 1, \ldots, k$.

MULTI-COMMODITY FLOW OVER TIME PROBLEM

Given: A network $G = (V, E)$ with capacities and transit times on the arcs; k commodities $i = 1, \ldots, k$, each with a source node $s_i \in V$, a sink node $t_i \in V$, and a demand $d_i \geq 0$; a time horizon $T \geq 0$.

Task: Find a feasible multi-commodity flow over time with time horizon T satisfying the given demands.

We mention the following theorem on the complexity of the Multi-Commodity Flow Over Time Problem without proof (see references at the end of this section).

Theorem 5.2. *For $k \geq 2$ commodities, the Multi-Commodity Flow Over Time Problem is weakly NP-hard.*

It is not clear whether the Multi-Commodity Flow Over Time Problem is contained in the class NP since the encoding size of a feasible solution might be exponential in the input size. This is an interesting open problem for future research.

On the other hand, the concept of time-expanded networks yields the following positive result on the complexity of the problem.

Theorem 5.3. *The Multi-Commodity Flow Over Time Problem can be solved in pseudo-polynomial time.*

Proof. The Multi-Commodity Flow Over Time Problem on network G can be reduced to a static multi-commodity flow problem on the time-expanded network G^T (see Definition 4.3). Static multi-commodity flows can be computed by general linear programming techniques in time polynomial in the size of the network. □

As for the case of single-commodity flows over time, we consider multi-commodity flows over time with a simple structure and generalize the concept of temporally repeated flows.

Definition 5.4 (Temporally repeated multi-commodity flow)**.** A temporally repeated multi-commodity flow f with time horizon T is a collection of temporally repeated flows f^i with time horizon T, for each commodity $i = 1, \dots, k$, such that the underlying s_i-t_i-flows x^i form a feasible multi-commodity flow x, that is, $\sum_{i=1}^{k} x_e^i \leq u_e$ for each $e \in E$.

Lemma 5.5. *A temporally repeated multi-commodity flow f with time horizon T is a feasible multi-commodity flow over time with time horizon T.*

Proof. By definition, f^i is an s_i-t_i-flow over time with time horizon T, for $i = 1, \dots, k$. Moreover, capacity constraints are satisfied since $f_e^i(\theta) \leq x_e^i$, for $e \in E$, $\theta \in [0, T)$, and thus

$$\sum_{i=1}^{k} f_e^i(\theta) \;\leq\; \sum_{i=1}^{k} x_e^i \;=\; x_e \;\leq\; u_e \qquad\qquad \text{for } e \in E,\, \theta \in [0, T).$$

This concludes the proof. □

We obtain the following approximation result.

Theorem 5.6. *If there is a feasible multi-commodity flow over time f with time horizon T satisfying demands $d_1, \dots, d_k$, then there exists a temporally repeated multi-commodity flow with time horizon $2T$ satisfying demands $d_1, \dots, d_k$. Moreover, such a temporally repeated multi-commodity flow can be computed in polynomial time.*

Proof. Consider a feasible multi-commodity flow over time f with time horizon T satisfying demands $d_1, \dots, d_k$. We define a static multi-commodity flow $\bar{x}$ by averaging f over the entire time horizon, that is,

$$\bar{x}_e^i \;:=\; \frac{1}{T} \cdot \int_0^T f_e^i(\theta)\, d\theta \qquad\qquad \text{for } i = 1, \dots, k,\, e \in E. \tag{7}$$

The feasibility of f implies that the capacity constraint $\sum_{i=1}^{k} f_e^i(\theta) \leq u_e$ holds for $e \in E$, $\theta \in [0, T)$. As a consequence, $\bar{x}$ also obeys capacity constraints:

$$\sum_{i=1}^{k} \bar{x}_e^i \;=\; \frac{1}{T} \cdot \int_0^T \sum_{i=1}^{k} f_e^i(\theta)\, d\theta \;\leq\; \frac{1}{T} \cdot \int_0^T u_e\, d\theta \;=\; u_e \;.$$

Similarly, since $\mathrm{ex}_{f^i}(v,T) = 0$ for $v \in V \setminus \{s_i, t_i\}$, $i = 1, \ldots, k$ (see Definition 2.3 (c)) it follows that $\bar{x}^i$ obeys flow conservation:

$$\sum_{e \in \delta^-(v)} \bar{x}_e^i - \sum_{e \in \delta^+(v)} \bar{x}_e^i \;=\; \frac{1}{T} \cdot \left(\sum_{e \in \delta^-(v)} \int_0^T f_e^i(\theta)\, d\theta - \sum_{e \in \delta^+(v)} \int_0^T f_e^i(\theta)\, d\theta \right)$$
$$= \; \frac{\mathrm{ex}_{f^i}(v,T)}{T} \;=\; 0$$

for $v \in V \setminus \{s_i, t_i\}$, $i = 1, \ldots, k$. Applying the same argument for node t implies that $|\bar{x}^i| = |f^i|/T$ for $i = 1, \ldots, k$.

We have thus shown that $\bar{x}$ is a feasible multi-commodity flow with $|\bar{x}^i| = |f^i|/T$, for $i = 1, \ldots, k$. Moreover, for $i = 1, \ldots, k$,

$$\sum_{e \in E} \tau_e \cdot \bar{x}_e^i \;=\; \frac{1}{T} \cdot \sum_{e \in E} \tau_e \cdot \int_0^T f_e^i(\theta)\, d\theta \;. \tag{8}$$

If we set $c_e := \tau_e$ for each $e \in E$, the term on the right hand side of (8) is equal to $1/T$ times the cost $c(f^i)$ of the flow over time f^i; see (6). Since f^i has time horizon T, flow can only travel along paths of cost at most T. This means that each flow unit causes cost at most T and the total cost $c(f^i)$ is thus upper bounded by $T \cdot |f^i|$.

Putting things together we show that the temporally repeated multi-commodity flow with time horizon $2T$ corresponding to the feasible multi-commodity flow $\bar{x}$ satisfies demands $d_1, \ldots, d_k$. By Lemma 2.7 the flow value of the ith commodity in the temporally repeated multi-commodity flow is equal to

$$2T \cdot |\bar{x}^i| - \sum_{e \in E} \tau_e \cdot \bar{x}_e^i \;=\; 2 \cdot |f^i| - \frac{1}{T} \cdot c(f^i) \;\geq\; |f^i| \;\geq\; d_i \;,$$

for $i = 1, \ldots, k$. Thus, the temporally repeated multi-commodity flow with time horizon $2T$ corresponding to the feasible multi-commodity flow $\bar{x}$ has the desired properties. This concludes the proof of existence.

It remains to show that such a temporally repeated multi-commodity flow can be computed in polynomial time. The given proof of existence requires knowledge of the multi-commodity flow over time f which is NP-hard to compute. We can, however, compute a static multi-commodity flow x which mimics $\bar{x}$ and still yields the desired temporally repeated multi-commodity flow with time horizon $2T$. The task is to find a feasible multi-commodity flow x such that x^i is an s_i-t_i-flow with

$$2T \cdot |x^i| - \sum_{e \in E} \tau_e \cdot x_e^i \;\;\geq\; d_i \;, \tag{9}$$

for all $i = 1, \ldots, k$. This is a multi-commodity flow problem with linear cost constraints and can easily be formulated as a linear program of polynomial size. Moreover, as argued above, the temporally repeated multi-commodity flow with time horizon $2T$ corresponding to the feasible multi-commodity flow x has the desired properties and can be obtained from x in polynomial time. □

The result in Theorem 5.6 implies an approximation algorithm with performance guarantee 2 for the Quickest Multi-Commodity Flow Problem.

Quickest Multi-Commodity Flow Problem
Given: A network $G = (V, E)$ with capacities and transit times on the arcs; k commodities $i = 1, \ldots, k$, each with a source node $s_i \in V$, a sink node $t_i \in V$, and a demand $d_i \geq 0$.
Task: Find a feasible multi-commodity flow over time with minimum time horizon T satisfying the given demands.

Corollary 5.7. *There is an approximation algorithm with performance guarantee* 2 *for the Quickest Multi-Commodity Flow Problem.*

Sketch of proof. If one knew the minimum time horizon T^*, the algorithm described in the proof of Theorem 5.6 immediately yields a 2-approximate solution. The remaining problem is thus to find an appropriate estimate of T^*. The algorithm described in the proof of Theorem 5.6 can, for example, be embedded into a binary search for the minimum time horizon T such that a feasible static multi-commodity flow x satisfying (9) exists. It follows from the proof of Theorem 5.6 that this minimum value T is a lower bound on T^*. Standard binary search can, however, only determine the minimum value T up to a certain finite precision ϵ (notice that neither T nor T^* are in general integral). This yields an approximation algorithm with performance guarantee $2 + \epsilon$ for any fixed $\epsilon > 0$. In order to get rid of the additional ϵ in the performance guarantee, one can replace the binary search by a parametric search. We omit further details. □

Pointers to the Literature

The NP-hardness result in Theorem 5.2 is due to Hall, Hippler, and Skutella [18]. They also show that the Multi-Commodity Flow Over Time Problem with simple flow paths and strict flow conservation is strongly NP-hard. The presented approximation result for the Quickest Multi-Commodity Flow Problem is due to Fleischer and Skutella [11]. The described approach also works in a more general setting with costs by using length-bounded static flow computations. Martens and Skutella [28] make use of this approach to approximate s-t-flows over time with a bound on the number of flow-carrying paths (*k-splittable flows over time*). Moreover, fully polynomial-time approximation schemes for various quickest flow problems can be obtained by using condensed time-expanded networks [11]. More results on single-commodity quickest flows are described at the end of Section 2.

Exercises

Exercise 1. A simple upper bound on the value of a maximum s-t-flow over time with time horizon T can be obtained by multiplying the minimum capacity of a (static) s-t-cut by T. Prove that this upper bound can diverge from the maximum value of a feasible s-t-flow over time by an arbitrarily large factor.

Exercise 2. Consider the network G depicted in Figure 8.

(a) Use the Ford-Fulkerson Algorithm in order to determine a maximum s-t-flow over time with time horizon $T = 18$.

(b) Determine a minimum capacity s-t-cut over time with time horizon $T = 18$.

Exercise 3. Consider a transshipment problem with a given vector of supplies and demands at the nodes of the network G. It is well-known that the static flow problem to satisfy all supplies and demands can be reduced to a maximum s-t-flow problem by introducing a super source s and a super sink t. Try to find a similar reduction for the case of flows over time and discuss the principal difficulties that occur.

Exercise 4. Find a network such that no temporally repeated flow has the earliest arrival property.

Exercise 5. Use the Earliest Arrival Algorithm in order to determine an earliest arrival flow f with time horizon $T = 18$ in the network depicted in Figure 8. Draw the graph of the function $\theta \mapsto \mathrm{ex}_f(t, \theta)$.

Exercise 6. Definition 3.1 of an earliest arrival flow can be generalized straightforwardly to a setting with multiple sources, each with a given supply. It turns out that an earliest arrival flow still exists in this case. On the other hand, we run into problems when there are multiple sinks $t_1, \ldots, t_k$ with given demands $d_1, \ldots, d_k$ that bound the excess of the sinks. That is, $\mathrm{ex}_f(t_i, \theta) \leq d_i$ must hold for all i and θ. Construct an example with two sinks and one source for which no flow over time f maximizes $\mathrm{ex}_f(t_1, \theta) + \mathrm{ex}_f(t_2, \theta)$ for all θ simultaneously.

Exercise 7. An s-t-flow over time f with time horizon T is a *latest departure flow*, if it maximizes the amount of flow leaving the source after time θ for all $\theta \in [0, T]$ (subject to the constraint that all flow must arrive at t before time T). Prove that the flow over time computed by the Earliest Arrival Algorithm is a latest departure flow.

Exercise 8. We consider temporally repeated flows in the context of min-cost flows over time.

(a) Construct a network with two temporally repeated flows f and f' corresponding to two different path decompositions of the same static s-t-flow x such that $c(f) \neq c(f')$.

(b) Construct an example in which the cost of any temporally repeated flow with time horizon T and value d is larger than the minimum cost of an s-t-flow over time with time horizon T and value d.

Exercise 9. Show that an arbitrary s_0-$t_{(T-1)}$-cut with capacity $\delta < \infty$ in the time-expanded network G^T naturally induces an s-t-cut over time with time horizon T and capacity δ in G.

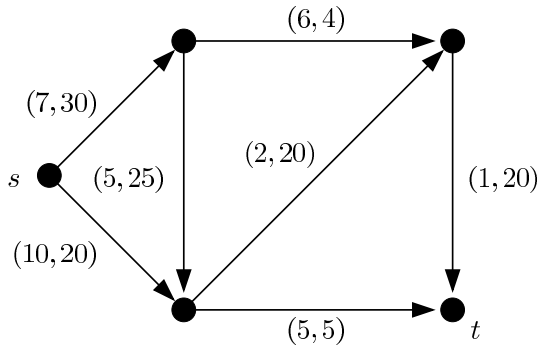

Figure 8: An instance of the Maximum Flow Over Time Problem. The arc labels indicate transit times and capacities, i.e., the label at arc e is (τ_e, u_e). The time horizon is $T = 18$.

Exercise 10. It follows from Theorem 2.12 that there always exists a maximum s-t-flow over time obeying the strict flow conservation constraints. The purpose of this exercise is to show that this result does not hold in the more general setting with multiple commodities.

(a) Find an instance of the Multi-Commodity Flow Over Time Problem such that any multi-commodity flow over time with time horizon T satisfying all demands uses storage of flow at intermediate nodes for at least one commodity.
Hint: Consider the instance given in Figure 9.

(b) Try to come up with an instance of the Quickest Multi-Commodity Flow Problem such that the ratio of the minimal possible time horizon with and without storage of flow at intermediate nodes, respectively, is as large as possible.
Remark: This is an open research question. There is no instance known which achieves a larger ratio than the one depicted in Figure 9. On the positive side, Theorem 5.6 implies that the ratio is always upper bounded by 2.

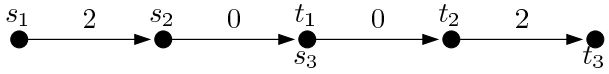

Figure 9: An example with $k = 3$ commodities. Commodities 1 and 3 each have demand 1, commodity 2 has demand 2. The numbers at the arcs indicate transit times; all arcs have unit capacity.

Acknowledgements. We thank an anonymous referee for many useful comments that helped to improve the presentation of the paper.

References

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows. Theory, Algorithms, and Applications*, Prentice Hall, Englewood Cliffs, NJ, 1993.

[2] E. J. Anderson and P. Nash, *Linear Programming in Infinite-Dimensional Spaces*, Wiley, New York, 1987.

[3] E. J. Anderson, P. Nash, and A. B. Philpott, *A class of continuous network flow problems*, Mathematics of Operations Research, 7 (1982), pp. 501–514.

[4] E. J. Anderson and A. B. Philpott, *Optimisation of flows in networks over time*, in Probability, Statistics and Optimisation, F. P. Kelly, ed., Wiley, New York, 1994, ch. 27, pp. 369–382.

[5] J. E. Aronson, *A survey of dynamic network flows*, Annals of Operations Research, 20 (1989), pp. 1–66.

[6] N. Baumann and M. Skutella, *Solving evacuation problems efficiently: Earliest arrival flows with multiple sources*, in Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science, Berkeley, CA, 2006, pp. 399–408.

[7] G. N. Berlin, *The use of directed routes for assessing escape potential*, National Fire Protection Association, Boston, MA, 1979.

[8] R. E. Burkard, K. Dlaska, and B. Klinz, *The quickest flow problem*, ZOR — Methods and Models of Operations Research, 37 (1993), pp. 31–58.

[9] L. G. Chalmet, R. L. Francis, and P. B. Saunders, *Network models for building evacuation*, Management Science, 28 (1982), pp. 86–105.

[10] 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.

[11] ——, *Quickest flows over time*, SIAM Journal on Computing, 36 (2007), pp. 1600–1630.

[12] L. K. Fleischer, *Faster algorithms for the quickest transshipment problem*, SIAM Journal on Optimization, 12 (2001), pp. 18–35.

[13] L. K. Fleischer and E. Tardos, *Efficient continuous-time dynamic network flow algorithms*, Operations Research Letters, 23 (1998), pp. 71–80.

[14] L. R. Ford and D. R. Fulkerson, *Constructing maximal dynamic flows from static flows*, Operations Research, 6 (1958), pp. 419–433.

[15] ——, *Flows in Networks*, Princeton University Press, 1962.

[16] D. Gale, *Transient flows in networks*, Michigan Mathematical Journal, 6 (1959), pp. 59–63.

[17] B. Hajek and R. G. Ogier, *Optimal dynamic routing in communication networks with continuous traffic*, Networks, 14 (1984), pp. 457–487.

[18] A. Hall, S. Hippler, and M. Skutella, *Multicommodity flows over time: Efficient algorithms and complexity*, Theoretical Computer Science, 379 (2007), pp. 387–404.

[19] H. W. Hamacher and S. Tufecki, *On the use of lexicographic min cost flows in evacuation modeling*, Naval Research Logistics, 34 (1987), pp. 487–503.

[20] B. Hoppe, *Efficient dynamic network flow algorithms*, PhD thesis, Cornell University, 1995.

[21] B. Hoppe and E. Tardos, *Polynomial time algorithms for some evacuation problems*, in Proceedings of the 5th Annual ACM–SIAM Symposium on Discrete Algorithms, Arlington, VA, 1994, pp. 433–441.

[22] ——, *The quickest transshipment problem*, Mathematics of Operations Research, 25 (2000), pp. 36–62.

[23] J. Jarvis and H. Ratliff, *Some equivalent objectives for dynamic network flow problems*, Management Science, 28 (1982), pp. 106–108.

[24] B. Klinz and G. J. Woeginger, *Minimum-cost dynamic flows: The series-parallel case*, Networks, 43 (2004), pp. 153–162.

[25] B. Korte and J. Vygen, *Combinatorial Optimization: Theory and Algorithms*, Springer, Berlin, 4th ed., 2008.

[26] B. Kotnyek, *An annotated overview of dynamic network flows*, Rapport de recherche 4936, INRIA Sophia Antipolis, 2003.

[27] S. E. Lovetskii and I. I. Melamed, *Dynamic network flows*, Automation and Remote Control, 48 (1987), pp. 1417–1434. Translated from Avtomatika i Telemekhanika, 11 (1987), pp. 7–29.

[28] M. Martens and M. Skutella, *Length-bounded and dynamic k-splittable flows*, in Operations Research Proceedings 2005, H.-D. Haasis, H. Kopfer, and J. Schönberger, eds., Springer, 2006, pp. 297–302.

[29] N. Megiddo, *Optimal flows in networks with multiple sources and sinks*, Mathematical Programming, 7 (1974), pp. 97–107.

[30] ——, *Combinatorial optimization with rational objective functions*, Mathematics of Operations Research, 4 (1979), pp. 414–424.

[31] E. Minieka, *Maximal, lexicographic, and dynamic network flows*, Operations Research, 21 (1973), pp. 517–527.

[32] A. B. Philpott, *Continuous-time flows in networks*, Mathematics of Operations Research, 15 (1990), pp. 640–661.

[33] 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., vol. 8 of Handbooks in Operations Research and Management Science, North–Holland, Amsterdam, The Netherlands, 1995, ch. 3, pp. 141–295.

[34] A. Schrijver, *Combinatorial Optimization: Polyhedra and Efficiency*, Springer, Berlin, 2003.

[35] S. Tjandra, *Dynamic Network Optimization with Application to the Evacuation Problem*, PhD thesis, Universität Kaiserslautern, Shaker Verlag, Aachen, 2003.

[36] W. L. Wilkinson, *An algorithm for universal maximal dynamic flows in a network*, Operations Research, 19 (1971), pp. 1602–1612.

[37] N. Zadeh, *A bad network problem for the simplex method and other minimum cost flow algorithms*, Mathematical Programming, 5 (1973), pp. 255–266.