Flows over time

Winter semester 2011/2012

14th February 2012

Contents

1 Introduction

1.1 Motivation

Consider a number of persons in a city who want to walk from place a to place b within a certain time T. Assume that you can guide them though the city. How could you help them?

To formalize this problem, we may model the city by a (street) network such that a and b are nodes. We assume that we know for each edge (v, w) the time $t(v, w)$ for traversing it from v to w.

Now, if the number of persons is small, it is sufficient to compute a shortest a-b-path in the network and to compare its length to T.

But if there are many persons (e.g. the audience of soccer games or open-air-concerts), one would possibly try to avoid sending all of them through the same streets. It is a straight-forward idea to model this version as a max-flow-problem. We assume that we know for each edge (v, w) the number of persons $n(v, w)$ that can traverse edge (v, w) within one time unit. The classical approach would be to assign a capacity $u(v, w)$ to (v, w) and to compute a maximum a-b-flow.

The question is how to define $u(v, w)$. Setting $u(v, w) := T \cdot n(v, w)$ for all edges $u(v, w)$ would be too optimistic because people cannot spend the whole time T with traversing a certain edge (v, w). On the other hand, always setting $u(v, w)$ to something smaller than $T \cdot n(v, w)$ (e.g. $u(v, w) := t(v, w) \cdot n(v, w)$) would be too pessimistic because when traveling along different paths people may arrive at different times at node v and hence traverse edge (v, w) at different times. Hence the total number of people traversing edge (v, w) may be larger than (e.g.) $t(v, w) \cdot n(v, w)$.

Hence, modelling this problem just by edge capacities is not appropriate.

Therefore, time has to be considered directly whenever logistic transportation problems are modelled as flow problems (and such problems from logistics were the first motivation to analyse flow problems, see Schrijver [2002]).

1.2 Static Flows

We first want to summarize some definitions and important results on static network flows. For a more comprehensive introduction to static flow problems we refer to standard textbooks, e.g. Ahuja, Magnanti, and Orlin [1993], Korte and Vygen [2012], and Schrijver [2003]. Our notation concerning graphs is very similar to the notation used in Korte and Vygen [2012].

Definition 1.1 (a) A *(static) flow network* is a pair (G, u) where G is a directed graph and $u : E(G) \to \mathbb{R}_{>0} \cup \{\infty\}$ is a mapping. We call $u(e)$ the *capacity of* e (for $e \in E(G)$).

(b) A *(static) flow* in a static flow network (G, u) is a mapping $f : E(G) \to \mathbb{R}_{\geq 0}$ with $f(e) \leq u(e)$ for $e \in E(G)$. If f is a flow in (G, u), then we call $\mathrm{ex}_f(v) := \sum_{e \in \delta^-(v)} f(e) - \sum_{e \in \delta^+(v)} f(e)$ the *balance of* f *in* v (for $v \in V(G)$). If $\mathrm{ex}_f(v) = 0$ for all $v \in V(G)$, then f is called a *circulation.*

(c) Let (G, u) be a static flow network with two different nodes $s, t \in V(G)$. A flow f in (G, u) is a *(static) s-t-flow* if $\mathrm{ex}_f(v) = 0$ for all $Vv \in (G) \setminus \{s, t\}$ and $\mathrm{ex}_f(t) \geq 0$. The *value of* f is defined as $|f| = \mathrm{ex}_f(t)$.

(d) Let (G, u) be a static flow network, and let $b : V(G) \to \mathbb{R}$ be a mapping with $\sum_{v \in V(G)} b(v) = 0$. A flow f in (G, u) is called *b-flow in* (G, u) if $\mathrm{ex}_f(v) = -b(v)$ for all $v \in V(G)$.

(e) Given a static flow network (G, u) and a cost function $c : E(G) \to \mathbb{R}$, the cost of a flow f in (G, u) is $\sum_{e \in E(G)} c(e) \cdot f(e)$.

Facts on static flows:

- A maximum s-t-flow in a static flow network (G, u) can be computed in time $O(nm \log(n^2/m))$ (see Goldberg and Tarjan [1988]) and in time $O(n^3/\log n)$ (see Cheriyan, Hagerup, and Mehlhorn [1996]) where $n = |V(G)|$ and $m = |E(G)|$.
- Max-flow min-cut theorem: Let (G, u) be a static flow network with two different nodes $s, t \in V(G)$. Then, the value of a maximum s-t-flow equals the value of a minimum s-t-cut which is defined as
$$\min \left\{ \sum_{e \in E(G) \cap (X \times (V(G) \setminus X))} u(e) \mid s \in X \subseteq (V(G) \setminus \{t\}) \right\}.$$
- Let (G, u) be a static flow network and $b : V(G) \to \mathbb{R}$ be a mapping with $\sum_{v \in V(G)} b(v) = 0$. Then, there is a b-flow in G if and only if for all $U \subseteq V(G)$: $u(\delta^+(U)) := \sum_{e \in \delta^+(U)} u(e) \geq \sum_{v \in U} b(v)$ (see Gale [1957]).
- A minimum-cost b-flow in a network with n vertices and m edges can be computed in time $O(m \log m(m + n \log n))$. If all capacities are infinite, a minimum-cost flow can be computed in time $O(n \log m(m + n \log n))$ (see Orlin [1993]).
- Flow decomposition: Let f be an s-t-flow in a flow network (G, u) with n nodes and m edges. Then, there is a set $\mathcal{C}$ of cycles and a set $\mathcal{P}$ of s-t-paths in G with numbers $f_P \geq 0$ for $P \in \mathcal{C} \cup \mathcal{P}$ such that
$$f(e) = \sum_{P \in \mathcal{C} \cup \mathcal{P} : e \in E(P)} f_P$$

for all $e \in E(G)$. The set $\mathcal{C} \cup \mathcal{P}$ can be chosen such that $|\mathcal{C} \cup \mathcal{P}| \leq m$. We denote the decomposition by $(f_p)_{p \in \mathcal{C} \cup \mathcal{P}}$. If f is a circulation, then there is a decomposition where $\mathcal{P}$ is empty. Such a decomposition can be computed in time $O(nm)$.

Definition 1.2 Let G be a directed graph. We define the graph $\overset{\leftrightarrow}{G}$ by $V(\overset{\leftrightarrow}{G}) = V(G)$ and $E(\overset{\leftrightarrow}{G}) = E(G) \dot{\cup} \{\overset{\leftarrow}{e} \mid e \in E(G)\}$ where $\overset{\leftarrow}{e}$ is an edge from w to v if e is an edge from v to w. $\overset{\leftarrow}{e}$ is called the *reverse edge* of e. Note that $\overset{\leftrightarrow}{G}$ may have parallel edges even if G is simple. If we have edge costs $c : E(G) \to \mathbb{R}$ these are extended canonically to edges in $E(\overset{\leftrightarrow}{G})$ by setting $c(\overset{\leftarrow}{e}) = -c(e)$.
Let (G, u) be a static flow network with edge costs $c : E(G) \to \mathbb{R}$, and let f be a flow in (G, u). Then, the *residual graph* $G_{u,f}$ is defined by $V(G_{u,f}) := V(G)$ and $E(G_{u,f}) := \{e \in E(G) \mid f(e) < u(e)\} \dot{\cup} \{\overset{\leftarrow}{e} \in \overset{\leftrightarrow}{E}(G) \mid f(e) > 0\}$. For $e \in E(G)$ we define the *residual capacity* of e by $u_f(e) = u(e) - f(e)$ and the residual capacity of $\overset{\leftarrow}{e}$ by $u_f(\overset{\leftarrow}{e}) = f(e)$.

The residual graph contains the edges where flow can be increased as forward edges and edges where flow can be reduced as reverse edges. In both cases, the residual capacity is the maximum value by which the flow can be modified. If P is a subgraph of the residual graph, then an augmentation along P by γ means that we increase the flow on forward edges by γ and reduce it on reverse edge by γ. Note that the resulting mapping is only a flow if γ is at most the minimum of the residual capacity of the edges in P.

1.3 Flows over Time

Definition 1.3 A *discrete/continuous dynamic flow network* is a triple (G, u, τ) where (G, u) is a static flow network and τ is a mapping $\tau : E(G) \to \mathbb{N}$ (discrete case) or $\tau : E(G) \to \mathbb{R}_{\geq 0}$ (continuous case), respectively. The value $\tau(e)$ is called the *transit time* of e (for $e \in E(G)$).

Definition 1.4 Let (G, u, τ) be a discrete dynamic flow network and $T \in \mathbb{N}$. A *discrete flow over time with time horizon* T in (G, u, τ) is a set f of functions $f_e : \{0, \ldots, T\} \to [0, u(e)]$ such that $f_e(\theta) = 0$ for $\theta > T - \tau(e)$. We define $\mathrm{dom}(f) := \{0, \ldots, T\}$. The balance of f in v at time $\theta \in \{0, \ldots, T\}$ is

$$\mathrm{ex}_f(v, \theta) := \sum_{e \in \delta^-(v)} \sum_{\zeta=0}^{\theta - \tau(e)} f_e(\zeta) - \sum_{e \in \delta^+(v)} \sum_{\zeta=0}^{\theta} f_e(\zeta).$$

In this model, $f_{(v,w)}(\theta)$ is the amount of flow that enters the edge (v, w) at time θ and reaches w at time $\theta + \tau(v, w)$.

Definition 1.5 A *continuous flow over time with time horizon* T in a dynamic flow network (G, u, τ) is a set f of Lebesgue-integrable functions $f_e : \mathbb{R} \to [0, u(e)]$ such that $f_e(\theta) = 0$ for $\theta \notin [0, T - \tau(e))$ (for $e \in E(G)$). We define $\operatorname{dom}(f) := [0, T)$. The balance of f in v at time $\theta \in \mathbb{R}$ is the value

$$\operatorname{ex}_f(v, \theta) := \sum_{e \in \delta^-(v)} \int_0^{\theta - \tau(e)} f_e(\zeta) d\zeta - \sum_{e \in \delta^+(v)} \int_0^{\theta} f_e(\zeta) d\zeta.$$

Another term for "flows over time" will be "dynamic flow".

Discrete and continuous flows over time are closely related and we will see many results that are true for both kinds of flow. An important difference is that in discrete flows, packages of flow are sent through an edge that arrive at the same time. Consider for example the graph G with $V(G) = \{s, t\}$, $E(G) = \{(s, t)\}$, $u(s, t) = 1$ and $\tau(s, t) = 2$. Then, with a time horizon of $T = 3$ we could send a flow of size 2 in the discrete model (by sending two flow packages of size 1 at time 0 and 1) but only a flow of size 1 in the continuous model (by starting to send flow at time 0 and stopping to send flow at time 1).

2 Maximum Flows over Time

Definition 2.1 Let (G, u, τ) be a dynamic flow network with two different vertices s and t, and let $f = \{f_e \mid e \in E(G)\}$ be a (discrete or continuous) flow over time in (G, u, τ) with time horizon T. f is called a *discrete/continuous s-t-flow over time with time horizon* T if $\operatorname{ex}_f(v, \theta) \geq 0$ for $v \in V(G) \setminus \{s\}$ and $\theta \in \operatorname{dom}(f)$ and $\operatorname{ex}_f(v, T) = 0$ for $v \in V(G) \setminus \{s, t\}$ (*weak flow conservation constraint*). The *value* of f is $\operatorname{val}(f) := \operatorname{ex}_f(t, T)$.

Remark: In the above definition flow may be stored in a node (i.e. $\operatorname{ex}_f(v, \theta)$ may be positive for $v \in V(G) \setminus \{s\}$ and $\theta \in \operatorname{dom}(f)$). Sometimes we will ask for flows where the *strong flow conservation constraint* is met, i.e. where $\operatorname{ex}_f(v, \theta) = 0$ for $v \in V(G) \setminus \{s, t\}$ and $\theta \in \operatorname{dom}(f)$.

2.1 Time-expanded networks

Definition 2.2 Let (G, u, τ) be a dynamic flow network with integral transit times and let $T \geq 0$ be an integer. Then, the *time-expanded network for* (G, u, τ) *and* T is a static flow network (G', u') such that:

- $V(G') := V(G) \times \{0, \dots, T\}$,
- $E(G') := \{((v, \theta), (w, \zeta)) \in V(G') \times V(G') \mid (v = w \text{ and } \zeta = \theta + 1) \text{ or } ((v, w) \in E(G) \text{ and } \tau(v, w) = \zeta - \theta)\}$, and
- $u'((v, \theta), (v, \theta + 1)) := \infty$ for $v \in V(G)$, and $u'((v, \theta), (w, \zeta)) := u(v, w)$ for all $(v, w) \in E(G)$.

The graph G' in the above defintion contains $T + 1$ copies of the vertex set $V(G)$.

The edges of the form $((v, \theta), (v, \theta + 1))$ are called *hold edges*. If storing flow is not allowed for some vertices, then the corresponding hold edges have to be removed.

Clearly, a discrete dynamic s-t-flow in (G, u, τ) with time horizon T corresponds to a static $(s, 0)$-(t, T)-flow in the time-expanded network with the same value and vice-versa.

For integral transit times we get a similar correspondence for continuous flows, too:

Lemma 2.1 Let (G, u, τ) by a dynamic flow network with integral transit times and let T be an integral time horizon. Let s and t be to nodes in G.

(a) If g is a static $(s, 1)$-(t, T) flow in the time-expanded network, then there is a dynamic flow f in (G, u, τ) with time horizon T such that $\text{val}(f) = |g|$.

(b) If f is a dynamic flows in (G, u, τ) with time horizon T, then there is a static $(s, 1)$-(t, T) flow g in the time-expanded network with $|g| = \text{val}(f)$.

Proof:

(a) For an edge $e \in (v, w)$ in G and $\theta \in [0, T]$, set $f_e(\theta) = g((v, \lceil\theta\rceil), (w, \lceil\theta\rceil + \tau(e)))$. Then, the flow f respects the capacity constraints because g respects them. The flow conservation rule is met because for a node $v \in V(G) \setminus \{s\}$, the balance $\text{ex}_f(v, \theta)$ of a node v at time θ is between $g((v, \lceil\theta\rceil - 1), (v, \lceil\theta\rceil)$ and $g((v, \lceil\theta\rceil), (v, \lceil\theta\rceil + 1)$. This also shows that $\text{val}(f) = \text{ex}_f(t, T) = |g|$.

(b) For an edge $e \in (v, w)$ in G and $\theta \in \{1, \dots, T\}$, define $g((v, \theta), (w, \theta + \tau(e)) = \int_{\theta-1}^{\theta} f_e(\zeta) d\zeta$. For a node $v \in V(G) \setminus \{s\}$ set $g((v, \theta), (v, \theta + 1)) = \text{ex}_f(v, \theta)$. And set $g((s, \theta), (s, \theta + 1)) = \text{ex}_f(s, \theta) + \text{val}(f)$. Then, g obviously respects the capacity and the balance constraints. $\square$

Several dynamic flow problems with integral transit times and time horizons can be reduced to static flow problems in the corresponding time-expanded network, e.g. the maximum-flow problem:

Proposition 2.2 Given a dynamic flow network (G, u, τ) with integral transit times and an integral time horizon T, a maximum (discrete or continuous) s-t-flow over time with time horizon T can be computed in time $O\left(\frac{T^3n^3}{\log(Tn)}\right)$ where n is the number of nodes in the network.

Proof: Apply the maximum-flow algorithm by Cheriyan, Hagerup, and Mehlhorn [1996] to compute a maximum $(s, 0)$-(t, T)-flow (or $(s, 1)$-(t, T)-flow, respectively) in the time-expanded network. $\square$

Since the size of the time-expanded network is exponential in the input size, it cannot be used to find efficient algorithms. Nevertheless, the time-expanded graph will be useful for some proofs.

2.2 Temporally repeated flows

In this section we will show how a maximum s-t-flow over time can be computed in polynomial time. We will restrict our description to continuous flows but it should be noted that the algorithm can be applied as well to discrete flows. In fact, it was first fomulated for the discrete case.

Notation: For a directed graph G, a mapping $c : E(G) \to \mathbb{R}$, and a path P in G, let $c(P)$ be the length of P, so $c(P) := \sum_{e\in E(P)} c(e)$. If v and w are two vertices in $V(P)$ such that w is reachable from v in P, then $P_{v,w}$ is the subpath in P with start node v and end node w.

The idea of the following algorithm for a maximum s-t-flow over time is quite simple. We first compute an appropriate static s-t-flow in (G, u), decompose it into s-t-paths, and send on each path repeatedly the same amount of flow. More formally we define:

Definition 2.3 Let (G, u, τ) be a dynamic flow network, let s and t be two vertices, and $T \geq 0$ be a time horizon. Let f be a static s-t-flow in (G, u) with flow decomposition $\Gamma_f = (f_p)_{P\in\mathcal{P}\cup\mathcal{C}}$. Then, the corresponding *temporally repeated s-t-flow* $[\Gamma_f]^T$ is defined by

$$[\Gamma_f]^T_e(\theta) := \sum_{P\in\mathcal{P}_e(\theta)} f_P \text{ for } e \in E(G), \theta \in [0, T)$$

where

$$\mathcal{P}_{(v,w)}(\theta) := \left\{P \in \mathcal{P} : (v, w) \in E(P) \wedge \tau(P_{s,v}) \leq \theta \wedge \tau(P_{v,t}) < T - \theta\right\}.$$

Lemma 2.3 Let (G, u, τ) be a dynamic flow network, and let f be a static s-t-flow in (G, u) with flow decomposition $\Gamma_f = (f_P)_{P\in\mathcal{C}\cup\mathcal{P}}$. Then a corresponding temporally repeated s-t-flow with time horizon T is an s-t-flow over time with time horizon T without internal storage.

Proof: The strict flow conservation constraint is met because for any path the flow that enters a node immediately leaves the node. Moreover, f is nonnegative, because the number f_P are nonnegative. It remains to show that the capacity constraints are met. For any edge $e \in E(G)$ and any $\theta \in [0, T)$ we have $[\Gamma_f]_e^T(\theta) \leq \sum_{P \in \mathcal{P}: e \in E(P)} f_P \leq f(e) \leq u(e)$. This shows that $[\Gamma_f]^T$ fulfills the capacity constraints. $\square$

Remark: Note that $[\Gamma_f]^T$ in fact depends on the flow decomposition but not only on f. Consider, e.g., the graph shown in Figure 1. Assume that all capacities and traversal times are 1. If f is a static flow which has value 1 on all edges (which is obviously a maximum s-t-flow), than it can either be decomposed into the path s-d-b-t and the cycle a-b-c-d or into the path s-d-a-b-t and the cycle b-c-d. The temporally repeated flows for both decompostions are different (for $T > 3$) and, in fact even their values differ.

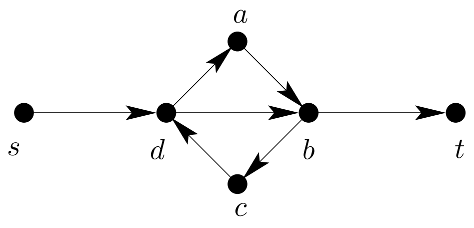

Figure 1: Example: the temporally repeated flow depends on the flow decomposition.

Another example, this time without cycles, is shown in Figure 2. The edges e labeled with "2" have capacity $u(e) = 2$, the remaining edges have capacities $u(e) = 1$. All traversal times are 1. Let f be a flow with $f(e) = u(e)$ on all edges e. Let $T = 7$ be the time horizon. Then, f can be decomposed into the paths s-a-c-d-f-t and s-a-b-c-d-e-f-t. A second decompostion consists of the two paths s-a-b-c-d-f-t and s-a-c-d-e-f-t. It is easy to see that the two decompositions lead to different temporally repeated flows (e.g. at time $\theta = 6$, the balance of t is positive in the first case, while it is 0 in the second case). Nevertheless, note that the value of the two temporally repeated flows is 2. In fact, the next lemma will show that their values are not just by accident identical.

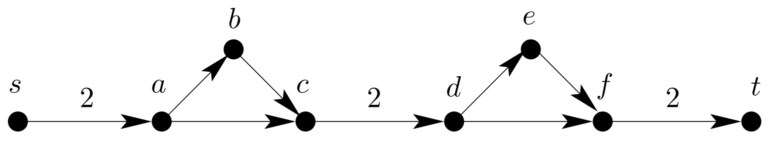

Figure 2: Example: the temporally repeated flow may depend on the flow decomposition even if the graph is acyclic.

Lemma 2.4 Let (G, u, τ) be a dynamic flow network with two vertices s and t and a time horizon $T \geq 0$. Let f be a static s-t-flow in (G, u) with flow decomposition $\Gamma_f = (f_P)_{P \in \mathcal{P} \cup \mathcal{C}}$ such that $f_P = 0$ with $P \in \mathcal{C}$ or $\tau(P) > T$. Then:

$$\mathrm{val}([\Gamma_f]^T) = T \cdot |f| - \sum_{e \in E(G)} \tau(e) \cdot f(e).$$

Proof: We have

$$\begin{aligned}
\mathrm{val}([\Gamma_f]^T) &= \mathrm{ex}_{[\Gamma_f]}(t, T) \\
&= \sum_{e=(v,t) \in E(G)} \int_0^{T-\tau(e)} [\Gamma_f]_e^T(\theta) d\theta \\
&= \sum_{e=(v,t) \in E(G)} \int_0^{T-\tau(e)} \left(\sum_{P \in \mathcal{P}_e(\theta)} f_P \right) d\theta \\
&= \sum_{e=(v,t) \in E(G)} \sum_{P \in \mathcal{P} : e \in E(P)} \int_{\tau(P_{s,v})}^{T-\tau(e)} f_P \, d\theta \\
&= \sum_{e=(v,t) \in E(G)} \sum_{P \in \mathcal{P} : e \in E(P)} (T - \tau(P)) f_P \\
&= \sum_{P \in \mathcal{P}} (T - \tau(P)) \cdot f_P \\
&= T \cdot \sum_{P \in \mathcal{P}} f_P - \sum_{P \in \mathcal{P}} \sum_{e \in E(P)} \tau(e) \cdot f_P \\
&= T \cdot |f| - \sum_{e \in E(G)} \tau(e) \sum_{P \in \mathcal{P} : e \in E(P)} f_P \\
&= T \cdot |f| - \sum_{e \in E(G)} \tau(e) \cdot f(e).
\end{aligned}$$

□

Note that the previous lemma shows in particular that in this case the value of the temporally repeated flow does not depend on the flow decomposition of f.

If we want to compute a maximum temporally repeated flow, we have to maximize $T \cdot |f| - \sum_{e \in E(G)} \tau(e) \cdot f(e)$. This motivates the following algorithm:

FORD-FULKERSON ALGORITHM

Input: A dynamic flow network (G, u, τ), two vertices s and t, and a time horizon $T \geq 0$.

Output: A temporally repeated flow with time horizon T

```
①    Compute a graph $\tilde{G}$ that results from $G$ by adding an edge $(t,s)$ with $\tau(t,s) := -T$
     and $u(t,s) := \sum_{e \in E(G)} u(e)$.
②    Compute a static minimum-cost circulation $f$ in $(\tilde{G}, u)$ with edge weights $\tau(e)$.
     Let $g$ be the restriction of $f$ to $G$.
③    Compute a flow decomposition $\Gamma_g = (g_P)_{P \in \mathcal{P} \cup \mathcal{C}}$ of $g$.
④    Output $[\Gamma_g]^T$.
```

According to the previous lemma, the output of the FORD-FULKERSON ALGORITHM is a maximum temporally repeated flow. In order to show that it is maximum over all s-t-flows over time, we generalize the concept of s-t-cuts to dynamic networks.

Definition 2.4 Let (G, u, τ) be a dynamic flow network, $T \geq 0$, and $s, t \in V(G)$. A *discrete/continuous s-t-cut over time in (G, u, τ) with time horizon T* is a set of numbers $a_v \in \{0, \ldots, T\}$ (in the discrete case) or $a_v \in [0, T]$ (in the continuous case) for $v \in V(G)$ with $a_s = 0$ and $a_t \geq T$. The capacity of an s-t-cut $(a_v)_{v \in V(G)}$ is defined as

$$\sum_{e=(v,w) \in E(G)} \max\{a_w - a_v - \tau(e), 0\} \cdot u(e).$$

Static s-t-cuts can be seen as a special case of s-t-cuts over time where $T = 1$, all transit times are 0, and all numbers a_v are 0 or 1. The s-side of the cut is then the set of nodes v for which the a_v is 0.

As a little detour, we will consider discrete flows again because in this context cuts over time have a nice interpretation:

Lemma 2.5 (Discrete version of the dynamic max-flow min-cut theorem) Let (G, u, τ) be a dynamic flow network. Let $K_{\min}$ be the minimum capacity of a dynamic s-t-cut in (G, u, τ) with time horizon T, and let f be a maximum discrete s-t-flow over time in (G, u, τ) with time horizon $T - 1$. Then $K_{\min} = \mathrm{val}(f)$.

Proof: For discrete instances, s-t-cuts over time can be considered as $(s, 0)$-$(t, T-1)$-cuts in the time-expanded network G' with hold edges: For a vector $(a_v)_{v \in V(G)}$ of numbers in $\{0, \ldots, T\}$ with $a_s = 0$ and $a_t \geq T$ let $X = \{(v, \theta) \in V(G') \mid \theta \geq a_v\}$. Then, we have for $(v, w) \in E(G)$:

$$\sum_{((v,\theta),(w,\zeta)) \in (E(G') \cap (X \times (V(G) \setminus X)))} u'((v,\theta),(w,\zeta)) = \max\{a_w - a_v - \tau(v,w), 0\},$$

so the capacity of the edges in $X \times (V(G') \setminus X)$ equals the capacity of the cut over time.

Due to the hold edges, all $(s, 0)$-$(t, T-1)$-cuts are sets of edges leaving a vertex set of the type $\{(v, \theta) \in V(G') \mid \theta \geq d_v\}$ for some value d_v with $d_s = 0$ and $d_t \geq T$.

Thus, the theorem is a consequence of the (static) max-flow min-cut theorem applied to the time-expanded network. □

For an illustration see Figures 3 and 4. Figure 3 presents a dynamic flow network where the edge labels denote the transit times. Figure 4 shows the time-expanded network (for $T = 4$) and a cut indicated by the sets $\{(v, \theta) \in V(G') \mid \theta \geq a_v\}$ which are for each v marked by a rectangle (i.e. we have $a_s = 0$, $a_x = 2$, $a_y = 1$, and $a_t = 4$). Only the edge from $(s, 0)$ to $(x, 1)$ is counted in the cut (and, in fact it is not a min-cut because there is a cut with capacity 0).

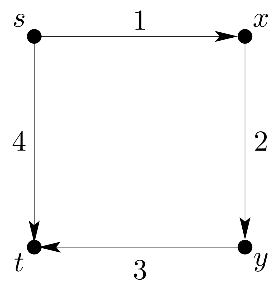

Figure 3: A network.

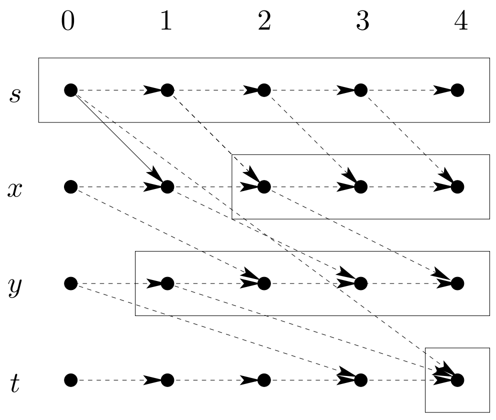

Figure 4: Illustration of a discrete s-t-cut.

Now, we return to the continuous case where we get an analogous result though it does not follow directly from the static max-flow min-cut theorem.

Lemma 2.6 Let (G, u, τ) be a dynamic flow network. Let K be the capacity of a dynamic s-t-cut in (G, u, τ) with time horizon T, and let f be a continuous s-t-flow over time in (G, u, τ) with time horizon T. Then:

$$\mathrm{val}(f) \leq K.$$

Proof: We have $\mathrm{val}(f) = \mathrm{ex}_f(t,T) \leq \mathrm{ex}_f(t,a_t) \leq \sum_{v\in V(G)} \mathrm{ex}_f(v,a_v)$ because $\mathrm{ex}_f(s,a_s) = \mathrm{ex}_f(s,0) = 0$ and $\mathrm{ex}_f(v,a_v) \geq 0$ for $v \in V(G) \setminus \{s\}$.

We can conclude

$$\begin{aligned}
\mathrm{val}(f) &\leq \sum_{v\in V(G)} \mathrm{ex}_f(v,a_v) \\
&= \sum_{v\in V(G)} \left(\sum_{e\in\delta^-(v)} \int_0^{a_v-\tau(e)} f_e(\theta)d(\theta) - \sum_{e\in\delta^+(v)} \int_0^{a_v} f_e(\theta)d(\theta). \right) \\
&= \sum_{e=(v,w)\in E(G)} \left(\int_0^{a_w-\tau(e)} f_e(\theta)d\theta - \int_0^{a_v} f_e(\theta)d\theta \right) \\
&= \sum_{e=(v,w)\in E(G)} \left(\int_{a_v}^{a_w-\tau(e)} f_e(\theta)d\theta \right) \\
&\leq \sum_{\substack{e=(v,w)\in E(G): \\ a_w-\tau(e)-a_v>0}} \left(\int_{a_v}^{a_w-\tau(e)} u(e)d\theta \right) \\
&= \sum_{e=(v,w)\in E(G)} \max\{a_w - \tau(e) - a_v, 0\} \cdot u(e)
\end{aligned}$$

□

Theorem 2.7 The FORD-FULKERSON ALGORITHM computes a maximum s-t-flow over time. It can be implemented to run in time $O(m \log m(m + n \log n))$ where n is the number of nodes and m the number of edges in G.

Proof: For the running time, the most critical step is ②. With the ORLIN'S ALGORITHM (see Orlin [1993]), this step can be performed in time $O(m \log m(m + n \log n))$.

We will apply Lemma 2.6 to prove optimality. For $v \in V(G)$ let $\pi(v) := \mathrm{dist}_{\tilde{G}_{u,f},\tau}(s,v)$. We will show that the numbers $\pi(v)$ define an s-t-cut over time with capacity $\mathrm{val}([\Gamma_g]^T)$.

The residual graph does not contain any negative cycle, so $\pi(s) = 0$ and $\pi(t) \geq T$.

Since f is a circulation, we have $\sum_{e=(v,w)\in E(\tilde{G})}(\pi(v) - \pi(w)) \cdot f(e) = 0$ (f can be written as the sum of flow on cycles, and for each cycle the $\pi(v)$-values sum up to 0). Moreover, if $f(v,w) > 0$, then $\pi(v) \leq \pi(w) - \tau(v,w)$ and if $f(t,s) > 0$ then $\pi(t) = T$.

Putting this together leads to:

$$\begin{aligned}
\mathrm{val}([\Gamma_g]^T) &= T \cdot |g| - \sum_{e \in E(G)} \tau(e) \cdot g(e) \\
&= - \sum_{e \in E(\tilde{G})} \tau(e) \cdot f(e) \\
&= \sum_{e=(v,w) \in E(\tilde{G})} (\pi(w) - \tau(e) - \pi(v)) f(e) \\
&= \sum_{e=(v,w) \in E(G)} (\pi(w) - \tau(e) - \pi(v)) f(e) \\
&= \sum_{\substack{e=(v,w) \in E(G): \\ \pi(w) - \tau(e) > \pi(v)}} (\pi(w) - \tau(e) - \pi(v)) f(e) \\
&= \sum_{\substack{e=(v,w) \in E(G): \\ \pi(w) - \tau(e) > \pi(v)}} (\pi(w) - \tau(e) - \pi(v)) u(e).
\end{aligned}$$

The last equality follows from the fact that if $f(v,w) < u(v,w)$, then $(v,w) \in E(\tilde{G}_{u,f})$ and therefore $\pi(w) \leq \pi(v) + \tau(v,w)$.

This shows that the s-t-cut over time that is defined by the numbers $\pi(v)$ has capacity $\mathrm{val}([\Gamma_g]^T)$, and since any cut capacity is an upper bound for the value of any s-t-flow over time, this concludes the proof. □

Corollary 2.8 (Continuous version of the dynamic max-flow min-cut theorem) Let (G, u, τ) be a dynamic flow network, let $s \neq t$ be two vertices of G, and let $T \geq 0$. Then, the value of a maximum continuous s-t-flow over time with time horizon T in (G, u, τ) equals the value of a mimimum s-t-cut in (G, u, τ) with time horizon T.

Remark: Note that the maximum s-t-flow over time that is computed by the FORD-FULKERSON-ALGORITHM does not make use of storing flow in nodes. Therefore, it does not matter for this purpose if we are allowed to store flow.

In a variation of the maximum flow over time problem, the MAXIMUM PATH FLOW OVER TIME PROBLEM, one asks for the maximum amount of flow that can be shipped from s to t under the restriction that the whole flow is sent along one single path. This problem can be solved in polynomial time using the construction described in section 3.1.

3 Quickest Flows over Time

Instead of asking for a maximum s-t-flow with a given time horizon we may ask for a smallest possible time horizon for which a dynamic s-t-flow of a given value X exists. We call this minimum time horizon T_X. Computing T_X for a given value X is the Quickest Flow over Time Problem.

In this section, we again restrict ourselves to the continuous version.

Theorem 3.1 Let (G, u, τ) be a dynamic flow network with two vertices s and t. For given values X and ε we can compute a time horizon $\tilde{T}_X$ with $\tilde{T}_X \le T_X + \varepsilon$ for which a dynamic s-t-flow of size X exists by computing $O(\log(X + \sum_{e \in E(G)} \tau(e) \cdot u(e)) - \log K - \log \varepsilon)$ maximum dynamic s-t-flows where K is the minimum of all edge capacities.

Proof: We know that for a given time horizon T the maximum dynamic s-t-flow has value $T \cdot |f| - \sum_{e \in E(G)} \tau(e) \cdot f(e)$ for an appropriate static s-t-flow f. Let f^* be a maximum static s-t-flow in G. We may assume that t is reachable from s in G (otherwise the problem is trivial) which implies $|f^*| \ge K$. So for $T_X^* := \frac{1}{K}(X + \sum_{e \in E(G)} \tau(e) \cdot u(e))$, and a maximum static s-t-flow f^* we have

$$T_X^* \cdot |f^*| - \sum_{e \in E(G)} \tau(e) \cdot f^*(e) \ge X \frac{|f^*|}{K} + \frac{|f^*|}{K} \sum_{e \in E(G)} \tau(e) \cdot u(e) - \sum_{e \in E(G)} \tau(e) \cdot f^*(e) \ge X.$$

This means that T_X^* is an upper bound on T_X. Since $T_X \ge 0$, we can apply binary search and get the absolute approximation claimed in the theorem by subdividing the interval $[0, T_X^*]$ into interval of length *varepsilon*. □

Remarks:

- The binary search only provides an absolute approximation. To prove a relative approximation factor, a positive lower bound l_X for T_X must be known (e.g. the minimum transit time of an s-t-path) contributing an additional term of $O(\log l_X)$ to the running time.
- If X and all transit times and edge capacities are integral then T_X is a fraction whose denominator is bounded by the minimum capacity of a (static) s-t-flow because of $T_X = (X + \sum_{e \in E(G)} \tau(e) \cdot f(e))/|f|$ for some static (integral) s-t-flow f (see Fleischer and Tardos [1998]). Hence, in this case, we can even compute T_X exactly by binary search.
- For discrete flows flows over time, only integral time horizons are allowed, so we can solve the Quickest Flow over Time Problem exactly by binary search.

The approximation algorithm proposed in the above proof is polynomial but not strongly polynomial. However, there is also a strongly polynomial algorithm that solves the problem exactly:

Theorem 3.2 The Quickest Flow over Time Problem can be solved in time $O((m \log m(m + n \log n))^2)$ where n is the number of nodes and m is the number of edges in the dynamic flow network.

Proof: We use parametric search (see Megiddo [1979]). For a fixed time horizon T we can apply the Ford-Fulkerson Algorithm to decide if we can send flow of size X in time T from s to t.

The Ford-Fulkerson Algorithm mainly consists of Orlin's Algorithm that computes a circulation of minimum cost. Now, the idea is to run Orlin's Algorithm without specifying T in advance. During the algorithm we only keep track of an interval $[\mu, \nu]$ that contains an optimum value of of T. We may start with $\mu = 0$ and $\nu = \frac{1}{K}(X + \sum_{e \in E(G)} \tau(e) \cdot u(e))$ where K is the minimum capacity of an edge in G. Then, instead of real numbers, affine-linear functions of the type $a + b \cdot T$ occur as edge costs. In order to see how we can handle such functions we summarize what kind of operations Orlin's Algorithm performs on edge costs:

- Addition and substraction of two edge costs,
- Multiplication of an edge cost with a real number,
- Comparison of two edge costs.

The first two operations can be performed easily for affine-linear functions. If we compare two edge costs we have to decide if $a + b \cdot T \geq 0$ for some numbers a and b (where $b \neq 0$ is the difficult case). To do this we compute $T^* = -\frac{a}{b}$. If T^* is outside $[\mu, \nu]$, then either $a + b \cdot T \geq 0$ is true for all $T \in [\mu, \nu]$ or $a + b \cdot T \geq 0$ is false for all $T \in [\mu, \nu]$, so we can continue.

On the other hand, if $T^* \in [\mu, \nu]$, then we solve the maximum dynamic flow problem for $T = T^*$. Let X^* be the value of the result. Then, we have two cases:

Case 1: $X^* \geq X$. Then we know that the optimum time is at most T^*, and we replace $[\mu, \nu]$ by $[\mu, T^*]$. Then we can also decide if $a + b \cdot T \geq 0$ is true for all $T \in [\mu, T^*]$.

Case 2: $X^* < X$. then, we know that the optimum time is bigger that T^*, and we replace $[\mu, \nu]$ by $[T^*, \nu]$. Also in this case we can decide if $a + b \cdot T \geq 0$ holds for all $T \in [T^*, \nu]$.

In the worst case, we have to replace all comparisons of edge weights by a complete minimum-cost flow computation. Since Orlin's Algorithm has running time $O(m \log m(m + n \log n))$ there are also $O(m \log m(m + n \log n))$ comparisons which leads to the running time claimed in the theorem. $\square$

Remark: Our running time analysis was quite pessimistic: Not all of the $O(m \log m(m+n \log n))$ steps of Orlin's Algorithm are comparisons of edge weights, and most of the edge weights are still real numbers. A more sophisticated analysis leads to a running time of $O(m^2 \log^3 n(m + n \log n))$ (see Burkard, Dlaska, and Klinz [1993]).

3.1 Quickest path problem

In some applications (e.g. transmission of data through a computer network or some evacuation problems) flow should be sent along one path only. This leads to the Quickest Path Problem, a variation of the Quickest Flow over Time Problem in which we ask for an s-t-path through which we can send a given amount of flow as fast as possible. Chen and Chin [1990], Rosen, Sun, and Xue [1991], and Martins and dos Santos [1997] propose algorithms that solve this problem in polynomial time:

Proposition 3.3 The QUICKEST PATH PROBLEM can be solved in time $O(rm + rn \log n)$ where n is the number of nodes, m the number of edges, and r the number of different edge capacities in the input network.

Proof: (Martins and dos Santos [1997]) Let (G, u, τ) be the the given dynamic flow network. Note that when we ship flow of size X through an s-t-path P, it arrives at t within a time horizon of $\tau(P) + \frac{X}{u_{\min}(P)}$ where $u_{\min}(P) = \min\{u(e) \mid e \in E(P)\}$.

Let the edge capacities be denoted by $u_1, \dots, u_r$. For $i \in \{1, \dots, r\}$ we compute a shortest s-t-path P_i in G_i where $V(G) := V(G)$ and $E(G_i) := \{e \in E(G) \mid u(e) \geq u_i\}$ (with respect to the edge lengths $\tau(e)$). For each such path P_i we compute $\tau(P_i) + \frac{X}{u_{\min}(P_i)}$ and choose a path P_l that minimizes this value. With a fast implementation of Dijkstra's algorithm (see Dijkstra [1959] and Fredman and Tarjan [1987]), each of the r shortest-path computations can be performed in time $O(m + n \log n)$.

We claim that sending the flow through P_l is optimal. To see this, assume that P is an s-t-path and let u_k (for an appropriate $k \in \{1, \dots, r\}$) be the minimum edge capacity in P. Then, we have $\tau(P_k) \leq \tau(P)$ and $\min(P_k) \geq u_k = \min(P)$. This implies $\tau(P_l) + \frac{X}{u_{\min}(P_l)} \leq \tau(P_k) + \frac{X}{u_{\min}(P_k)} \leq \tau(P) + \frac{X}{u_{\min}(P_k)} \leq \tau(P) + \frac{X}{u_{\min}(P)}$. □

4 Earliest Arrival Flows

Definition 4.1 A (discrete or continuous) dynamic s-t-flow f is an *earliest arrival s-t-flow* if it maximizes $ex_f(t, \theta)$ for all $\theta \in \mathrm{dom}(f)$.

Sometimes, earliest arrival flows are also called *universally quickest flows.*

Figure 5 shows that a maximum flow over time is not necessarily an earliest arrival flow. Assume that in this example all edge capacities and traversal times are 1. Let the time horizon be $T = 4$. Then, for a continuous flow it is easy to check that the value of a maximum flow is 2. A flow with this value can be found by sending flow of value 1 along the path through s,v, and t starting a time 0 and stopping a time 2. This would be an earliest arrival flow as well. But if we send instead during the time interval $[0, 1]$ the flow along the path through s, v, w, and t (and then switch to the path through s,v, and t) we also get a maximum flow, but it is not an earliest arrival flow, because at time 3 no flow has reached t.

The surprising fact is that there is always a maximum flow that is also an earliest arrival flow.

For discrete flows, this can be shown quite easily by considering the time-expanded network: For $\theta \in \{0, \dots, T\}$ let X_θ be the maximum flow that can be sent to t within the time horizon θ. We set b-values to the nodes in the time-expanding network by defining $b((t, 0)) = 0$ and $b((t, \theta)) := -(X_\theta - X_{\theta-1})$ for $\theta > 0$ and $b((s, 0)) := X_T$. For all remaining nodes the b-values are 0. Obviously, a b-flow in the time-expanded network corresponds to an earliest arrival flow.

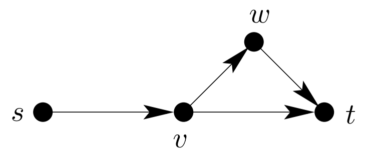

Figure 5: A network in which a maximum flow exists that is not an earliest flow.

Moreover, there is always a feasible b-flow in the time-expanded network: This is due to the fact that such a b-flow exists if and only if for all vertex sets U the total capacity on outgoing edges is at least the sum of all b-values in U (Gale [1957]; see also the summary on static flows in the introduction). This is obviously true for all vertex sets U for which a node (t,θ) exists with $(t,\theta) \in U$ and $(t,\theta+1) \notin U$ because the edge $((t,\theta),(t,\theta+1))$ has infinite capacity. But if $U \cap \{(t,0),\ldots,(t,T)\} = \{(t,\theta),(t,\theta+1),\ldots,(t,T)\}$ for some appropriate θ, then the sum of the b-values in U is $X_{\theta-1}$, and since there is a dynamic s-t-flow with time horizon $\theta-1$ and value $X_{\theta-1}$, the capacity of the edges leaving U must be at least $X_{\theta-1}$.

This construction also leads to an algorithm (whose running time, however, depends on T). The continuous case and a description of a polynomial-time approximation scheme will be the topic of the next sections.

4.1 Earliest Arrival Algorithm

Definition 4.2 Let (G,u) be a static flow network with two nodes s and t, and let f be a (static) s-t-flow in (G,u). Let $\mathcal{P}$ be a set of s-t-paths in $\overset{\leftrightarrow}{G}$ with numbers $f_P \geq 0$ for $P \in \mathcal{P}$. Then $(f_P)_{P\in\mathcal{P}}$ is called a *generalized path decomposition* of f if for each $e \in E(G)$:

$$f(e) = \sum_{P\in\mathcal{P}:e\in E(P)} f_P - \sum_{P\in\mathcal{P}:\overleftarrow{e}\in E(P)} f_P$$

Note that, in contrast to flow decomposition we do not allow cycles in a generalized path decomposition (that is why there are s-t-flows without generalized path decomposition).

Definition 4.3 Let (G, u, τ) be a dynamic flow network, let s and t be two vertices, and $T \geq 0$ a time horizon. Let f be a static s-t-flow in (G, u) with generalized path decomposition $\Gamma_f = (f_P)_{P \in \mathcal{P}}$. Then, the corresponding *generalized temporally repeated s-t-flow* $[\Gamma_f]^T$ is defined by

$$[\Gamma_f]^T_e(\theta) := \sum_{P \in \mathcal{P}_e(\theta)} f_P - \sum_{P \in \mathcal{P}_{\overleftarrow{e}}(\theta)} f_P \quad \text{for } e = (v, w) \in E(G), \theta \in [0, T)$$

where

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

and

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

for $e = (v, w) \in E(G)$.

Remark: Generalized temporally repeated s-t-flows are not necessarily feasible s-t-flows over time. See Figure 6 for an example. Let all capacities and transit times be 1. The edge labels in the first picture are flow values of a static s-t-flow f. The two other pictures show the two paths of a generalized path decomposition of f. Assume that the time horizon is sufficiently large, e.g. $T = 5$. Then, for $\theta \in (0, 2)$ the generalized temporally repeated flow on the edge where f is 0 will be negative. Note that we start subtracting flow on this edge due to the second path at time 0 because for reverse edges we have negative transit times.

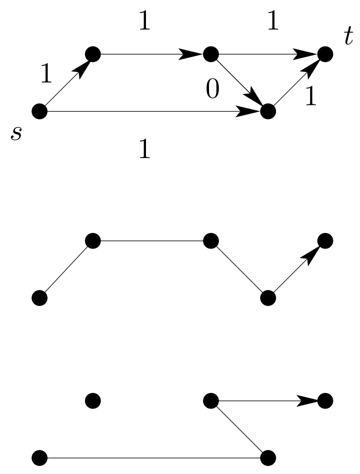

Figure 6: A flow network with a static s-t-flow and the two paths of a generalized path decomposition.

Earliest Arrival Algorithm

Input: A dynamic flow network (G, u, τ), two vertices s and t, and a time horizon $T \geq 0$.

Output: A generalized temporally repeated flow with time horizon T

```
①   Set $f(e) = 0$ for all $e \in E(G)$.
    Set $\mathcal{P} = \emptyset$.

②   WHILE $(\mathrm{dist}_{G_{f,u}}(s,t) < T)$
    {
        Find a shortest $s$-$t$-path $P$ in $G_{f,u}$ (with $\tau(e)$ as edge weights).
        Let $\rho$ be the minimal residual edge capacity on $P$.
        Augment $f$ along $P$ by $\rho$.
        Set $f_P = \rho$. Add $P$ to $\mathcal{P}$.
    }
③   Output $[(f_P)_{P\in\mathcal{P}}]^T$.
```

For the analysis of the algorithm we denote the sequence of shortest paths in a run by $P_1, \dots, P_k$. Let f_i be the static flow f after i augmentations ($i \in \{0, \dots, k\}$).

Lemma 4.1 f_i is a minimum-cost b_i-flow in (G, u) (for all $i \in \{0, \dots, k\}$) where $b_i(s) := |f_i|$, $b_i(t) := -|f_i|$ and $b_i(v) = 0$ for $v \in V(G) \setminus \{s, t\}$.

Proof: We start with a zero-flow which is clearly a b-flow for $b = 0$. Then, in each iteration we augment f_i by the minimum residual capacity of P, so the flow cannot become negative and the capacity constraints are respected. Since we augment along s-t-paths, the flow conservations rule for nodes $v \in V(G) \setminus \{s, t\}$ is met.

Now assume that f_i is not a minimum-cost flow, and assume that i is the first index with this property. Since all edge costs are nonnegative, i must be positive. There must be a negative cycle C in G_{u,f_i}. Consider the graph $H = (V(G), E(C)\dot\cup E(P_i))$ in which we remove pairs of reverse edge but count parallel edges twice. Then, $\tau(E(H)) = \tau(E(C)) + \tau(E(P_i)) < \tau(E(P_i))$. We have $E(H) \subseteq E(G_{u,f_{i-1}})$ because the only edges in $E(G_{u,f_i}) \setminus E(G_{u,f_{i-1}})$ are reverse edges of edges in P_i. If we add an edge (t, s) to H, then the resulting graph is Eulerian. Therefore, the edge set of H can be written $E(H) = E_1\dot\cup E(P)$ where E_1 is the edge set of a set of cycles and P is an s-t-path. By assumption, f_{i-1} has minimum cost, so $\tau(E_1) \geq 0$. By the choice of P_i, we have $\tau(P_i) \leq \tau(P)$. This leads to $\tau(E(H)) = \tau(E_1) + \tau(E(P)) \geq \tau(P_i)$ which is a contradiction to $\tau(E(H)) < \tau(E(P_i))$. □

Lemma 4.2 For all $v \in V(G)$ and $i \in \{0, \dots, k\}$, $\mathrm{dist}_{G_{u,f_{i+1}}}(s, v) \geq \mathrm{dist}_{G_{u,f_i}}(s, v)$ and $\mathrm{dist}_{G_{u,f_{i+1}}}(v, t) \geq \mathrm{dist}_{G_{u,f_i}}(v, t)$.

Proof: Assume that there is an iteration i with $\mathrm{dist}_{G_{u,f_{i+i}}}(s, v) < \mathrm{dist}_{G_{u,f_i}}(s, v)$. Let Q be a shortest s-v-path in $G_{u,f_{i+i}}$. Consider the graph $H = (V(G), E(P_{i+1})\dot\cup E(Q))$ in which we remove again pairs of reversing edges but count parallel edges twice. Then, H is a subgraph of G_{u,f_i}. H consists of an s-t-path P, an s-v-path Q' which is edge-disjoint from P and a (possibly empty) set of cycles. The cycles have nonnegative weight (since f_i has minimum cost), so we have $\tau(E(H)) \geq \tau(E(P)) + \tau(E(Q'))$. On the other hand, we have $\tau(E(H)) = \tau(E(P_{i+1})) + \tau(E(Q))$. This leads to a contradiction because $\tau(E(P_{i+1})) \leq \tau(E(P))$ and $\tau(E(Q)) < \tau(E(Q'))$.

An analogous argument shows that the distances from any vertex v to t cannot decrease. □

Lemma 4.3 Let $i \in \{0, \dots, k\}$. Let $\tilde{G}_i$ result from G by adding an edge (t, s) with infinite capacity and edge cost $-\theta_i$ where $\theta_i = \mathrm{dist}_{G_{f_i}}(s,t)$, and let $f'_i(e) = f_i(e)$ for $e \in E(G)$ and $f'_i(t,s) = |f_i|$. Then, f'_i is a minimum-cost circulation in $\tilde{G}_i$.

Proof: To prove optimality we only have check that there is no negative cycle in the residual graph. Since f_i is a minimum-cost b_i-flow, such a cycle would contain the additional edge (t, s). But all s-t-paths in G_{u,f_i} have at least length θ, so this edge cannot be in a negative augmenting cycle. □

Theorem 4.4 The Earliest Arrival Algorithm computes an s-t-flow over time with time horizon T.

Proof: Consider an edge $e = (v, w)$ and a time $\theta \in [0, T)$. We have to show that $0 \leq [(f_P)_{P\in\mathcal{P}}]^T_e(\theta) \leq u(e)$. Let

$$l := \max\{i \in \{0, \dots, k\} \mid \mathrm{dist}_{G_{u,f_i}}(s, v) \leq \theta \wedge \mathrm{dist}_{G_{u,f_i}}(v, t) < T - \theta\}.$$

We can conclude:

$$\begin{aligned}
[(f_P)_{P\in\mathcal{P}}]^T_e(\theta) &= \sum_{P\in\mathcal{P}_e(\theta)} f_P - \sum_{P\in\mathcal{P}_{\overleftarrow{e}}(\theta)} f_P \\
&= \sum_{i\in\{1,\dots,l+1\}: e\in E(P_i)} f_{P_i} - \sum_{i\in\{1,\dots,l+1\}: \overleftarrow{e}\in E(P_i)} f_{P_i} \\
&= f_{l+1}(e).
\end{aligned}$$

Since we have already seen that f_{l+1} is a feasible static s-t-flow, this proves $0 \leq [(f_P)_{P\in\mathcal{P}}]^T_e(\theta) \leq u(e)$ □

Theorem 4.5 The Earliest Arrival Algorithm computes an earliest arrival s-t-flow with time horizon T.

Proof: Let $g = [(f_P)_{P\in\mathcal{P}}]^T$ be the output of the algorithm. Fix a time $\theta \in [0, T)$. We have to show that $\mathrm{ex}_g(t, \theta)$ is maximal among all s-t-flows over time with time horizon T. Let $l := \max\{i \in \{0, \dots, k\} \mid \mathrm{dist}_{G_{u,f_i}}(s,t) \leq \theta\}$. When we augment along P_i, this increases the incoming flow of t up to time θ by $(\theta - \tau(P_i)) \cdot f_{P_i}$. In particular only the first $l+1$ augmentations change $\mathrm{ex}_g(t, \theta)$, and we have:

$$\mathrm{ex}_g(t, \theta) = \sum_{i=1}^{l+1} (\theta - \tau(P_i)) \cdot f_{P_i} = \theta \cdot |f_{l+1}| - \sum_{e \in E(G)} \tau(e) \cdot f_{l+1}(e).$$

Moreover, Lemma 4.3 implies that f_{l+1} maximizes the value $\theta \cdot |f| - \sum_{e \in E(G)} \tau(e) \cdot f(e)$. This means that $\mathrm{ex}_g(t, \theta)$ is maximum because it is maximum among all flows with time horizon θ which are temporally repeated flow and we have seen in the analysis of the FORD-FULKERSON ALGORITHM that there is always a temporally repeated flow that is a maximum flow over time. □

Remark: The algorithm can be seen as a variant of the SUCCESSIVE SHORTEST PATH ALGORITHM. As for the SUCCESSIVE SHORTEST PATH ALGORITHM, there is no polynomial upper bound for the number of augmentations, so the EARLIEST ARRIVAL ALGORITHM is not a polynomial-time algorithm (see Zadeh [1973] for an example where the SUCCESSIVE SHORTEST PATH ALGORITHM has exponential running time). In fact, for abitrary real edge capacities, it does not necessarily terminate at all (while for integral capacties it has at least a final running time).

4.2 Scaled Earliest Arrival Algorithm

In the following algorithm, we will reduce the capacities of edges in the residual graph. To do this, we will introduce also lower bounds $l(e)$ on the flow on a edge e. Then, the residual capacity of the reverse edge of e is $f(e) - l(e)$. The residual graph with respect to these lower bounds on the flow values contains all edges in $\overset{\leftrightarrow}{G}$ with positive residual capacity and is denoted by $G_{l,u,f}$. We will, of course, start with $l(e) = 0$.

We have to restrict the capacities to integral numbers.

SCALED EARLIEST ARRIVAL ALGORITHM	
Input:	A dynamic flow network (G, u, τ) with integral capacities $u(e)$ for $e \in E(G)$, two vertices s and t, a time horizon $T \geq 0$, and a number $\varepsilon > 0$.
Output:	A generalized temporally repeated flow with time horizon T.

```
①    For $e \in E(G)$ set $f(e) = 0$, $\tilde{u}(e) = u(e)$ and $\tilde{l}(e) = 0$.
     Set $\mathcal{P} = \emptyset$, $\sigma = 0$, and $\Delta = 1$.

②    WHILE ($\mathrm{dist}_{G_{\tilde{l},\tilde{u},f},\tau}(s,t) < T$)
     {
         Set $\sigma = 0$.
         WHILE ($\sigma < m\Delta/\varepsilon$ and $\mathrm{dist}_{G_{\tilde{l},\tilde{u},f},\tau}(s,t) \leq T$)
         {
             Find a shortest $s$-$t$-path $P$ in $G_{\tilde{l},\tilde{u},f}$ (with $\tau(e)$ as edge weights).
             Let $\rho$ be the minimal residual edge capacity on $P$.
             Augment $f$ along $P$ by $\rho$.
             Set $f_P = \rho$. Add $P$ to $\mathcal{P}$.
             Set $\sigma = \sigma + \rho$.
         }
         Set $\Delta = 2 \cdot \Delta$.
         Set $\tilde{u}(e) = \tilde{u}(e) - ((\tilde{u} - f) \bmod \Delta)$ and
         $\tilde{l}(e) = \tilde{l}(e) + ((f - \tilde{l}) \bmod \Delta)$ for $e \in E(G)$.
     }

③    Output $[(f_P)_{P \in \mathcal{P}}]^T$.
```

Remark: At the end of an iteration of the outer loop, all residual capacities are integral multiples of Δ.

For the analysis we will use the same notation as for the EARLIEST ARRIVAL ALGORITHM.

Theorem 4.6 The SCALED EARLIEST ARRIVAL ALGORITHM computes an s-t-flow with time horizon T.

Proof: The proof is the same as for Theorem 4.4. We only decrease capacities but the flows f_l are still feasible static s-t-flows and the lengths of the paths do not decrease. Therefore, the capacity constraints are met and the output is an s-t-flow with time horizon T. □

Theorem 4.7 The SCALED EARLIEST ARRIVAL ALGORITHM can be implemented to run in time $O(\frac{m}{\varepsilon}(m + n \log n) \log U)$ where $n = |V(G)|$, $m = |E(G)|$, and $U = \max\{u(e) \mid e \in E(G)\}$.

Proof: The outer loop of step ② is performed at most $\log_2 U$ times because Δ is doubled in each iteration and the loop stops if $\Delta \geq U$.

For a fixed value of Δ the interior loop of ② is performed at most $\frac{m}{\varepsilon}$ times because ρ is always an integral multiple of Δ (since all capacities are integral multiples of Δ after scaling).

In each iteration of the interior loop, the most time-consuming part is the computation of the shortest s-t-path which can be solved in time $O(m + n \log n)$ by Dijkstra's algorithm. □

Let there be $k + 1$ iterations of the outer loop (called phases) in the Scaled Earliest Arrival Algorithm, numbered from 0 to k, so in iteration i we have $\Delta = 2^i$. We use the following notation:

- Let $\Gamma_i = (f_P)_{P \in \mathcal{P}}$ at the end of phase i (where we denote by Γ_{-1} an empty flow decomposition).
- Let T_i be the length of the longest path in $\mathcal{P}$ at the end of phase i.
- Let g_i be the static flow at the end of phase i.
- Let $\tilde{u}_i$ be the capacity function and $\tilde{l}_i$ the lower bound function at the beginning of phase i.

Lemma 4.8 For $i \in \{0, \ldots, k-1\}$, let Λ_i^* be a generalized path decomposition corresponding to an earliest arrival flow in $G_{\tilde{l}_i, \tilde{u}_i, g_i}$, and let $\tilde{\Lambda}_i$ be a generalized path decomposition corresponding to an earliest arrival flow in $G_{\tilde{l}_{i+1}, \tilde{u}_{i+1}, g_i}$. Then, for all $\theta \in [0, T]$ we have

$$\mathrm{ex}_{[\Lambda_i^*]^T}(t, \theta) - \mathrm{ex}_{[\tilde{\Lambda}_i]^T}(t, \theta) \leq \varepsilon \left(\mathrm{ex}_{[\Gamma_i]^T}(t, \theta) - \mathrm{ex}_{[\Gamma_{i-1}]^T}(t, \theta) \right).$$

Proof: Claim: We have $\mathrm{ex}_{[\Lambda_i^*]^T}(t, \theta) - \mathrm{ex}_{[\tilde{\Lambda}_i]^T}(t, \theta) \leq m 2^i (\theta - T_i)$.

Proof of the claim: We may transform the static flow corresponding to Λ_i^* into a flow meeting the stricter bounds $\tilde{l}_{i+1}$ and $\tilde{u}_{i+1}$ by iteratively de-augmenting flow by 2^i on s-t-paths. With each reduction, the capacity constraint is met in at least one additional edge, so at most m reductions are necessary. Since each of the paths has length at least T_i, this leads of a flow with generalized path decomposition Π_i such that

$$\mathrm{ex}_{[\Pi_i]^T}(t, \theta) \geq \mathrm{ex}_{[\Lambda_i^*]^T}(t, \theta) - m 2^i (\theta - T_i).$$

Since $\mathrm{ex}_{[\tilde{\Lambda}_i]^T}(t, \theta) \geq \mathrm{ex}_{[\Pi_i]^T}(t, \theta)$, this proves the claim.

In the i-th phase, the sum of all ρ-values is at least $m\Delta/\varepsilon = m 2^i/\varepsilon$, and all augmenting paths have a length of at most T_i. Therefore, we have

$$\mathrm{ex}_{[\Gamma_i]^T}(t, \theta) - \mathrm{ex}_{[\Gamma_{i-1}]^T}(t, \theta) \geq \frac{m 2^i}{\varepsilon} (\theta - T_i).$$

Together with the above claim, this proves the lemma. □

Theorem 4.9 (Hoppe and Tardos [1994]) Let $\Gamma_k = (f_P)_{P\in\mathcal{P}}$ be the generalized path decomposition of the output of the Scaled Earliest Arrival Flow Algorithm. For $\theta \in [0,T)$, let X_θ be the maximum value of an s-t-flow over time with time horizon θ. Then, we have

$$X_\theta \le (1+\varepsilon)\mathrm{ex}_{[\Gamma_k]^T}(t,\theta).$$

Proof: If we did not stop a phase of the algorithm as soon as $\sigma \ge m\Delta/\varepsilon$, we would compute in phase $i+1$ an earliest arrival flow with respect to the lower bounds $\tilde{l}_{i+1}$ and the capacities $\tilde{u}_{i+1}$. Therefore we have (with the same notation as in the previous proof):

$$\mathrm{ex}_{[\Gamma_{i+1}]^T}(t,\theta) - \mathrm{ex}_{[\Gamma_i]^T}(t,\theta) + \mathrm{ex}_{[\Lambda^*_{i+1}]^T}(t,\theta) = \mathrm{ex}_{[\tilde{\Lambda}_i]^T}(t,\theta).$$

Of course, the same is true for the first phase in which the capacities are not yet scaled, so $X_\theta = \mathrm{ex}_{[\Gamma_0]^T}(t,\theta) + \mathrm{ex}_{[\Lambda^*_0]^T}(t,\theta)$.

We can conclude:

$$\begin{aligned}
X_\theta &= \mathrm{ex}_{[\Gamma_0]^T}(t,\theta) + \mathrm{ex}_{[\Lambda^*_0]^T}(t,\theta)\\
&= \mathrm{ex}_{[\Gamma_k]^T}(t,\theta) + \sum_{i=0}^{k-1}\Big(\mathrm{ex}_{[\Gamma_i]^T}(t,\theta) - \mathrm{ex}_{[\Gamma_{i+1}]^T}(t,\theta)\Big) + \mathrm{ex}_{[\Lambda^*_0]^T}(t,\theta)\\
&= \mathrm{ex}_{[\Gamma_k]^T}(t,\theta) + \sum_{i=0}^{k-1}\Big(\mathrm{ex}_{[\Lambda^*_{i+1}]^T}(t,\theta) - \mathrm{ex}_{[\tilde{\Lambda}_i]^T}(t,\theta)\Big) + \mathrm{ex}_{[\Lambda^*_0]^T}(t,\theta)\\
&= \mathrm{ex}_{[\Gamma_k]^T}(t,\theta) + \sum_{i=0}^{k-1}\Big(\mathrm{ex}_{[\Lambda^*_i]^T}(t,\theta) - \mathrm{ex}_{[\tilde{\Lambda}_i]^T}(t,\theta)\Big) + \mathrm{ex}_{[\Lambda^*_k]^T}(t,\theta)\\
&= \mathrm{ex}_{[\Gamma_k]^T}(t,\theta) + \sum_{i=0}^{k-1}\Big(\mathrm{ex}_{[\Lambda^*_i]^T}(t,\theta) - \mathrm{ex}_{[\tilde{\Lambda}_i]^T}(t,\theta)\Big)\\
&\le \mathrm{ex}_{[\Gamma_k]^T}(t,\theta) + \sum_{i=0}^{k-1}\varepsilon\Big(\mathrm{ex}_{[\Gamma_i]^T}(t,\theta) - \mathrm{ex}_{[\Gamma_{i-1}]^T}(t,\theta)\Big)\\
&= \mathrm{ex}_{[\Gamma_k]^T}(t,\theta) + \varepsilon\cdot\mathrm{ex}_{[\Gamma_{k-1}]^T}(t,\theta)\\
&\le (1+\varepsilon)\mathrm{ex}_{[\Gamma_k]^T}(t,\theta)
\end{aligned}$$

□

5 Dynamic b-flows

Definition 5.1 Let (G, u, τ) be a dynamic flow network, $T \geq 0$, and let $b : V(G) \to \mathbb{R}$ be a mapping with $\sum_{v \in V(G)} b(v) = 0$. A *discrete (continuous) b-flow f over time with time horizon T* is a discrete (continuous) flow over time f such that $\mathrm{ex}_f(v, T) = -b(v)$ and $\mathrm{ex}_f(v, \theta) \geq \min\{-b(v), 0\}$ for all $v \in V(G)$ and $\theta \in dom(f)$.

One can also consider special cases of dynamic b-flows where all b-values are 0 (*circulation over time*) but there are only non-trivial dynamic circulation if there are cycles of total length 0, so, different to static flows, this special case is not very interesting.

5.1 Transshipment Problems

In the Dynamic Transshipment Problem we are given (G, u, τ), T and the b-values and ask if there is a b-flow over time with time horizon T. In the Quickest transshipment Problem we are given (G, u, τ) and the b-values and ask for the smallest T such that there is a b-flow over time with time horizon T. A quickest transshipment problem with one sink only is called *evacuation problem.*

Remark: Different to the static case we cannot reduce the transshipment problem with many sources and sinks to the quickest flow problem (with only one source and sink) by simply adding a super source and a super sink. For example, for a source we cannot bound the amount of flow sent from the super source to the source. In the static case this can be done by putting a finite capacity on the edge, but in the dynamic case the capacity is not an upper bound for the total amount of flow sent through an edge.

In the remainder of this section, we will consider discrete instances.

5.2 Lexicographically maximum flows over time

Following Hoppe and Tardos [1994] and Hoppe and Tardos [2000] we will solve the quickest transshipment problem by computing a lexicographically maximum flow in a core subroutine:

Definition 5.2 Let (G, u, τ) be a dynamic flow network, $T \geq 0$ a time horizon, and $(s_1, \ldots s_k)$ be an ordered set of nodes of G. Let $S = \{s_1, \ldots, s_k\} = S_+ \dot\cup S_-$ (where we call the elements of S_+ sources and the elements of S_- sinks). Then, a *lexicographically maximum flow with time horizon T* is a flow over time f with time horizon T with $\mathrm{ex}_f(v, T) \leq 0$ for $v \in S_+$, $\mathrm{ex}_f(v, T) \geq 0$ for $v \in S_-$, $\mathrm{ex}_f(v, \theta) \geq 0$ and $\mathrm{ex}_f(v, T) = 0$ for $v \in V(G) \setminus S$ and $\theta \in \mathrm{dom}(f)$ such that $(-\mathrm{ex}_f(s_1, T), \ldots, -\mathrm{ex}_f(s_k, T))$ is lexicographically maximum.

We may assume that the sources have no incoming edges and the sinks have no outgoing edges. Moreover we assume that there are no cycles of zero transit time.

Lexicographically Maximum Dynamic Flow Algorithm

Input: A dynamic flow network (G, u, τ), a time horizon $T \geq 0$, and a vector $(s_1, s_2, \ldots, s_k)$ of nodes such that $\{s_1, \ldots, s_k\} = S_+ \dot\cup S_-$.

Output: A generalized path decomposition of a static flow.

```
①    Set $V(G') = V(G) \dot\cup \{s\}$.
     Set $E_{k+1} = E(G') \cup \{(s, s_j) \mid s_j \in S_+\}$ where $u(s, s_j) := \infty$ and $\tau(s, s_j) = 0$.
     Let $g_{k+1}$ be the static zero-flow in $(V(G), E_{k+1})$.
     Set $\Gamma_{k+1} = \emptyset$.
②    FOR $(i = k$ to $1)$
     {
         Set $E_i = E_{i+1}$ and $G^i = (V(G'), E_i)$.
         IF $(s_i \in S_-)$
         {
             $E_i = E_i \dot\cup \{(s_i, s)\}$.
             Set $u(s_i, s) = \infty$ and $\tau(s_i, s) = -T - 1$.
             Compute a minimum-cost circulation $f_i$ in $G^i_{g_{i+1},u}$ (w.r.t. $\tau$).
         }
         IF $(s_i \in S_+)$
         {
             $E_i = E_i \setminus \{(s, s_i)\}$.
             Compute a minimum-cost maximum $s$-$s_i$-flow $f_i$ in $G^i_{g_{i+1},u}$ (w.r.t. $\tau$).
         }
         Augment $g_{i+1}$ by $f_i$ (i.e. augment $g_{i+1}(e)$ by $f_i(e)$ for $e \in E(\overset{\leftrightarrow}{G'})$).
         Let $g_i$ be the result of the augmentation.
         Let $\Delta_i$ be a (standard) path decomposition of the restriction of $f_i$
         to $\overset{\leftrightarrow}{G'} \setminus \delta^-_{\overset{\leftrightarrow}{G'}}(s)$.
         Set $\Gamma_i = \Gamma_{i+1} \cup \Delta_i$.
     }
③    Output $\Gamma_1$.
```

Lemma 5.1 In each iteration of the Lexicographically Maximum Dynamic Flow Algorithm, g_i is a minimum-cost circulation in G^i.

Proof: At the beginning, the zero-flow g_{k+1} is a minimum-cost circulation because all edge costs are nonnegative. We have to show that this remains true during each iteration of the loop.

If s_i is a sink, f_i is a minimum-cost circulation in the residual graph. Therefore, g_i is a circulation, and $G^i_{g_i,u}$ does not contain a negative cycle (f_i could have been computed by the Minimum Mean Cycle Cancelling Algorithm).

If s_i is a source, then the value of the s-s_i-flow f_i will be $g_{i+1}(s, s_i)$ because g_{i+1} was a circulation and s_i does not have any incoming edges in G. Therefore g_i is also a circulation. It has minimum cost because f_i is a minimum-cost flow (that could have been computed by the Successive Shortest Path Algorithm). $\square$

Notation: For $i \in \{1, \dots, k\}$ and $v \in V(G)$, we define $d_i(v) = \mathrm{dist}_{G^i_{g_i,u},\tau}(s, v)$.

Lemma 5.2 Let $v \in V(G)$. For $i \in \{1 \dots, k\}$, we have

$$d_i(v) \geq d_{i+1}(v).$$

Proof: Neither adding an edge entering s nor deleting an edge leaving s can decrease the distance from s to v. We may assume that f_i is computed by augmenting iteratively flow on shortest paths s-s_i-paths (if s_i is a source) or along cycles containing the edges (s_i, s) (if s_i is a sink). As in the proof of Lemma 4.2, none of these augmentations can reduce the length of a shortest s-v-paths. $\square$

Lemma 5.3 For any $i \in \{2, \dots, k\}$, we have:

- If $P \in \Delta_{i-1}$ and $v \in V(G)$ with $v \in v(P)$, then $\tau(P_{s,v}) \geq d_i(v)$.
- If $P \in \Delta_i$ and $v \in V(G)$ with $v \in v(P)$, then $\tau(P_{s,v}) \leq d_i(v)$.

Proof: Follows directly from the previous lemma. $\square$

Before we examine the temporally repeated flow $[\Gamma_1]^T$, we will consider the temporally repeated flow $[\Gamma_1]^\infty$ which is defined analogously to $[\Gamma_1]^T$ with the difference that we never stop sending flow along the paths in Γ_1.

Theorem 5.4 $[\Gamma_1]^\infty$ is a feasible flow over time.

Proof: We will show by reverse induction in i that all $[\Gamma_i]^\infty$ meet the capacity constraints. For $i = k$ this is true because Γ_k is a (standard) flow decomposition of a feasible static flow.

We have to show that Γ_{i-1} remains feasible if Γ_i is feasible. Let $(v, w) \in E(G)$. If $f_{i-1}(v, w) = 0$, then nothing changes in this iteration for this edges, so in particular the capacity constraint is still met. Assume that $f_{i-1}(v, w) > 0$. For times $\theta \leq d_i(v)$ we have $[\Gamma_i]^\infty_{(v,w)}(\theta) = [\Gamma_{i-1}]^\infty_{(v,w)}(\theta)$ because for all paths $P \in \Delta_{i-1}$ we have $\tau(P_{s,v}) \geq d_i(v)$. If $\theta \geq d_{i-1}(v)$, then $[\Delta_{i-1}]^\infty_{(v,w)}(\theta) = f_{i-1}(v, w)$, so $[\Gamma_{i-1}]^\infty_{(v,w)}(\theta) = g_{i-1}(v, w)$. So let $d_i(v) < \theta < d_{i-1}(v)$. The first of these two inequalities leads to $[\Gamma_i]^\infty_{(v,w)}(\theta) = g_i(v, w)$. Since $f_{i-1}(v, w) > 0$, we have

$$g_i(v, w) = [\Gamma_i]^\infty_{(v,w)}(\theta) \leq [\Gamma_{i-1}]^\infty_{(v,w)}(\theta) \leq [\Gamma_i]^\infty_{(v,w)}(\theta) + f_{i-1}(v, w) = g_{i-1}(v, w).$$

This shows that the capacity constraint on this edge is still not violated.

The case $f_{i-1}(v, w) < 0$ can be handled analogously.

To see that the constraints on the node balances are met, we only have to check that these are not violated for sources. But for sources this follows from the assumption that they have no incoming edges. $\square$

Lemma 5.5 The total amount of flow that is sent by $[\Gamma_1]^\infty$ from $\{s_1, \dots, s_i\}$ to $\{s_{i+1}, \dots, s_k\}$ to is $-\sum\limits_{e \in E(G^{i+1})} \tau(e) g_{i+1}(e)$

Proof: At the end of the algorithm, all edges leaving s have been removed, so g_1 is a circulation in G. All paths from $\{s_1, \dots, s_i\}$ to $\{s_{i+1}, \dots, s_k\}$ in a flow decomposition of g_1 are contained in a flow decomposition of g_{i+1}. Thus, the total amount of dynamic flow sent from $\{s_{i+1}, \dots, s_k\}$ to $\{s_1, \dots, s_i\}$ equals the total cost (in terms of τ) of g_{i+1} which is $\sum\limits_{e \in E(G^{i+1})} \tau(e) g_{i+1}(e)$. Since we do not store flow in interior node, $-\sum\limits_{e \in E(G^{i+1})} \tau(e) g_{i+1}(e)$ is the value of the dynamic flow sent from $\{s_1, \dots, s_i\}$ to $\{s_{i+1}, \dots, s_k\}$. This shows that the amount of (dynamic) flow sent from $\{s_1, \dots, s_i\}$ to $\{s_{i+1}, \dots, s_k\}$ is maximum. $\square$

The proof of optimality is based on the analysis of cuts in the infinite time-expanded network G^∞ that is defined as the pair $(V(G^\infty), E(G^\infty))$ with:

- $V(G^\infty) := V(G) \times \mathbb{N}$,
- $E(G^\infty) := \{((v, \theta), (w, \zeta)) \in V(G^\infty) \times V(G^\infty) \mid (v = w \text{ and } \zeta = \theta + 1) \text{ or } ((v, w) \in E(G) \text{ and } \tau(v, w) = \zeta - \theta)\}$, and
- $u'((v, \theta), (v, \theta + 1)) := \infty$, and $u'((v, \theta), (w, \zeta)) := u(v, w)$.

Since $V(G^\infty)$ is infinite, G^∞ is not a graph, but many concepts that work for (finite) graphs can be generalized to such inifinite networks in a straightforward manner. We will fo this for cuts in the following where we restrict ourselves to cuts of finite size induced by a finite set of nodes.

Theorem 5.6 $[\Gamma_1]^\infty$ is a lexicographically maximum flow over time.

Proof: For $i \in \{1, \dots, k\}$ let

$$\begin{aligned} X_i \quad := \quad & \Big\{(v, \theta) \in V(G^\infty) \mid d_{i+1}(v) \le \theta\Big\} \cup \\ & \Big\{(v, T) \in V(G^\infty) \mid v \in S_- \cap \{s_1, \dots, s_i\}\Big\}. \end{aligned}$$

Claim 1: $\Big\{(v, 0) \in V(G^\infty) \mid v \in S_+ \cap \{s_1, \dots, s_i\}\Big\} \subset X_i$.

Proof of Claim 1: For $v \in S_+ \cap \{s_1, \dots, s_i\}$ there is still an edge (s, v) in $G^i_{g_{i+1}, u}$ which has transit time 0, so we have $d_{i+1}(s, v) = 0$ and therefore $(v, 0) \in X_i$. $\square$ Claim 1

Claim 2: $\Big\{(v,\theta) \in V(G^\infty) \mid v \in S_+ \setminus \{s_1,\dots,s_i\}\Big\} \cap X_i = \emptyset$ for all $\theta \in \{0,\dots,T\}$.

Proof of Claim 2: For $v \in S_+ \setminus \{s_1,\dots,s_i\}$ we removed the edge (s,v) from the graph and augmented the flow by a maximum s-v-flow. This means that after the augmentation, v is not reachable any more from s in the residual graph, so $d_{i+1}(v) = \infty$. $\square$ Claim 2

Claim 3: $\Big\{(v,\theta) \in V(G^\infty) \mid v \in S_- \setminus \{s_1,\dots,s_i\}\Big\} \cap X_i = \emptyset$ for all $\theta \in \{0,\dots,T\}$.

Proof of Claim 3: If $v \in S_- \setminus \{s_1,\dots,s_i\}$ we have already added the edge (v,s) with transit time $-T-1$ in $G_{g_{i+1},u}$. So if $d_{i+1}(v) \le T$, we had a negative cycle in the residual graph which is a contradiction to the optimality of g_{i+1}. $\square$ Claim 3

The definition of X_i and the three claims above show that $X_i \times (V(G^\infty) \setminus X_i)$ is a cut that separates $A_i := \Big\{(v,0) \in V(G^\infty) \mid v \in S_+ \cap \{s_1,\dots,s_i\}\Big\} \cup \Big\{(v,T) \in V(G^\infty) \mid v \in S_- \cap \{s_1,\dots,s_i\}\Big\}$ from $B_i := \Big\{(v,0) \in V(G^\infty) \mid v \in S_+ \setminus \{s_1,\dots,s_i\}\Big\} \cup \Big\{(v,T) \in V(G^\infty) \mid v \in S_- \setminus \{s_1,\dots,s_i\}\Big\}$. We will show that the residual capacity (with respect to $[\Gamma_1]^\infty$) on the cut edges is zero. This means that the flow from A_i to B_i is maximum and since this is true for $i \in \{1,\dots,k\}$, $[\Gamma_1]^\infty$ is lexicographically maximum.

Claim 4: $u(X_i \times (V(G^\infty) \setminus X_i) = -\sum_{(v,w) \in E(G^{i+1})} \tau(v,w) g_{i+1}(v,w)$

Proof of Claim 4: Let $e = ((v,\theta),(w,\zeta)) \in E(G^\infty) \cap (X_i \times (V(G^\infty) \setminus X_i))$.

Since no sink has an outgoing edge in G and $(v,\theta) \in X_i$, we have $\theta \ge d_{i+1}(v)$. By the definition of X_i we also have $(v,\theta+1) \in X_i$, so e cannot be a hold edge.

Therefore, $\zeta = \theta + \tau(v,w)$ and $d_{i+1}(v) \le \theta < d_{i+1}(w)$. The capacity of the edge is $u(v,w)$. If we sum up the capacities of all these edges $((v,\theta),(w,\zeta))$ we get

$$\begin{aligned} u(X_i \times (V(G^\infty) \setminus X_i)) &= \sum_{e=(v,w) \in E(G)} \max\big\{0, d_{i+1}(w) - d_{i+1}(v) - \tau(e)\big\} \cdot u(e) \\ &= \sum_{e=(v,w) \in E(G)} (d_{i+1}(w) - d_{i+1}(v) - \tau(e)) \cdot g_{i+1}(e) \end{aligned}$$

where the correctness of last equality follows can be seen as follows: If $g_{i+1}(v,w) < u(v,w)$, then $(v,w) \in E(G^{i+1}_{g_{i+1},u})$ and therefore $d_{i+1}(w) \le d_{i+1}(v) + \tau(v,w)$, so in the equation we can replace $u(e)$ be $g_{i+1}(e)$. Moreover, if $g_{i+1}(v,w) > 0$, then $(w,v) \in E(G^{i+1}_{g_{i+1},u})$ and therefore

$d_{i+1}(v) \le d_{i+1}(w) - \tau(v,w)$. Hence we have

$$\begin{aligned}
&u((X_i \times (V(G^\infty) \setminus X_i)) \\
&= \sum_{e=(v,w)\in E(G)} (d_{i+1}(w) - d_{i+1}(v))g_{i+1}(e) - \sum_{e=(v,w)\in E(G)} \tau(e)\cdot g_{i+1}(e) \\
&= \sum_{v\in V(G)} d_{i+1}(v)\left(\sum_{w:(w,v)\in E(G)} g_{i+1}(w,v) - \sum_{w:(v,w)\in E(G)} g_{i+1}(v,w)\right) - \sum_{e=(v,w)\in E(G)} \tau(e)\cdot g_{i+1}(e) \\
&= \sum_{v\in\{s_1,\dots,s_k\}} d_{i+1}(v)\left(\sum_{w:(w,v)\in E(G)} g_{i+1}(w,v) - \sum_{w:(v,w)\in E(G)} g_{i+1}(v,w)\right) - \sum_{e=(v,w)\in E(G)} \tau(e)\cdot g_{i+1}(e). \\
&= \sum_{v\in S_+} d_{i+1}(v)\left(-\sum_{w:(v,w)\in E(G)} g_{i+1}(v,w)\right) + \sum_{v\in S_-} d_{i+1}(v)\left(\sum_{w:(w,v)\in E(G)} g_{i+1}(w,v)\right) \\
&\quad - \sum_{e=(v,w)\in E(G)} \tau(e)\cdot g_{i+1}(e) \\
&= -\sum_{e=(v,w)\in E(G^{i+1})} \tau(e)g_{i+1}(e)
\end{aligned}$$

For the last equation note that if there is an edge $(v,w) \in E(G)$ with $v \in S_+$ and $g_{i+1}(v,w) > 0$, then the edge $(s,v) \in E(G^{i+1})$ and $d_{i+1}(v) = 0$. Moreover, if $v \in S_-$ and there is an edge $(w,v) \in E(G)$ with $g_{i+1}(w,v) > 0$, then $(v,s) \in E(G^{i+1})$ and $g_{i+1}(v,s) = \sum_{w:(w,v)\in E(G)} g_{i+1}(w,v)$ and $d_{i+1}(v) = T+1 = -\tau(v,s)$. □ Claim 4

□

Theorem 5.7 $[\Gamma_1]^\infty$ has time horizon T.

Proof: The flow g_1 is a minimum-cost circulation in G and there are no cycles of transit time 0, g_1 is the zero flow. Therefore, so for any edge after a certain time the dynamic flow must be 0. Since the sinks have no incoming flow after time T (due to the edges to s with transit time $-T-1$), any flow on an edge after time T could never reach a sink, so there can be no flow on any edge after time T. □

Corollary 5.8 A lexicographically maximum flow over time with k sources and sinks in a dynamic flow network (G,u,τ) can be computed in time $O(km\log m(m + n\log n))$ where $n = |V(G)|$ and $m = |E(G)|$.

Proof: The running time of the Lexicographically Maximum Dynamic Flow Algorithm is dominated by the k computations of minimum-cost flows. Each of these minimum-cost flow can be computed in time $O(m\log m(m+n\log n))$ (Orlin [1993]). □

5.3 Finding the optimum transshipment time horizon

We will first determine the smallest time horizon T for which we can find a transshipment for a given dynamic flow network (G, u, τ) and values $b : V(G) \to \mathbb{R}$. To this end we will apply results from static flow to the time-expanded network. In a second step we will then compute a b-flow with time horizon T.

Notation: Let $S := \{v \in V(G) \mid b(v) \neq 0\}$ and $k := |S|$. Let $S_+ = \{v \in V(G) \mid b(v) > 0\}$ be the set of sources and $S_- = \{v \in V(G) \mid b(v) < 0\}$ be the set of sinks. For $A \subseteq S$ let $b(A) = \sum_{v \in A} b(v)$, and let $r(A)$ be the maximum amount of flow that can be sent from $A \cap S_+$ to $(S \setminus A) \cap S_-$ in time T (ignoring the supply an demand values).

Lemma 5.9 *There is a b-flow over time with time horizon T in (G, u, τ) if and only if $r(A) \geq b(A)$ for all $A \subseteq S$.*

Proof: Consider the time-expanded network G'. Translate the b-values in G to b-values in G' by setting $b(v, 1) = b(v)$ if $b(v) > 0$, $b(v, T) = b(v)$ if $b(v) < 0$ and $b(v, \theta) = 0$ for all remaining nodes. Then, a dynamic (discrete) b-flow with time horizon T in (G, u, τ) exists if and only if there is a static b-flow in G'. And such a static b-flow in G' exists if and only if $\sum_{e \in \delta^+(U)} u(e) \geq \sum_{v \in U} b(v)$ for all sets $U \subset V(G')$ which is equivalent to the condition that $r(A) \geq b(A)$ for all $A \subseteq S$. $\square$

Remark: For the proof, we make use of the fact that we consider the discrete model. However, the statement is also true for the continuous model if we have integral transit times and an integral time horizon T. In Lemma 7.3 (with $\Delta = 1$) we will see how continuous dynamic b-flow can be translated into static b-flows in the time-expanded network and vice-versa.

Theorem 5.10 *It can be checked in time $O(2^k m \log m(m + n \log n))$ if a b-flow over time with time horizon T exists in (G, u, τ).*

Proof: For each set $A \subseteq S$ we can check if $r(A) \geq b(A)$ by computing a maximum s-t-flow over time where s is an additional super source with edges to the elements of A and t is a super sink with edges from the elements of $S \setminus T$. Each such computation can be performed in time $O(m \log m(m + n \log n))$ by the FORD-FULKERSON ALGORITHM, and, of course, there are 2^k sets $A \subseteq S$. $\square$

Lemma 5.11 *$r : \text{Pow}(S) \to \mathbb{R}$ is submodular, i.e. $r(A \cup B) \leq r(A) + r(B) - r(A \cap B)$ for all $A, B \subseteq S$.*

Proof: For $Z \subseteq S$, define $A'_+ = \{(v, 1) \mid b(v) > 0, v \in Z\}$ and $\bar{A}'_- = \{(v, T) \mid b(v) > 0, v \in S \setminus Z\}$. Since $r(A)$ is the minimum value of a cut in the time-expanded network G' separating A'_+ from $\bar{A}'_-$, there must be a set $X_A \subset V(G)$ with $A'_+ \subseteq X_A$, $X_A \cap \bar{A}'_- = \emptyset$ and $u(\delta^+_{G'}(X_A)) = r(A)$. Let X_B, $X_{A \cup B}$ and $X_{A \cap B}$ be corresponding sets for B, $A \cup B$ and $A \cap B$, respectively. Then,

$r(A \cup B) = u(\delta^+_{G'}(X_{A\cup B})) \leq u(\delta^+_{G'}(X_A \cup X_B)) \leq u(\delta^+_{G'}(X_A)) + u(\delta^+_{G'}(X_B)) - u(\delta^+_{G'}(X_A \cap X_B)) = r(A) + r(B) - r(A \cap B)$ where we make use of the fact that $\delta^+_{G'}(X_{A\cup B})$ has minimum capacity and that the function $u(\delta^+_{G'}(\cdot))$ is submodular. □

Theorem 5.12 There is a strongly polynomial-time algorithm that checks if a b-flow over time with time horizon T exists in (G, u, τ).

Proof: We have to decide if there is a set $A \subseteq S$ with $r(A) - b(A) < 0$. Since $A \mapsto r(A) - b(a)$ is also a submodular function (as b is modular), we may apply the strongly polynomial-time algorithm for minimizing submodular functions that is described in (Grötschel, Lovász, and Schrijver [1988]). Obviously, the instance is infeasible if and only if the algorithm returns a set A with $r(A) - b(A) < 0$. □

To find an optimum time horizon T in polynomial time we may apply binary search. As a starting point we need an an upper bound for T. It is easy to see that e.g. $n \max\{\tau(e) \mid e \in E(G)\} + \sum_{v \in S^+} b(v)$ is such an upper bound.

We can also achieve a strongly polynomial running by applying the parametric search proposed by Megiddo [1979] that we have already used to solve the Quickest Flow over Time Problem (see the proof of Theorem 3.2).

5.4 Computing a b-flow with given time horizon

In this section we assume that we are given an instance (G, u, τ), b of the *quickest transshipment problem* and a time horizon T such that a b-flow over time with time horizon T exists. The task is to compute such a b-flow.

Remark: If there is only a fixed number of sources and sinks, then we can compute a feasible b-flow in the following way: We first compute lexicographically maximum flows for all orderings of the source and sink set. Then, if there is a feasible solution, the b-values must be a convex combination of the supply and demand vectors of the lexicographically maximum flows. A feasible b-flow is then be given by the corresponding convex combination of the lexicigraphically maximum flows.

Definition 5.3 A set $A \subseteq S$ is called *tight* if $b(A) = r(A)$.

Note that $b(\emptyset) = r(\emptyset) = b(S) = r(S) = 0$, so both $\emptyset$ and S are tight. We call them trivially tight.

Lemma 5.13 If there is a b-flow over time with time horizon T in (G, u, τ) and $A, B \subseteq S$ are tight sets, then both $A \cup B$ and $A \cap B$ are tight.

Proof: Let $A, B \subseteq S$ be tight. Then

$$b(A \cup B) = b(A) - b(A \cap B) + b(B) \geq r(A) - r(A \cap B) + r(B) \geq r(A \cup B),$$

so $b(A \cup B) = r(A \cup B)$ and $A \cup B$ is tight. Moreover, the two inequalities above must in fact be equalities (because otherwise we had $b(A \cup B) > r(A \cup B)$, so there would not be a b-flow), hence we have $b(A \cap B) = r(A \cap B)$ and $A \cap B$ is tight.

$\square$

Observation: If there is an ordering $s_1, \ldots, s_k$ of the sources and the sinks such that $\{s_1, \ldots s_i\}$ is tight for all $i \in \{1, \ldots, k\}$, then we can compute a feasible b-flow by computing a lexicographically maximum flow.

Generally, there will not be such an ordering but we will modify the instance equivalently such that there is one in the modified instance.

First we modify (G, u, τ) and b in the following way:

- For each source s
  - add a node s^0 to $V(G)$,
  - add an edge (s^0, s) with $\tau'(s^0, s) = 0$ and $u'(s^0, s) = \infty$,
  - set $b'(s^0) = b(s)$ and $b'(s) = 0$.
- For each sink s
  - add a node s^0 to $V(G)$,
  - add an edge (s, s^0) with $\tau'(s, s^0) = 0$ and $u'(s, s^0) = \infty$,
  - set $b'(s^0) = b(s)$ and $b'(s) = 0$.

Let H be the resulting graph and S' be the resulting source and sink set. For all remaining nodes and edges, b', u', and τ' are identical to b, u, and τ, respectively.

We will add more copies $s^1, s^2, \ldots$ of the sources and sink s until we get an instance to which we can apply the LEXICOGRAPHICALLY MAXIMUM DYNAMIC FLOW ALGORITHM. The copies s^0 that we have added so far are called *first terminals*.

We will always keep track of a chain $\mathcal{C}$ of tight sets in $\text{Pow}(S')$ (ordered by the inclusion relation). At the beginning we set $\mathcal{C} = \{\emptyset, S'\}$. We call two elements $P, Q \in \mathcal{C}$ *adjacent*, if there is no set $R \in \mathcal{C} \setminus \{P, Q\}$ with $P \subset R \subset Q$ or $Q \subset R \subset P$.

Goal: Increase $|\mathcal{C}| - |S'|$ until it equals 1.

Invariant (*): During the modifications, the chain has the following property: If $P, Q \in \mathcal{C}$ are two adjacent sets with $|P \setminus Q| > 1$, then $P \setminus Q$ consists of first terminals only.

INCREASING CHAIN ALGORITHM

Input: A dynamic flow network (G, u, τ), $b : V(G) \to \mathbb{R}$, a time horizon $T \geq 0$ for which a b-flow over time with time horizon T exists.

Output: An equivalent network (H, u', τ') with terminal set S' and a chain $\mathcal{C} \subseteq \mathrm{Pow}(S')$ of tight subsets with $|\mathcal{C}| - |S'| = 1$.

```
①        Compute $(H, u', \tau')$, $b'$, and $S'$ as above. Set $\mathcal{C} = \{\emptyset, S'\}$.
②        WHILE $(|\mathcal{C}| - |S'| < 1)$
         {
             Choose adjacent sets $P, Q \in \mathcal{C}$ with $Q \subset P$ such that $|P| > |Q| + 1$.
             Choose $s \in S$ such that $s^0 \in P \setminus Q$.
             IF $(Q \cup \{s^0\}$ is tight$)$
             {
                 Add $Q \cup s^0$ to $\mathcal{C}$.
                 CONTINUE.
             }
             Let $s^0, s^1, \ldots, s^{i-1}$ be the copies of $s$.
             IF $(s$ is a source$)$
             {
                 Choose $\alpha$ maximal such that the following modifications lead to a
                 feasible instance:
                 Add a copy $s^i$ of $s$ and an edge $(s^i, s)$ with $\tau'(s^i, s) = 0$ and $u'(s^i, s) =$
                 $\alpha$ to $H$.
                 Set $b'(s^i) = r_\alpha(Q \cup \{s^i\}) - r_\alpha(Q)$ and $b'(s^0) = b'(s^0) - b'(s^i)$.
                 Choose $\delta$ minimal such that the following modifications lead to a
                 feasible instance:
                 Add a copy $s^{i+1}$ of $s$ and an edge $(s^{i+1}, s)$ with $\tau'(s^{i+1}, s) = \delta$ and
                 $u'(s^{i+1}, s) = 1$ to $H$.
                 Set $b'(s^{i+1}) = r_\delta(Q \cup \{s^i, s^{i+1}\}) - r_\delta(Q \cup \{s^i\})$ and $b'(s^0) = b'(s^0) -$
                 $b'(s^i)$.
             }
             ELSE
             {
                 Modify the instance analogously but with edges from $s$ to $s^i$ and to
                 $s^{i+1}$.
             }
             Add $Q \cup \{s^i\}$ and $Q \cup \{s^i, s^{i+1}\}$ to $\mathcal{C}$.
             For each $A \in \mathcal{C}$ with $Q \subsetneq A$ replace $A$ by $A \cup \{s^i, s^{i+1}\}$.
             Find a tight set $R$ with $Q \cup \{s^i, s^{i+1}\} \subsetneq R \subsetneq P \cup \{s^i, s^{i+1}\}$ and add it to $\mathcal{C}$.
         }
```

We only describe and analyze the modification of a source in detail but based on this the handling of sinks is not difficult.

Lemma 5.14 In each iteration, we have $0 \leq \alpha < n \max\{u(e) \mid e \in E(G)\}$ and $0 < \delta \leq T+1$ and hence can compute α and δ by binary search.

Proof: If $\alpha = 0$, the edge (s^i, s) has capacity 0 and the b'-values are not changed. Therefore the instance remains feasible. On the other hand if $\alpha \geq n \max\{u(e) \mid e \in E(G)\}$, then this is equivalent to setting the edges capacity of (s^i, s) to ∞ and then s^i plays the same role as s_0. Then before resetting the b'-value of s^0 we had: $b'(s^0) - r_\alpha(Q \cup \{s^i\}) + r(Q) = b'(s^0) - r_\alpha(Q \cup \{s^0\}) + b'(Q) < b'(s^0) - b'(Q \cup \{s^0\}) + b'(Q) = 0$ because $Q \cup \{s^0\}$ is not tight. This means that after resetting the b'-value of s^0 it would become negative, so we would not have a feasible instace (because s^0 has no incoming edge).

If $\delta = T+1$, then this does not really change the instance, to it remains feasible. If $\delta = 0$ this has the same effect as increasing the capacity of the edge (s^i, s) by 1, so it would lead to an infeasible instance (because α is maximum). $\square$

Lemma 5.15 At the end of each iteration of the algorithm, the following statements hold:

(a) $Q \cup \{s^i\}$ and $Q \cup \{s^i, s^{i+1}\}$ are tight.

(b) If $A \in \mathcal{C}$ and $A \subseteq Q$, then A remains tight.

(c) If $A \in \mathcal{C}$ and $Q \subsetneq A$, then $A \cup \{s^i, s^{i+1}\}$ is tight.

Proof:

(a) We have

$$b'(Q \cup \{s^i\}) = b'(Q) + b'(s^i) = b'(Q) + r_\alpha(Q \cup \{s^i\}) - r_\alpha(Q) = r_\alpha(Q \cup \{s^i\}),$$

because Q is tight, and

$$\begin{aligned} b'(Q \cup \{s^i, s^{i+1}\}) &= b'(Q \cup \{s^i\}) + b'(s^{i+1}) \\ &= r_\delta(Q \cup \{s^i, s^{i+1}\}) - r_\delta(Q \cup \{s^i\}) + b'(Q \cup \{s^i\}) \\ &= r_\delta(Q \cup \{s^i, s^{i+1}\}), \end{aligned}$$

because $Q \cup \{s^i\}$ is tight.

(b) The statement is trivial because $b'(A)$ and $r'(A)$ will not be changed for such sets.

(c) Let $A \in \mathcal{C}$ be a set with $Q \subsetneq A$. Due to submodularity of r, applied to A and $Q \cup \{s^i, s^{i+1}\}$ we have

$$r(A \cup \{s^i, s^{i+1}\}) = r(A \cup (Q \cup \{s^i, s^{i+1}\})) \leq r(A) + r(Q \cup \{s^i, s^{i+1}\}) - r(Q),$$

so

$$\begin{aligned} b'(A \cup \{s^i, s^{i+1}\} &= b'(A) + b'(s^i) + b'(s^{i+1}) = r(A) + r(Q \cup \{s^i, s^{i+1}\}) - r(Q) \\ &\geq r(A \cup \{s^i, s^{i+1}\}). \end{aligned}$$

□

Theorem 5.16 At the end of an iteration of the Increasing Chain Algorithm there is a tight set R with $Q \cup \{s^i, s^{i+1}\} \subsetneq R \subsetneq P \cup \{s^i, s^{i+1}\}$, and such a set can be computed in polynomial time.

Proof: Assume that we had used $\delta - 1$ instead of δ in the last modification. Due to the definition of δ, this leads to an infeasible instance. Let $\tilde{b}$ and $\tilde{r}$ be the functions that we had in this case instead of b' and r at the end of the iteration (and b' and r_δ are the values with the correct value δ after inserting s^{i+1}).

Let R' be any set with $\tilde{b}(R') > \tilde{r}(R')$.

Claim 1: R' is tight w.r.t. b' and r' and $s^{i+1} \in R'$ but $s^0 \notin R'$.

Proof of Claim 1: Because of $\tilde{r}(R') \geq r_\delta(R')$ we have $b'(R') < \tilde{b}(R') \leq b'(R') + 1$, so $s^{i+1} \in R'$ but $s^0 \notin R'$. Moreover,

$$\tilde{b}(R') > \tilde{r}(R') \geq r_\delta(R') \geq b'(R') \geq \tilde{b}(R') - 1.$$

So, in fact, we have equality in all these inequalities which means that R' is tight w.r.t. b' and r'. □ Claim 1

Claim 2: $(R' \cap P) \subsetneq P$

Proof of Claim 2: Follows from the fact that $s^0 \in P \setminus R'$ □ Claim 2

Claim 3: $(R' \cap P) \not\subset Q \cup \{s^i, s^{i+1}\}$.

Proof of Claim 3: We first show that for any set $A \subseteq Q \cup \{s^i, s^{i+1}\}$: $\tilde{r}(A) \geq \tilde{b}(A)$. This follows from Claim 1 if $s^{i+1} \notin A$, so assume that $s^{i+1} \in A$. With $Q \cup \{s^i, s^{i+1}\} = A \cup (Q \cup \{s^i\})$ we have

$$\begin{aligned} \tilde{r}(A) &\geq \tilde{r}(Q \cup \{s^i, s^{i+1}\}) + \tilde{r}(A \cap (Q \cup \{s^i\})) - \tilde{r}(Q \cup \{s^i\}) \\ &\geq \tilde{b}(Q \cup \{s^i, s^{i+1}\}) + \tilde{b}(A \cap (Q \cup \{s^i\})) - \tilde{b}(Q \cup \{s^i\}) \\ &= \tilde{b}(A) \end{aligned}$$

where the last inequality follows form the fact that $Q \cup \{s^i\}$ and $Q \cup \{s^i, s^{i+1}\}$ are tight and $s^{i+1} \notin (A \cap (Q \cup \{s^i\}))$.

Now assume that $(R' \cap P) \subset Q \cup \{s^i, s^{i+1}\}$. Then, $R' \cap (P \cup \{s^i, s^{i+1}\}) \subset Q \cup \{s^i, s^{i+1}\}$ and

$$\begin{aligned} \tilde{r}(R') &\geq \tilde{r}(R' \cup P \cup \{s^i, s^{i+1}\}) + \tilde{r}(R' \cap (P \cup \{s^i, s^{i+1}\})) - \tilde{r}(P \cup \{s^i, s^{i+1}\}) \\ &\geq \tilde{b}(R' \cup P \cup \{s^i, s^{i+1}\}) + \tilde{b}(R' \cap (P \cup \{s^i, s^{i+1}\})) - \tilde{b}(P \cup \{s^i, s^{i+1}\}) \\ &= \tilde{b}(R') \end{aligned}$$

where the last inequality follows from the fact that $s^0 \in P \subset (R' \cup P \cup \{s^i, s^{i+1}\})$. This is a contradiction to the choice of R'. □ Claim 3

Let $R := (R' \cap P) \cup Q \cup \{s^i, s^{i+1}\}$. Then, the two last claims show that $Q \cup \{s^i, s^{i+1}\} \subsetneq R \subsetneq (P \cup \{s^i, s^{i+1}\})$. We have $R = (Q \cup \{s^i, s^{i+1}\} \cup R') \cap (P \cup \{s^i, s^{i+1}\})$, so R is tight.

Computing such a set is as difficult as checking the feasibility of the instance with $\delta - 1$, so it can be done in polynomial time. □

Theorem 5.17 Given a dynamic flow network (G, u, τ), $b : V(G) \to \mathbb{R}$ and a time horizon T such that a b-flow over time with time horizon T exists in (G, u, τ), a feasible b-flow over time with time horizon T can be computed in strongly polynomial time.

Proof: We run the Increasing Chain Algorithm and then the Lexicographically Maximum Dynamic Flow Algorithm. The most time-consuming part of each iteration of the Increasing Chain Algorithm is the computation of α and δ. Both values can be computed in polynomial time by binary search. To get a strongly polynomial running time, we may apply again parametric search (Megiddo [1979]). □

6 Earliest arrival transshipment

In the *earliest arrival transshipment problem*, we ask for a b-flow over time in a dynamic flow network with the smallest possible time horizon such that at any time the amount of flow that has reached a sink is maximized.

The following example which is due to Fleischer [2001] proves that, in general, there will not be a a feasible soltion even if a b-flow over time in the given network exists: The network shown in Figure 7 is an example. The figure shows supply- and demand-values. Assume that we have edge capacities and transit times of 1s. For a time horizon $T = 2$, it is possible to send 4 units of flow to the sink by sending one unit of flow along each of the for edges (and this is the only way to send at least four units of flow in time $T = 2$ to the sinks). However, if one does so it take two more time unit to ship the remaining two units of flow becaus they have to be sent along the edge (v_2, w_1). On the other hand, it would be possible to send the whole amount of flow within a time horizon of $T = 3$ along the three edges (v_1, w_1), (v_2, w_1), and (v_2, w_2) (start to send flow at time 0 and stop sending it at time 2). This means that there is no solution that maximizes the total flow that has reached the sinks for all times simultaneously.

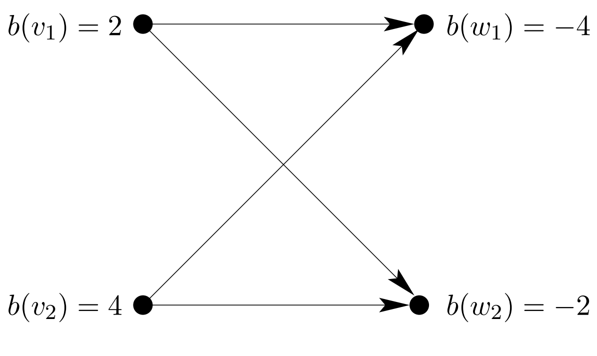

Figure 7: A flow network with a static s-t-flow and the two paths of a generalized path decomposition.

For a characterization of networks in which (no matter what transit times and the edge capacities are used), see Schmidt and Skutella [2010].

If there is only one sink, then an earliest arrival transshipment always exists. In the discrete model, this follows from the same construction as in the proof of the existence of an earliest arrival flow.

For the special case where there is only one sink and all transit times are 0, Hajek and Ogier [1984] and Fleischer [2001] describe polynomial-time algorithms to compute earliest arrival transshipments. For general transit times (but only one sink) Baumann and Skutella [2009] propose an algorithm whose running time is polynomial in the sum of the input and output size.

7 Minimum-cost flows over time

Definition 7.1 Let (G, u, τ) be a dynamic flow network with edge costs $c : E(G) \to \mathbb{R}$. Let f be a (discrete or continuous) b-flow over time with time horizon T in (G, u, τ) (for a mapping $b : V(G) \to \mathbb{R}$ with $\sum_{v \in V(G)} b(v) = 0$). Then, the cost of f is defined as

$$c(f) := \sum_{\theta=0}^{T} \sum_{e \in E(G)} c(e) f_e(\theta)$$

if f is a discrete flow, and

$$c(f) := \int_0^T \sum_{e \in E(G)} c(e) f_e(\theta) d\theta$$

if f is a continuous flow.

Remark: Now that we take edge costs into account, it can be reasonable to define costs for storing flow in interior nodes. In the discrete case, this can be modelled by adding loops (i.e. edges with the same start and end node) with specified costs and transit times 1. For continuous flows, loops cannot model the storage of flow exactly because they have to have a fixed length, so it would not be possible to store flow for an arbitrarily small time interval.

In this section, our goal is to compute b-flows over time f that either respect a given time horizon and minimize the cost $c(f)$ (Minimum Cost Transshipment Problem) or respect an upper bound C on the total cost and minimize the arrival time (Quickest Cost-Bound Transshipment Problem). We assume (w.l.o.g.) that no source (i.e. no vertex v with $b(v) > 0$) has incoming edges and no sink (i.e. no vertex v with $b(v) < 0$) has outgoing edges.

7.1 NP-hardness

The special case of the Minimum Cost Transshipment Problem in which we have only one source s and one sink t is called Minimum Cost s-t-Flow over Time Problem. Even this special case is NP-hard:

Theorem 7.1 (Klinz and Woeginger [2004]) The Minimum Cost s-t-Flow over Time Problem is NP-hard.

Proof: Here we only consider the continuous problem. The proof is a reduction from the Partition Problem. In the Partition Problem we are given positive integral numbers $a_1, \dots, a_n$ with $\sum_{i=1}^n a_i = 2B$ for an integral number B. The task is to decide if there is a set $I \subset \{1, \dots, n\}$ with $\sum_{i \in I} a_i = B$. The Partition Problem is an NP-complete decision problem (see Garey and Johnson [1979], SP12).

Given an instance $a_1, \dots, a_n$ (with $n > 2$) of the Partition Problem we construct a graph G with vertex set $V(G) = \{v_1, \dots, v_{n+1}\} \dot\cup \{w_1 \dots, w_n\} \dot\cup \{s, t\}$ and edge set $E(G) = \{(s, v_1)\} \cup \{(v_i, v_{i+1}) \mid i = 1, \dots, n\} \cup \{(v_i, w_i) \mid i = 1, \dots, n\} \cup \{(w_i, v_{i+1}) \mid i = 1, \dots, n\} \cup \{(v_{n+1}, t)\}$. The edges (s, v_1) and (v_{n+1}, t) have capacity 2, while all other edges have capacity 1. We define $\tau(v_i, w_i) = a_i$ for $i \in \{1, \dots, n\}$, all other edges have transit time 0. Moreover, we set $c(v_i, v_{i+1}) = a_i$ (for $i \in \{1, \dots, n\}$), and set $c(e) = 0$ for all other edges. Finally, let $T := B + 1$ and $b(s) := -b(t) := 2$ (and $b(v) := 0$ for all $v \in V(G) \setminus \{s, t\}$.

Note that for any s-t-path P in this instance we have $\tau(P) + c(P) = 2B$

Now assume that the instance of the Partition Problem is a YES-instance, so there is $I \subset \{1, \dots, n\}$ with $\sum_{i \in I} a_i = B$. Then consider two s-t paths P and Q in G where $E(P) = \{(s, v_1)\} \cup \{(v_i, v_{i+1}) \mid i \in I\} \cup \{(v_i, w_i) \mid i \in \{1, \dots, n\} \setminus I\} \cup \{(w_i, v_{i+1}) \mid i \in \{1, \dots, n\} \setminus I\} \cup \{(v_{n+1}, t)\}$ and $E(Q) = (E(G) \setminus E(P)) \cup \{(s, v_1)\} \cup \{(v_{n+1}, t)\}$. Then both paths have transit time and cost B. If we send one unit of flow along each path (starting to send flow at time 0 and stopping to send it at time 1), then 2 units of flow will reach t in time $B + 1$, and the total cost of this flow is $2B$.

Now, let us assume that the instance is a NO-instance. Then, there is no s-t-path with transit time B, so any flow that could reach t in time would be along a path of transit time $B - M$ where M is a positive integer. However, the cost for sending one unit of flow along such a path would be at least $B + M$, so in this case the cost of any feasible flow would be bigger than $2B$. □

With the same reduction, one can show that the special case of the Quickest Cost-Bound Transshipment Problem with only one source s and one sink t (the Quickest Cost-Bound s-t-Flow over Time Problem) is NP-hard.

Theorem 7.2 In the discrete model, the Minimum Cost s-t-Flow over Time Problem can be solved in time $O(Tm \log(Tm)(Tm + Tn \log(Tn)))$ (where n is the number of nodes and m is the number of edges).

Proof: Use Orlin's Algorithm to compute a minimum-cost $(s,0)$-(t,T)-flow in the time-expanded network. $\square$

7.2 A polynomial-time approximation scheme

In this section, we assume that the time horizon, transit times, capacities and supply- and demand-values are integral numbers. Moreover, all transit times have to be positive.

Definition 7.2 Let $\Delta > 0$ be an integer. Let (G,u,τ) be a dynamic flow network and $T \geq 0$ be a time horizon such that all transit times and T are integer multiples of Δ. Then, the *condensed time-expanded network for* (G,u,τ) *and* T is a static flow network (G'_Δ, u'_Δ) such that:

- $V(G'_\Delta) := V(G) \times \{0,\dots,T/\Delta\}$,
- $E(G'_\Delta) := \{((v,\theta),(w,\zeta)) \in V(G'_\Delta) \times V(G'_\Delta) \mid (v = w \text{ and } \zeta = \theta + 1) \text{ or } ((v,w) \in E(G) \text{ and } \tau(v,w)/\Delta = \zeta - \theta)\}$, and
- $u'_\Delta((v,\theta),(v,\theta+1)) := \infty$, and $u'_\Delta((v,\theta),(w,\zeta)) := \Delta \cdot u(v,w)$ for $v \neq w$.

When we have a mapping $b : V(G) \to \mathbb{R}$ with $\sum_{v \in V(G)} b(v) = 0$ and ask for a dynamic b-flow, we model this in the (condensed) time-expanded network by setting $b(v,1) := b(v)$ for all nodes $v \in V(G)$ with $b(v) > 0$ and $b(v,T/\Delta) := b(v)$ for all nodes $v \in V(G)$ with $b(v) < 0$. All remaining nodes get a b-value of 0.

When we consider minimum-cost flow instances, so we have a mapping $c : E(G) \to \mathbb{R}$, then we translate the edge costs to edge costs in the (condensed) time-expanded network in the natural way, i.e. we set all edge costs to 0 on hold edges and set $c((v,\theta),(w,\zeta)) = c(v,w)$ on all other edges.

Continuous flows over time correspond to flows in the (condensed if possible) time-expanded network of the same cost (and vice-versa):

Lemma 7.3 Let $\Delta > 0$ be an integer. Let (G,u,τ) be a dynamic flow network with a time horizon $T \geq 0$ such that all transit times and T are integer multiples of Δ. Let $b : V(G) \to \mathbb{R}$ be a mapping with $\sum_{v \in V(G)} b(v) = 0$.

(a) If g is a static b-flow in the condensed time-expanded network, then there is a dynamic b-flow f in (G,u,τ) with time horizon T such that $c(f) = c(g)$. Moreover, g can be transformed efficiently into f.

(b) If f is a dynamic b-flow in (G,u,τ) with time horizon T, then there is a static b-flow g in the condensed time-expanded network with $c(g) = c(f)$.

Proof: The proof is more or less the same as the proof of Lemma 2.1.

(a) For an edge $e \in (v,w)$ in G and $\theta \in (0,T]$, set $f_e(\theta) = g((v,\lceil\frac{\theta}{\Delta}\rceil),(w,\lceil\frac{\theta+\tau(e)}{\Delta}\rceil))/\Delta$. Then, the flow f respects the capacity constraints because g respects them. We will show that f is a b-flow: For $v \in V(G)$ and $\theta \in (0,T]$, we have, with $k = \lceil\frac{\theta}{\Delta}\rceil$:

$$\begin{aligned}
\mathrm{ex}_f(v,\theta) &= \sum_{e\in\delta^-(v)}\int_0^{\theta-\tau(e)} f_e(\zeta)d\zeta - \sum_{e\in\delta^+(v)}\int_0^{\theta} f_e(\zeta)d\zeta \\
&= \sum_{i=1}^{k}\left(\sum_{e\in\delta^-((v,i))} g(e) - \sum_{e\in\delta^+((v,i))} g(e)\right) \\
&\quad -\left(k-\frac{\theta}{\Delta}\right)\left(\sum_{e\in\delta^-((v,k))} g(e) - \sum_{e\in\delta^+((v,k))} g(e)\right) \\
&= -b((v,1)) - b((v,k)) + g((v,k),(v,k+1)) \\
&\quad -\left(k-\frac{\theta}{\Delta}\right)\Big(g((v,k),(v,k+1)) - g((v,k-1),(v,k)) - b((v,k))\Big),
\end{aligned}$$

so $\mathrm{ex}_f(v,\theta) \geq \min\{-b(v),0\}$ and $\mathrm{ex}_f(v,T) = -b(v)$. Moreover, we have $c(f) = c(g)$ because of $\int_{(i-1)\Delta}^{i\Delta} f_e(\theta) = g((v,i),(w,(i+\tau(e)/\Delta)))$ for $i \in \{1,\dots T/\Delta\}$ and $e=(v,w)\in E(G)$.

(b) For an edge $e=(v,w)$ in G and $\theta \in \{1,\dots,T/\Delta\}$, define $g((v,\theta),(w,\theta+\frac{\tau(e)}{\Delta})) = \int_{\Delta\theta-\Delta}^{\Delta\theta} f_e(\zeta)d\zeta$. For a node v set $g((v,\theta),(v,\theta+1)) = \mathrm{ex}_f(v,\Delta\theta) + b((v,1))$. Then, g obviously respects the capacity constraints and is nonnegative. Moreover, the balance (with respect to g) of a node (v,θ) which is not a source or a sink is

$$\begin{aligned}
\mathrm{ex}_g((v,\theta)) &= \mathrm{ex}_f(v,\Delta\theta-\Delta) + b((v,1)) \\
&\quad + \sum_{e\in\delta^-(v)}\int_{\Delta\theta-\tau(e)-\Delta}^{\Delta\theta-\tau(e)} f_e(\zeta)d\zeta \\
&\quad - \sum_{e\in\delta^+(v)}\int_{\Delta\theta-\Delta}^{\Delta\theta} f_e(\zeta)d\zeta - \mathrm{ex}_f(v,\Delta\theta) - b((v,1)) = 0,
\end{aligned}$$

because

$$\begin{aligned}
\mathrm{ex}_f(v,\Delta\theta-\Delta) - \mathrm{ex}_f(v,\Delta\theta) &= \sum_{e\in\delta^-(v)}\int_0^{\Delta\theta-\theta-\tau(e)} f_e(\zeta)d\zeta - \sum_{e\in\delta^+(v)}\int_0^{\Delta\theta-\theta} f_e(\zeta)d\zeta \\
&\quad - \sum_{e\in\delta^-(v)}\int_0^{\Delta\theta-\tau(e)} f_e(\zeta)d\zeta + \sum_{e\in\delta^+(v)}\int_0^{\Delta\theta} f_e(\zeta)d\zeta \\
&= -\sum_{e\in\delta^-(v)}\int_{\Delta\theta-\tau(e)-\Delta}^{\Delta\theta-\tau(e)} f_e(\zeta)d\zeta + \sum_{e\in\delta^+(v)}\int_{\Delta\theta-\Delta}^{\Delta\theta} f_e(\zeta)d\zeta.
\end{aligned}$$

For a source $(v,1)$ we have

$$\mathrm{ex}_g((v,1)) = \sum_{e\in\delta^-(v)}\int_{-\tau(e)}^{\Delta-\tau(e)} f_e(\zeta)d\zeta - \sum_{e\in\delta^+(v)}\int_0^{\Delta} f_e(\zeta)d\zeta - \mathrm{ex}_f(v,\Delta) - b((v,1)) = -b((v,1)).$$

And for a sink $(v, T/\Delta)$

$$\begin{aligned}
\mathrm{ex}_g((v, T/\Delta)) &= \mathrm{ex}_f(v, T-\Delta) + b((v,1)) \\
&\quad + \sum_{e \in \delta^-(v)} \int_{T-\tau(e)-\Delta}^{T-\tau(e)} f_e(\zeta) d\zeta \\
&\quad - \sum_{e \in \delta^+(v)} \int_{T-\Delta}^{T} f_e(\zeta) d\zeta \\
&= \mathrm{ex}_f(v, T) = -b((v, T/\Delta)).
\end{aligned}$$

This shows that g is a b-flow in the condensed time-expanded network. By construction, it has the same cost as f. □

Instead of computing a flow in the time-expanded network, we rather want to compute it in the condensed time-expanded network. To this end, we have to round up transit times and the time horizon to integer multiples of Δ. In order to get an approximation scheme, we have to show that a flow in the time-expanded network can be transformed into a flow in the condensed time-expanded network and vice versa without changing the cost and the time horizon too much. The direction from the condensed to the non-condensed network, i.e. transforming a dynamic flow with respect to rounded transit times into a dynamic flow with respect to the given transit time, is trivial if we allow internal storage (and this is what we will do).

The other direction is more difficult due to possible violations of density constraints. For example, consider two edges (v, w) and (u, w) with transit times $\tau(v, w) = \Delta$ and $\tau(u, w) = \frac{\Delta}{2}$. If we send flow of size 1 both along (v, w) and (u, w) in the interval $[\theta, \theta + \frac{\Delta}{2})$, the incoming flow in w will never be larger than 1 hence an outgoing edge with capacity 1 would be sufficient. Now if we round up all transit time to integer multiples of Δ and send flow along (v, w) and (u, w) as described above, the incoming flow in w will be 2 in the interval $[\theta + \Delta, \theta + \frac{3\Delta}{2})$ which can lead to a violation of capacity constraints on edges leaving w.

However, the good news is that though we may run into large violations of density constraints, these violations can occur only during quite short time intervals. We will make use of this fact by smoothing flow along source-sink-paths.

Lemma 7.4 Let (G, u, τ) be a dynamic flow network with positive transit times, $b : V(G) \rightarrow \mathbb{R}$ a mapping, and let f be a continuous dynamic b-flow in (G, u, τ) with time horizon T without internal storage. Then, there is a set $\mathcal{P}$ of paths in G together with a vector $(f_P)_{P \in \mathcal{P}}$ of integrable functions $f_P : [0, T] \rightarrow \mathbb{R}_{\geq 0}$ where each path in $\mathcal{P}$ starts at a source and ends at a sink such that for each edge $e = (v, w) \in E(G)$ and each interval $[\theta, \zeta] \subseteq [0, T]$ with $\theta < \zeta$ we have

$$\int_{\theta}^{\zeta} f_e(\eta) d\eta = \sum_{P \in \mathcal{P} : e \in E(P)} \int_{\theta - \tau(P_{\rightarrow v})}^{\zeta - \tau(P_{\rightarrow v})} f_P(\eta) d\eta$$

where $\tau(P_{\rightarrow v})$ is the transit time of P between the start node of P and v. We call the set $\mathcal{P}$ together with the vector $(f_p)_{P \in \mathcal{P}}$ a *dynamic flow decomposition of f*.

Proof: Let $P_1, \ldots, P_k$ denote all paths in G that start at a source and end at a sink (in any order). Set $f'_e(\theta) = f_e(\theta)$ for all $e \in E(G)$ and $\theta \in [0, T]$. Now for $i \in \{1, \ldots, k\}$ and $P = P_i$, perform the following steps: Set $f_P(\theta) = \min\{f'_e(\theta + \tau(P_{\rightarrow v})) \mid e = (v, w) \in E(P)\}$ for $\theta \in [0, T]$ (we may assume that all flow values are zero outside $[0, T]$). Set $f'_e(\theta + \tau(P_{\rightarrow v})) = f'_e(\theta + \tau(P_{\rightarrow v})) - f_P(\theta)$ for $e = (v, w) \in E(P)$.

Then, the set $\mathcal{P}$ containing all P_i for which $\int_0^T f_{P_i}(\theta) d\theta \neq 0$ is the path set of a dynamic flow decomposition of f. □

Remarks:

- An exponential number of paths in the flow decomposition can be necessary even if we have only one source s and one sink t. For an example, consider an s-t-path with n nodes in which we replace all edges by two parallel edges. A flow may make use of all 2^{n-1} different choices for an s-t-path.
- There can be several dynamic flow decompositions for the same dynamic flow (e.g. in the network described in the previous remark, if flow is sent along all edges at the same time).
- The function $f_e(\theta) - \sum_{P \in \mathcal{P} : e \in E(P)} f_P(\theta - \tau(P_{\rightarrow v}))$ (for an edge $e = (v, w)$) may not be zero for some values of θ, only its integral over any interval of positive length must be zero.
- Any set $\mathcal{P}$ of source-sink-paths together with integrable functions $f_P : [0, T] \rightarrow \mathbb{R}_{\geq 0}$ defines via $f_e(\theta) = \sum_{P \in \mathcal{P} : e \in E(P)} f_P(\theta)$ for any transit times τ a dynamic b-flow (for some appropriate b-values and edge capacities).

Lemma 7.5 Let (G, u, τ) be a dynamic flow network with integral transit times, edge costs $c : E(G) \rightarrow \mathbb{R}$ and a mapping $b : V(G) \rightarrow \mathbb{R}$ such that $\sum_{v \in V(G)} b(v) = 0$. Let f be a dynamic b-flow in (G, u, τ) with internal storage. Then, there is a dynamic b-flow $\tilde{f}$ in (G, u, τ) that does not make use of internal storage.

Proof: Let g be a static flow in the time-expanded network that corresponds to a minimum-cost dynamic flow in (G, u, τ) (Lemma 7.3 with $\Delta = 1$ guarantees that such a flow exists).

In this proof, we may enlarge the time-expanded network to arbitrary time steps outside $\{0, \dots, T\}$ in the canonical way. However, the flow that we finally compute is a flow in the time-expanded network with time horizon T.

For a vertex (v, θ) in the time-expanded network, let $\gamma_g(v, \theta) := \sum_{e=(v,w)\in\delta^+_G(v)} g((v, \theta), (w, \theta + \tau(e))) - \sum_{e=(w,v)\in\delta^-_G(v)} g((v, \theta - \tau(e)), (w, \theta))$. We will show that there is a static flow h in the time-expanded network that corresponds to a minimum-cost dynamic flow such that $\gamma_h(v, \theta) = 0$ for all nodes v that are neither sources nor sinks and all $\theta \in \{0, \dots, T\}$. Let h be a static flow in the time-expanded network that corresponds to a minimum-cost dynamic flow such that $\sum_{x\in V(G)} \sum_{\zeta=0}^{T} |\gamma_h(x, \zeta)|$ is minimum (such a flow must exist because we minimize a convex function over a compact convex set).

Claim: h does not make use of any internal hold edges.

Proof of the claim: Assume that h uses the hold edge $((v, \theta), (v, \theta + 1))$ where v is neither a source nor a sink. Then, we can assume that $\gamma_h(v, \theta) < 0$. Since h meets the weak flow conservation constraint, there must be a number q such that $\gamma_h(v, \theta + q) > 0$. Let q as small as possible with this property.

For an edge $e = (x, y) \in E(G)$ and $\theta \in [0, T]$, color the edge $((x, \zeta), (y, \zeta + \tau(e)))$ in the time-expanded network red if $h((x, \zeta), (y, \zeta + \tau(e))) < h((x, \zeta - q), (y, \zeta + \tau(e) - q))$ and blue if $h((x, \zeta), (y, \zeta + \tau(e))) > h((x, \zeta - q), (y, \zeta + \tau(e) - q))$. All remaining edges are uncolored. Hence, the red edges and the reverse edge of the blue edges belong to the residual graph of h.

Now consider the set R of all nodes (x, ζ) that are reachable from $(v, \theta + q)$ in the time-expanded network using only blue edges and the reverse edges of red edges. We claim that this set must contain a vertex (z, μ) with $\gamma_h(z, \mu) < \gamma_h(z, \mu - q)$. Otherwise, we may consider the set $R' = \{(x, \zeta - q) \mid (x, \zeta) \in R\}$. Then, if we restrict the time-expanded network to transit edges (skipping the hold edges), R' would be a set whose total balance is strictly bigger than the total balance of R while the net flow leaving R' is at least as big as the net flow leaving R. This is a contradiction.

Hence, let $(z, \mu) \in R$ be a vertex with $\gamma_h(z, \mu) < \gamma_h(z, \mu - q)$. Let P be a $(v, \theta + q)$-(z, μ) path using only blue edges and the reverse edges of red edges. Let P' be the path P shifted by q to the left, i.e. for an $(x, \zeta) \in V(P)$ we have $(x, \zeta - q)$ as the corresponding vertex in $V(P')$.

Then for a small $\varepsilon > 0$, we can reduce flow on blue edges of P by ε and increase it on red edges on P by ε. For the corresponding edges in P' we do it the other way round, i.e. we increase flow on edges corresponding to blue edges by ε and reduce it on edges corresponding to red edges by ε. Moreover, we reduce h on the hold edges from (v, θ) to $(v, \theta + q)$ and increase it on the hold edges from $(z, \mu - q)$ to (z, μ) by ε. If ε is small enough, this does not lead to negative flow values and does not exceed the edge capacities. See Figure 8 for an example, where "+" and "-" indicate if we increase or decrease flow.

Since all copies of an edge in the time-expanded network have the same cost, this change has not effect on the cost of the flow. Moreover, the modification reduces $|\gamma_h(v, \theta)|$ and $|\gamma_h(v, \theta + q)|$ and

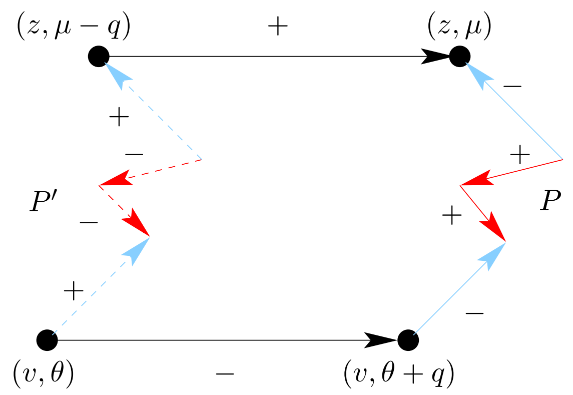

Figure 8: Paths P and P'.

does not increase $|\gamma_h(x, \mu - q)| + |\gamma_h(x, \mu)|$.

Hence, we can reduce $\sum_{x \in V(G)} \sum_{\zeta=0}^{T} |\gamma_h(x, \zeta)|$ which is a contradiction. This proves the claim and the lemma. □

Theorem 7.6 Let (G, u, τ) be a dynamic flow network with integral positive transit times, with edge costs $c : E(G) \to \mathbb{R}_{\geq 0}$, and let $b : V(G) \to \mathbb{R}$ be a mapping with $\sum_{v \in V(G)} b(v) = 0$. Let $C > 0$ be a number. Let $T \geq 0$ be an integral time horizon such that there is a continuous dynamic b-flow in (G, u, τ) with cost at most C. Let $\varepsilon > 0$ be given. Define $\Delta := \frac{\varepsilon^2 T}{|V(G)|}$, $T' := \lceil \frac{(1+\varepsilon)^2 T}{\Delta} \rceil \Delta$, and $\tau'(e) = \lceil \frac{\tau(e)}{\Delta} \rceil \Delta$ for $e \in E(G)$. Then, there is a continuous dynamic b-flow in (G, u, τ') with time horizon T' and cost at most C.

Proof: Let f be a dynamic b-flow in (G, u, τ) with time horizon T and cost at most C.

Define a dynamic flow g by $g_e(\theta) = f_e(\frac{\theta}{1+\varepsilon})$ for $e \in E(G)$ and $\theta \in [0, (1+\varepsilon)T]$. Then, one can check easily that g is a dynamic $(1+\varepsilon)b$-flow in $(G, u, (1+\varepsilon)\tau)$ with time horizon $(1+\varepsilon)T$. Its cost is $c(g) = (1+\varepsilon)c(f)$. Since we allow internal storage, there must also be a dynamic $(1+\varepsilon)b$-flow $\tilde{g}$ in (G, u, τ) with time horizon $(1+\varepsilon)T$ and cost $c(\tilde{g}) \leq (1+\varepsilon)c(f)$.

Using the previous lemma, we can get rid of the internal storage, so there must be a dynamic $(1+\varepsilon)b$-flow h in (G, u, τ) with time horizon $(1+\varepsilon)T$ and cost $c(h) \leq (1+\varepsilon)c(f)$ such that h does not make use of internal storage.

By Lemma 7.4, we may assume that we can decompose h into paths $\mathcal{P}$ with path flows h_P for $P \in \mathcal{P}$ such that for $e = (v, w) \in E(G)$ and $\theta \in [0, (1+\varepsilon)T]$ we have $h_e(\theta) = \sum_{P \in \mathcal{P}: e \in E(P)} h_P(\theta - \tau(P_{\to v}))$.

If we try to use the path flow decomposition to define a dynamic flow in (G, u, τ'), then we may violate the capacity constraints significantly. In order to meet them at least approximately, we smooth the path flows h_p by defining $\tilde{h}_P(\theta) := \frac{1}{\varepsilon T} \int_{\theta - \varepsilon T}^{\theta} h_P(\zeta) d\zeta$ for $\theta \in [0, (1+2\varepsilon)T]$ and $P \in \mathcal{P}$.

For an antiderivative H_P of h_P we have

$$\begin{aligned}\int_0^{(1+2\varepsilon)T} \frac{1}{\varepsilon T}\int_{\theta-\varepsilon T}^{\theta} h_P(\zeta)\; d\zeta\; d\theta &= \int_0^{(1+2\varepsilon)T} \frac{1}{\varepsilon T}(H_P(\theta) - H_P(\theta-\varepsilon T))d\theta \\ &= \frac{1}{\varepsilon T}\left(\int_0^{(1+2\varepsilon)T} H_P(\theta)d\theta - \int_{-\varepsilon T}^{(1+\varepsilon)T} H_P(\theta)d\theta\right) \\ &= \frac{1}{\varepsilon T}\int_{(1+\varepsilon)T}^{(1+2\varepsilon)T} H_P(\theta)d\theta \\ &= \frac{1}{\varepsilon T}\varepsilon T \cdot H_P((1+\varepsilon)T) \\ &= \int_0^{(1+\varepsilon)T} h_P(\theta)d\theta,\end{aligned}$$

so the amount of flow sent through P by h_P up to time $(1+\varepsilon)T$ and by $\tilde{h}_P$ up to time $(1+2\varepsilon)T$ is the same.

These smoothed path flows define a new dynamic flow $\tilde{h}$ in (G, u, τ') by setting $\tilde{h}_e(\theta) = \sum_{P\in\mathcal{P}:e\in E(P)} \tilde{h}_P(\theta - \tau'(P_{\rightarrow v}))$ for $e = (v, w) \in E(G)$ and $\theta \in [0, (1+2\varepsilon)T + \varepsilon^2 T]$. The flow $\tilde{h}$ is a $(1+\varepsilon)b$-flow with cost at most $(1+\varepsilon)c(f)$ (this is because h has these properties and the amount of flow sent through a path P is the same for h and $\tilde{h}$).

Note that for a path $P \in \mathcal{P}$ we have $0 \le \tau'(P) - \tau(P) \le |V(G)|\Delta = \varepsilon^2 T$. Hence $\tilde{h}$ has a time horizon of $(1 + 2\varepsilon + \varepsilon^2)T$.

For an edge $e = (v, w) \in E(G)$ and $\theta \in [0, (1+2\varepsilon+\varepsilon^2)T]$ we have

$$\begin{aligned}\tilde{h}_e(\theta) &= \sum_{P\in\mathcal{P}:e\in E(P)} \tilde{h}_P(\theta - \tau'(P_{\rightarrow v})) \\ &= \frac{1}{\varepsilon T}\sum_{P\in\mathcal{P}:e\in E(P)} \int_{\theta-\tau'(P_{\rightarrow v})-\varepsilon T}^{\theta-\tau'(P_{\rightarrow v})} h_P(\zeta)d\zeta \\ &\le \frac{1}{\varepsilon T}\sum_{P\in\mathcal{P}:e\in E(P)} \int_{\theta-\tau(P_{\rightarrow v})-\varepsilon T-\varepsilon^2 T}^{\theta-\tau(P_{\rightarrow v})} h_P(\zeta)d\zeta \\ &= \frac{1}{\varepsilon T}\int_{\theta-\varepsilon T-\varepsilon^2 T}^{\theta} \sum_{P\in\mathcal{P}:e\in E(P)} h_P(\zeta - \tau(P_{\rightarrow v}))d\zeta \\ &= \frac{1}{\varepsilon T}\int_{\theta-\varepsilon T-\varepsilon^2 T}^{\theta} h_e(\zeta)d\zeta \\ &\le \frac{\varepsilon T + \varepsilon^2 T}{\varepsilon T}u(e) \\ &= (1+\varepsilon)u(e)\end{aligned}$$

Hence, the dynamic flow q which is defined by $q_e(\theta) = \frac{\tilde{h}_e(\theta)}{(1+\varepsilon)}$ is a dynamic b-flow in (G, u, τ') with cost at most C and time horizon $(1 + 2\varepsilon + \varepsilon^2)T = (1+\varepsilon)^2 T \le T'$. $\square$

Theorem 7.7 For $\varepsilon > 0$, a $(1+\varepsilon)$-approximate solution of the Bounded-Cost Quickest Transshipment Problem with a single source and a single sink, can be found in a running time that is polynomial in the size of the network and $\frac{1}{\varepsilon}$.

Proof: We may assume that $\varepsilon < 3$. Let $\varepsilon_0 > 0$ be small enough such that $(1+\varepsilon_0)^2 \leq (1+\frac{\varepsilon}{3})$. Given a time horizon T, we apply the previous theorem, so if there is a dynamic flow in the given network with time horizon T and cost at most C, then there is also a dynamic flow f for time horizon $T' := \lceil\frac{(1+\varepsilon_0)^2 T}{\Delta}\rceil\Delta$ and transit times $\tau'(e) = \lceil\frac{\tau(e)}{\Delta}\rceil\Delta$ (for $e \in E(G)$) with cost at most C. Moreover, Lemma 7.3 shows that such a flow can be computed efficiently in the condensed time-expanded network. In other words, we can either decide that no flow with time horizon T and cost at most C exists, or compute a flow with cost at most C and a time horizon of $(1+\frac{\varepsilon}{3})T$.

Now we apply binary search to find an approximation to the smallest time horizon T^* for which a dynamic flow with cost at most C exists. We start with a lower bound of e.g. $T_L = 1$ and an upper bound of e.g. $T_U = \sum_{v\in V(G)} |b(v)| + \sum_{e\in E(G)} \tau(e)$. The fastest way to reduce the ratio $\frac{T_U}{T_L}$ is to apply geometric binary search (see Hassin [1992]). This means that we apply the check described above to $T = \sqrt{\frac{T_U T_L}{1+\frac{\varepsilon}{3}}}$. Then, we either know that $T^* > T$, or we know that $T^* \leq (1+\frac{\varepsilon}{3})T$. In both cases, we get new bounds T'_L and T'_U whith $\frac{T'_U}{T'_L} \leq \sqrt{\frac{T_U}{T_L}(1+\frac{\varepsilon}{3})}$. So after k iterations, the ratio between the upper and the lower bound is

$$Z^{\frac{1}{2^k}} \prod_{i=1}^{k} \left(1+\frac{\varepsilon}{3}\right)^{\frac{1}{2^i}} = Z^{\frac{1}{2^k}} \left(1+\frac{\varepsilon}{3}\right)^{\sum_{i=1}^k \frac{1}{2^i}} < Z^{\frac{1}{2^k}} \left(1+\frac{\varepsilon}{3}\right)$$

where Z is the initial value of $\frac{T_U}{T_L}$. After $k \geq \log\log Z - \log\log(1+\frac{\varepsilon}{3})$ iterations, we have $2^k \geq \frac{\log Z}{\log(1+\frac{\varepsilon}{3}))}$ and hence $Z^{\frac{1}{2^k}} \leq 1+\frac{\varepsilon}{3}$. Then, we have $Z^{\frac{1}{2^k}}(1+\frac{\varepsilon}{3}) \leq (1+\frac{\varepsilon}{3})^2 \leq 1+\epsilon$ (because $\varepsilon < 3$), so the ratio between the upper and the lower bound is at most $1+\epsilon$ which means that we have a $(1+\epsilon)$-approximation. $\square$

Remark: The b-flow that we compute only meets the weak flow conservation rule, i.e. it makes use of internal storage, though Lemma 7.5 shows that there is an optimum solution without internal storage. The algorithm can be modified such that it does not use the hold edges. To this end, compute a decomposition of the dynamic flow in (G, u, τ') into paths and smooth the flow along these paths. This leads to a dynamic flow in the unmodified instance that almost respects the capacity constraints (it respects $(1+\varepsilon)u(e)$ instead of $u(e)$) and does not make use of internal storage. The capacity constraints can be met exactly if we compute a $(1+\varepsilon)b$-flow with cost at most $(1+\varepsilon)C$ in (G, u, τ'), translate this into a flow in (G, u, τ) and divide everything by $(1+\varepsilon)$. For a detailed description see Fleischer and Skutella [2007].

References

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

Baumann, N., Skutella, M. [2009]: *Solving Evacuation Problems Efficiently: Earliest Arrival Flows with Multiple Sources.* Mathematics of Operations Research, 34, 2009, 499512.

Burkard, R.E., Dlaska, K., Klinz, B. [1993]: *The quickest flow problem.* ZOR – Methods and Models of Operations Research, 37, 1993, 31–59.

Chen, Y.L., Chin, Y.H. [1990]: *The quickest path problem.* Computers and Operations Research, 17, 2, 1990, 153–161.

Cheriyan, J., Hagerup, T., Mehlhorn, K. [1996]: *An $o(n^3)$-time maximum-flow algorithm.* SIAM Journal on Computing, 25, 1996, 1144–1170.

Dijkstra, E.W. [1959]: *A note on two problems in connexion with graphs.* Numerische Mathematik I, 1959, 269–271.

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

Fleischer, L., Skutella, M. [2003]: *Minimum cost flows over time without intermediate storage.* Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms, 2003, 66–75.

Fleischer, L., Skutella, M. [2007]: *Quickest flows over time.* SIAM Journal on Computing, 36, 6, 2007, 1600–1630.

Fleischer, L., Tardos, É. [1998]: *Efficient continuous-time dynamic flow algorithms.* Operations Research Letters, 23, 1998, 71–80.

Ford, L.R., Fulkerson, D.R. [1958]: *Constructing maximal dynamic flows from static flows.* Operations Research, 6, 1958, 71–80.

Fredman, M.L., Tarjan, R.E. [1987]: *Fibonacci heaps and their uses in improved network optimization problems.* Journal of the Association for Computing Machinery, 34, 1987, 596–615.

Gale, D. [1957]: *A theorem on flows in networks.* Pacific Journal if Mathematics, 7, 1073–1082, 1957.

Garey, M.R. and Johnson, D.S. [1979]: *Computers and Intractability. A Guide to the Theory of NP-Completeness.* Freeman, 1979.

Goldberg, A.V., Tarjan, R.E. [1988]: *A new approach to the maximum-flow problem.* Journal of the Association for Computing Machinery, 35, 1988, 910–940.

Grötschel, M., Lovász, L., Schijver, A. [1988]: *Geometric Algorithms and Combinatorial Optimization.* Springer, 1988.

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

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

Hall, A., Langkau, K., Skutella, M. [2007]: *An FPTAS for quickest multicommodity flows with inflow-dependent transit times.* Algorithmica, 47, 2007, 299–321.

Hassin, R. [1992]: *Approximation Schemes for the restricted shortest path problem.* Mathematics of Operations Research, 17, 1, 1992, 36–42.

Hoppe, B., Tardos, É. [1994]: *Polynomial time algorithms for some evacuation problems.* Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms, 1994, 433–441.

Hoppe, B., Tardos, É. [2000]: *The quickest transshipment problem.* Mathematics of Operations Research, 25, 1, 2000, 36–62.

Klinz, B., Woeginger, G.J. [2004]: *Minimum-cost flows: The series-parallel case.* Networks, 43, 3, 2004, 153–162

Korte, B., Vygen, J. [2012]: *Combinatorial Optimization: Theory and Algorithms.* Fifth edition, Springer, 2012

Martins, E.Q.V., dos Santos, J.L.E. [1997]: *An algorithm for the quickest path problem.* Operations Research Letters, 20, 1997, 195–198.

Megiddo, N. [1974]: *Optimal flows in networks with multiple sources ans sinks.* Mathematical Programming, 7, 1974, 97–107.

Megiddo, N. [1979]: *Combinatorial optimization with rational objective functions.* Mathematics of Operations Research, 4, 1979, 414–424.

Orlin, J.B. [1993]: *A faster strongly polynomial minimum cost flow algorithm.* Operations Research 41, 1993, 338–350.

Philpott, A.B. [1990]: *Continuous-time flows in networks.* Mathematics of Operations Research, 15, 4, 1990, 640–661.

Rosen, J.B., Sun, S.-Z., Xue, G.L. [1991]: *Algorithms for the quickest path problem and the enumeration of quickest paths.* Computers and Operations Research, 18, 6, 1991, 579–584.

Schmidt, M., Skutella, M. [2010]: *Earliest arrival flows in networks with multiple sinks.* Electronic Notes in Discrete Mathematics, 36, 2010, 607–614. International Symposium on Combinatorial Optimization, 2010.

Schrijver, A. [2002]: *On the history of the transportation and maximum flow problems.* Mathematical Programming, 91, 2002, 437–445.

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

Skutella, M. [2008]: *Introduction to Network Flows over Time.* in: Research Trends in Combinatorial Optimization, W.J. Cook, L. Lovász, J. Vygen (editors), Springer, 2008, 451–482.

Wilkinson, W.L. [1971]: *An algorithm for universal maximal dynamic flows in networks.* Operations Research, 19, 1971, 1602–1612.

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