# Generalized Maximum Flows over Time*

Martin Groß and Martin Skutella

Fakultät II – Mathematik und Naturwissenschaften,
Institut für Mathematik, Sekr. MA 5-2
Technische Universität Berlin, Straße des 17. Juni 136,
10623 Berlin, Germany
{gross,skutella}@math.tu-berlin.de

**Abstract.** Flows over time and generalized flows are two advanced network flow models of utmost importance, as they incorporate two crucial features occurring in numerous real-life networks. Flows over time feature time as a problem dimension and allow to realistically model the fact that commodities (goods, information, etc.) are routed through a network over time. Generalized flows allow for gain/loss factors on the arcs that model physical transformations of a commodity due to leakage, evaporation, breeding, theft, or interest rates. Although the latter effects are usually time-bound, generalized flow models featuring a temporal dimension have never been studied in the literature.
In this paper we introduce the problem of computing a generalized maximum flow over time in networks with both gain factors and transit times on the arcs. While generalized maximum flows and maximum flows over time can be computed efficiently, our combined problem turns out to be NP-hard and even completely non-approximable. A natural special case is given by lossy networks where the loss rate per time unit is identical on all arcs. For this case we present a (practically efficient) FPTAS that also reveals a surprising connection to so-called earliest arrival flows.

## 1 Introduction

Two crucial characteristics of network flows occurring in real-world applications are flow variation over time and physical transformation of flow resulting in a lesser or greater amount of flow. These characteristics are not captured by standard network flow models known from the literature.

Ford and Fulkerson [10, 11] introduce the notion of flows over time (also called dynamic flows) which model flow variation over time as well as the fact that flow does not travel instantaneously through a network but requires a certain amount of time to travel through each arc. Various interesting examples and applications can be found in the survey articles of Aronson [1] and Powell, Jaillet, and Odoni [27].

Generalized flows have been suggested as a tool in production planning as early as 1939 by Kantorovich [20]. They model the situation where flow is not

* Supported by the DFG Research Center MATHEON "Mathematics for key technologies" in Berlin.

necessarily conserved on every arc but may be physically transformed due to leakage, evaporation, breeding, theft, or interest rates. We refer to the PhD thesis of Wayne [33] for an in-depth treatment of various generalized flow problems.

Both from a practical and theoretical point of view, it seems to be natural to consider a combination of both flow models. However, to the best of our knowledge, generalized flows over time are considered for the first time in this paper. In particular, we hope that the paper will also stimulate further research in this interesting and challenging direction.

*Model and Problem.* Consider a directed graph $G$ with node set $V(G)$, arc set $E(G)$, capacities $u_e \in \mathbb{R}_{\geq 0}$, transit times $\tau_e \in \mathbb{N}_0$ and gain factors $\gamma_e \in \mathbb{R}_{>0}$ on the arcs $e \in E(G)$. These arc attributes have the following meaning: the capacity of an arc limits the amount of flow that can enter the arc in any time step. For each unit of flow entering the tail of an arc $e \in E(G)$ at time $\theta$, exactly $\gamma_e$ flow units leave the arc at its head at time $\theta + \tau_e$. We assume that we are given a single source node $s \in V(G)$ without incoming arcs and a single sink node $t \in V(G)$ without outgoing arcs and $s \neq t$. Furthermore, we are given a time horizon $T \in \mathbb{N}$. Combined, we call $(G, u, \tau, \gamma, s, t, T)$ a network. For a node $v \in V(G)$, we denote the outgoing and incoming arcs by $\delta^+_G(v)$ and $\delta^-_G(v)$, respectively.

A generalized flow over time $f : E(G) \times \{0, 1, \ldots, T-1\} \to \mathbb{R}_{\geq 0}$ in such a network is a mapping that assigns flow values $f_{e,\theta} \in [0, u_e]$ to every arc $e \in E(G)$ at every point in time[1] $\theta \in \{0, 1, \ldots, T-1\}$ with respect to (generalized) flow conservation:

$$\sum_{e \in \delta^-(v)} \sum_{\xi=0}^{\theta-\tau_e} \gamma_e f_{e,\xi} \geq \sum_{e \in \delta^+(v)} \sum_{\xi=0}^{\theta} f_{e,\xi} \quad v \in V(G) \setminus \{s\}, \theta \in \{0, \ldots, T-1\}, \quad (1)$$

$$\sum_{e \in \delta^-(v)} \sum_{\xi=0}^{T-\tau_e-1} \gamma_e f_{e,\xi} = \sum_{e \in \delta^+(v)} \sum_{\xi=0}^{T-1} f_{e,\xi} \quad v \in V(G) \setminus \{s, t\}.$$

Moreover, we require that $f_{e,\theta} = 0$ for all $\theta \geq T - \tau_e$ such that no flow remains in the network at time $T$. The above definition of flow conservation allows storage of flow in nodes; this is referred to as *holdover*. If holdover is not desired, we require that equality holds in (1) for all $v \in V(G) \setminus \{s, t\}$. The value $|f|$ of a generalized flow over time $f$ is the amount of flow sent to the sink within the time horizon: $|f| := \sum_{e \in \delta^-(t)} \sum_{\xi=0}^{T-\tau_e-1} \gamma_e f_{e,\xi}$. Similarly, we write $|x|$ for the value of a static generalized flow $x$. The *arrival pattern* of a flow over time is a mapping that assigns to every time step $\theta \in \{0, 1, \ldots, T-1\}$ the total amount of flow that has arrived at the sink in the time steps $\{0, \ldots, \theta\}$. The generalized maximum flow over time problem asks for a generalized flow over time of maximum value in a given network $(G, u, \tau, \gamma, s, t, T)$.

[1] In this paper, we use a discrete time model with time steps $0, 1, \ldots, T-1$ for a given time horizon $T \in \mathbb{N}$. Results in this setting often carry over to continuous time models; see, e. g., Fleischer and Tardos [7].

*Previous Work.* There has been considerable research on the static generalized maximum flow problem (i. e., our problem without transit times and temporal dimension) and on the maximum flow over time problem (i. e., our problem without gain factors). Since generalized flow problems can be formulated as linear programs [4], they can be solved in polynomial time. The first combinatorial polynomial time algorithm for computing generalized maximum flows was proposed by Goldberg, Plotkin and Tardos [14] and has subsequently been improved by Radzik [28, 29]. Fleischer and Wayne [8], Goldfarb and Jin [15], Goldfarb, Jin and Orlin [16], Restrepo and Williamson [30] and Wayne [33, 34] described further polynomial time algorithms. Truemper [32] noted that generalized maximum flow algorithms show several analogies to minimum cost flow algorithms, if the negative logarithm of a gain factor is used as the cost. Nonetheless, these analogies are limited – it is an open problem whether a strongly polynomial time algorithm for the generalized maximum flow problem exists, contrary to the minimum cost flow problem.

Maximum flows over time have been introduced by Ford and Fulkerson [10, 11] in the 1950's. They proposed two techniques for dealing with them – creating a pseudo-polynomially large time-expanded network to reduce their computation to a static maximum flow problem, and a reduction to the static minimum cost flow problem in the given network, which allows solving this problem in strongly polynomial time. Flows over time that maximize the amount of flow sent to the sink at *any* point in time are called *earliest arrival flows* or *universally maximum flows over time*; this concept is due to Gale [13]. Minieka [23] and Wilkinson [35] showed that the successive shortest path algorithm is capable of solving this problem. Hoppe and Tardos [19] as well as Fleischer and Skutella [6] describe different types of FPTASes for this problem. Nonetheless, the complexity of the earliest arrival flow problem is mostly open; it is, for example, unclear whether this problem is $NP$-hard or not. This is partly due to the fact, that the arrival pattern (i.e. the function describing the amount of flow arriving at the sink over time) is a piecewise linear function with exponentially many breakpoints (which follows from the work of Zadeh [36]).

Since both generalized maximum flows and maximum flows over time can be dealt with techniques for minimum cost flows, minimum cost flow over time algorithms might seem attractive candidates for generalized maximum flow over time algorithms. However, Klinz and Woeginger [22] showed that the minimum cost flow over time problem is $NP$-hard.

*Our Contribution.* In Section 3 we examine the complexity of the generalized maximum flow over time problem with arbitrary gain factors and show that there is no polynomial approximation algorithm, even for the special case of series-parallel networks, unless $P = NP$.

For the special case of lossy networks, we show in Section 4 that the concept of condensed time-expanded networks introduced by Fleischer and Skutella [6] can be successfully generalized to the setting of generalized flows over time and yields an FPTAS. Notice, however, that this FPTAS approximates the time horizon rather than the flow value. That is, for a given time horizon $T$ and $\varepsilon > 0$, the

algorithm computes a generalized flow over time with time horizon $(1+\varepsilon)T$ and value at least as big as the value of a maximum generalized flow over time with time horizon $T$.

Section 5 contains the main contribution of this paper. We consider an important special case of the generalized maximum flow over time problem where gain factors are proportional to transit times. Here *proportional* means that there exists a $c \in \mathbb{R}$ such that $\gamma_e = 2^{c \cdot \tau_e}$ for every arc $e \in E(G)$. Such gain factors are motivated by the fact that in many applications effects such as leakage, evaporation or interest rates are strictly time-bound. Also many processes of growth or decay in nature can be captured by such proportional gain factors. Notice that in this setting paths with equal transit time have equal gain factors and vice versa, due to transit times being additive and gain factors being multiplicative along paths.

For the case $c \leq 0$ we reveal interesting connections between generalized maximum flows over time and earliest arrival flows. In Section 5.1 we show how to compute generalized maximum flows over time with a variant of the successive shortest path algorithm on the static network. This result is particularly interesting since – apart from the most basic maximum flow over time problem – hardly any flow over time problem is known to be solvable by a static flow computation on the underlying static network. It also implies that there are always optimal solutions that do not need holdover. As the successive shortest path algorithm requires an exponential number of iterations in the worst case, our algorithm is not polynomial.

Therefore we prove in Section 5.2 that an FPTAS can be obtained by terminating the successive shortest path algorithm after a polynomial number of iterations. We wish to emphasize that this FPTAS approximates the maximum flow value rather than the required time horizon (which FPTASes for flow over time problems normally do).

Finally, in Section 5.3 we show that generalized maximum flows over time have a unique flow arrival pattern at the sink, thereby identifying an interesting connection to earliest arrival flows. In particular, the unique arrival pattern of generalized maximum flows over time is piecewise linear with exponentially many breakpoints in the worst case. In this respect, the algorithm presented in Section 5.1 is as good as it gets.

Finally, in Section 6 we conclude with interesting directions for future research.

## 2 Preliminaries

A *path* in a graph $G$ is a sequence of arcs $P = (e_1 = (v_1, v_2), \ldots, e_k = (v_k, v_{k+1}))$ for a $k \in \mathbb{N}$, $e_1, \ldots, e_k \in E(G)$, $v_1, \ldots, v_{k+1} \in V(G)$ and $v_i \neq v_j$ unless $i = j$. A *cycle* in a graph $G$ is a sequence of arcs $C = (e_1 = (v_1, v_2), \ldots, e_k = (v_k, v_1))$ for a $k \in \mathbb{N}$, $e_1, \ldots, e_k \in E(G)$, $v_1, \ldots, v_k \in V(G)$ and $v_i \neq v_j$ unless $i = j$. We will treat paths and cycles as sets, if the order of the arcs in a path or cycle is not relevant. With this convention, we will now extend the transit times and

gain factors to paths and cycles by defining: $\tau_P := \sum_{e \in P} \tau_e$, $\tau_C := \sum_{e \in C} \tau_e$, $\gamma_P := \prod_{e \in P} \gamma_e$, and $\gamma_C := \prod_{e \in C} \gamma_e$. Cycles $C$ with $\gamma_C = 1$ are called *unit gain cycles*, cycles with $\gamma_C > 1$ *flow-generating cycles* and cycles with $\gamma_C < 1$ *flow-absorbing cycles.*

In the following let $\overleftarrow{E(G)} := \{ \overleftarrow{e} \mid e \in E(G)\}$ denote the set of reverse arcs of $E(G)$, i. e., each arc $e = (v, w) \in E(G)$ has a reverse arc $\overleftarrow{e} := (w, v) \in \overleftarrow{E(G)}$. Moreover, we set $\overleftarrow{\overleftarrow{e}} := e$ for all $e \in E(G)$.

**Definition 1** *The residual network $(G_x, u_x, \tau, \gamma, s, t, T)$ of a generalized flow $x$ in a network $(G, u, \tau, \gamma, s, t, T)$ is defined as follows:*

$$V(G_x) := V(G)$$
$$E(G_x) := \{ e \in E(G) \mid x_e < u_e \} \cup \{ \overleftarrow{e} \mid e \in E(G),\, x_e > 0 \} \subseteq E(G) \cup \overleftarrow{E(G)},$$
$$(u_x)_e := \begin{cases} u_e - x_e & e \in E(G), \\ \gamma_{\overleftarrow{e}} x_{\overleftarrow{e}} & e \in \overleftarrow{E(G)}, \end{cases} \quad \text{for all } e \in E(G_x).$$

*Transit times and gain factors are extended to the reverse edges as follows:*

$$\tau_{\overleftarrow{e}} := -\tau_e \quad \text{and} \quad \gamma_{\overleftarrow{e}} = \frac{1}{\gamma_e}, \quad \text{for all } e \in E(G).$$

**Definition 2** *The time expanded network $(G^T, u^T, \gamma^T, s', t')$ is constructed from a network $(G, u, \tau, \gamma, s, t, T)$ by "copying the network for each time step":*

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

*We call the arcs in $H^T$* holdover arcs. *If holdover is forbidden at intermediate nodes, we simply let $H^T := \{ (v_\theta, v_{\theta+1}) \mid v \in \{s, t\}, \theta \in \{0, \ldots, T-2\} \}$. Capacities and gain factors are extended as follows:*

$$u^T_{e'} := \begin{cases} u_e & e' = e_\theta \in E(G)^T, \\ \infty & e' \in H^T, \end{cases} \qquad \gamma^T_{e'} := \begin{cases} \gamma_e & e' = e_\theta \in E(G)^T, \\ 1 & e' \in H^T, \end{cases}$$

*for all $e' \in E(G^T)$. Finally, we set $s' := s_0$ and $t' := t_{T-1}$.*

The main advantage of using time expanded networks is that they allow us to reduce a problem with a temporal dimension into a problem without it. This comes at the cost of a pseudopolynomial increase in size, however. The following theorem (based on the results of Ford and Fulkerson [10, 11]) formalizes the relation of flows over time in the static network and static flows in the time-expanded network.

**Theorem 1 (Time expansion).** *Let $(G, u, \tau, \gamma, s, t, T)$ be a network with the corresponding time expanded network $(G^T, u^T, \gamma^T, s', t')$. Then, for a given generalized flow over time $x$ in $(G, u, \tau, \gamma, s, t, T)$, there is a (static) generalized flow $x'$ in $(G^T, u^T, \gamma^T, s', t')$ that sends exactly the same amount of flow to the sink, and vice versa.*

*Proof.* Given $x$, we define $x'$ by

$$
\begin{aligned}
x'_{e_\theta} &:= x_{e,\theta} && \text{for all } e_\theta \in E(G^T),\\
x'_{(v_\theta, v_{\theta+1})} &:= \sum_{e \in \delta^-(v)} \sum_{\xi=0}^{\theta-\tau_e} \gamma_e x_{e,\xi} - \sum_{e \in \delta^+(v)} \sum_{\xi=0}^{\theta} x_{e,\xi} && \text{for all } \begin{array}{l} v \in V(G) \setminus \{s\}, \\ \theta \in \{0, \dots, T-2\}, \end{array}\\
x'_{(s_\theta, s_{\theta+1})} &:= \sum_{e \in \delta^+(s)} \sum_{\xi=\theta+1}^{T-1} x_{e,\xi} && \text{for all } \theta \in \{0, \dots, T-2\}.
\end{aligned}
$$

By construction of the time-expanded network and $x$ being a generalized flow over time it follows that $x'$ satisfies capacity and flow conservation constraints. It is easy to verify that if $x$ fulfills strict flow conservation constraints, then $x'$ does as well.

Given $x'$, we define $x$ by $x_{e,\theta} := x'_{e_\theta}$ for all $e \in E(G), \theta \in \{0, \dots, T-1-\tau_e\}$. As before, it is straightforward to see that $x$ fulfills capacity and flow conservation constraints and strict flow conservation constraints if $x'$ does the same. □

As a result of this theorem, we may use generalized flows over time in $G$ and generalized static flows in $G^T$ interchangeably. We refer to [31] for an introduction to flows over time and related concepts.

It is a well-known result that a static $s$-$t$-flow can be decomposed into flow along $s$-$t$-paths and cycles. A similiar decomposition exists for generalized static flows. The following theorem is due to Gondran and Minoux [17].

**Theorem 2.** *A generalized flow $x$ in a network $(G, u, \gamma, s, t)$ can be decomposed into generalized flows $x_1, .., x_k$, $k \leq |E(G)|$ with $x = \sum_{i=1}^{k} x_i$ such that each generalized flow $x_i$ is of one of the following five types (see Figure 1):*

- *Type I: a path from the source $s$ to the sink $t$,*
- *Type II: a flow-generating cycle connected to the sink $t$ by a path (possibly of zero length),*
- *Type III: a path from the source $s$ (possibly of zero length) to a flow-absorbing cycle,*
- *Type IV: a unit-gain cycle,*
- *Type V: a flow-generating cycle connected to a flow-absorbing cycle by a path (possibly of zero length).*

Similarly, there is a generalized optimality criterion due to Onaga [25, 26].

**Theorem 3.** *A generalized flow $x$ in a graph $G$ with source $s$ and sink $t$ is a generalized maximum flow if, in the residual network $G_x$, there is neither an $s$-$t$-path along which flow can be augmented nor a flow-generating cycle connected to the sink.*

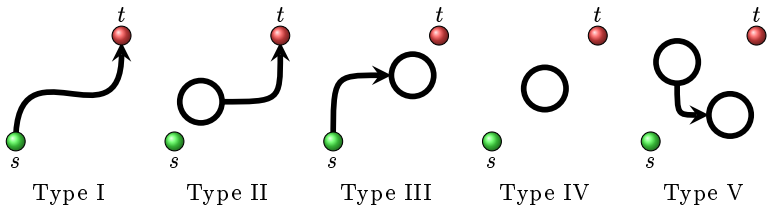


**Fig. 1.** The five types of elementary generalized flows

These results carry over to generalized flows over time, since we can treat generalized flows over time as static generalized flows in the time-expanded network and vice versa.

## 3 Complexity and Hardness of Approximation

In this section we study the problem in the general case, i. e., in the setting of arbitrary gain factors on the arcs. We begin by analyzing the computational complexity of the problem.

It is easy to see that the generalized maximum flow over time problem can be solved by using the algorithms known for the static generalized maximum flow problem on a time-expanded network. This yields pseudo-polynomial time algorithms, implying that the problem is not *strongly* NP- or PSPACE-hard, unless $P = NP$. As a lower bound for the complexity of the generalized maximum flow over time problem, we will show that there is no polynomial time approximation algorithm for it, unless $P = NP$. It is still unknown whether a strongly polynomial time algorithm exists for the static generalized maximum flow problem.

**Theorem 4.** *There is neither a polynomial algorithm nor a polynomial approximation algorithm for the generalized maximum flow over time problem, even on series-parallel graphs and proportional gains, unless $P = NP$.*

### 3.1 Proof of Theorem 4

*Proof.* Let

$$I_P = (a_1, \ldots, a_n), \quad \sum_{i=1}^{n} a_i = 2L, \quad a_1, \ldots, a_n, L \in \mathbb{N}$$

be an instance of the PARTITION problem. We define a graph

$$G = (V = \{v_0, v_1, \ldots, v_{n+1}\}, \{(v_0, v_1)\} \cup E \cup E')$$

by

$$E := \{e_1 = (v_1, v_2), e_2 = (v_2, v_3) \ldots, e_n = (v_n, v_{n+1})\}$$
$$E' := \{e'_1 = (v_1, v_2), e'_2 = (v_2, v_3) \ldots, e'_n = (v_n, v_{n+1})\}$$

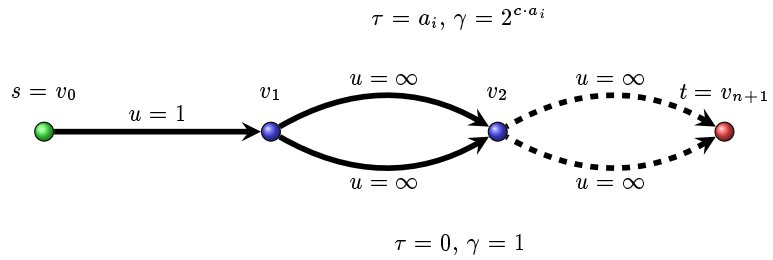


**Fig. 2.** The generalized maximum flow over time instance used in the reduction.

We set $s := v_0$, $t := v_{n+1}$, $T := L + 1$ and define

$$u_e := \begin{cases} 1 & e = (v_0, v_1), \\ \infty & \text{else}, \end{cases} \qquad \tau_e := \begin{cases} a_i & e = e_i \in E, \\ 0 & e \in \{(v_0, v_1)\} \cup E', \end{cases}$$

and gain factors $\gamma_e$ that are proportional to the transit times, i. e., $\gamma_e = 2^{c\tau_e}$ for a constant $c > 0$ that we will fix later. Note that the gain factors are encoded implicitly by the transit times; the resulting instance of the generalized maximum flow over time problem is therefore polynomial in the size of the partition instance. Figure 2 depicts this instance.

If $I_P$ is a YES-instance with a solution $I \subset \{1, 2, \ldots, n\}$, then there is a generalized flow over time that sends at least $2^{c \cdot L}$ units of flow from $s$ to $t$ within the time horizon $T$. This is, for example, achieved by sending at time 0 one unit of flow into the path induced by the edge set $\{ e_i \mid i \in I\} \cup \{ e'_i \mid i \notin I\}$. This path has length $L$ and gain factor $2^{c \cdot L}$ yielding $2^{c \cdot L}$ flow units arriving at the sink in time step $L = T - 1$. If $I_P$ is a NO-instance, then there exists no path of length $L$. Thus, due to the time horizon of $L + 1$, there exists no path with length at least $L$ that is able to carry flow to the sink in time. At most one unit of flow can leave the source at any point in time $\theta$, with $2^{c \cdot \min\{L-1, L-\theta\}}$ being the best gain factor possible for flow leaving at time $\theta$. This yields an upper bound of

$$\sum_{i=0}^{L-1} 2^{c \cdot i} + 2^{c(L-1)} = \frac{2^{c \cdot L} - 1}{2^c - 1} + 2^{c(L-1)}$$

on the amount of flow that can be sent into the sink in a NO-instance. Notice that, for $c$ sufficiently large, this amount is strictly smaller than the lower bound $2^{c \cdot L}$ on the maximum flow value for a YES-instance. The size of the gap depending on $c$ is:

$$\frac{2^{c \cdot L}}{\frac{2^{c \cdot L} - 1}{2^c - 1} + 2^{c(L-1)}} = \frac{2^{c \cdot L}(2^c - 1)}{2^{c \cdot L} - 1 + (2^c - 1)2^{c(L-1)}} = \frac{2^c - 1}{2 - \frac{1}{2^c} - \frac{1}{2^{c \cdot L}}} \geq \frac{2^c - 1}{2} \quad .$$

Thus, we can make the gap arbitrarily large by choosing $c$ appropriately. Therefore any polynomial time approximation algorithm for the generalized maximum flow over time problem is able to solve the $NP$-hard PARTITION problem. □

A similar result can be shown for lossy networks, i. e., networks where all gain factors are $\leq 1$. This requires non-proportional gain factors and transit times, however.

Note that Theorem 4 still holds if we encode a number $n$ as $n = m \cdot 2^e$ and use binary encoding for $m$ and $e$. This is common when using floating-point numbers, for instance. This allows us to encode a number $n$ that is a power of 2 in space $O(\log \log n)$; it is easy to check that the hardness proof still holds in this case. It is an interesting open question whether the theorem holds when a number $n$ in the input has to be encoded in space $O(\log n)$. We will see in Section 5.3 that arrival patterns of optimal solutions to the generalized maximum flow over time problem bear resemblance to earliest arrival flows – another flow over time problem of unclear complexity which dates back to Gale [13] who referred to it as *universally maximum dynamic flow problem.*

On very restricted classes of graphs like, e. g., shortest-paths networks, it is possible to deal with the generalized maximum flow over time problem efficiently, along the lines of the work of Hall, Hippler, and Skutella [18].

## 4 Lossy Networks

In this section, we consider the special case of $\gamma_e \leq 1$, for all arcs $e \in E(G)$. This means that flow is only lost, but never gained along arcs. We refer to such networks as *lossy networks.* It is well-known that any network without flow-generating cycles can be turned into a lossy network by node-dependent scaling of flow values. Thus, the results discussed in this section hold for all networks without flow-generating cycles.

Approximating the maximum flow value is hard in general, as we have seen in the last section. This result even carries over to lossy networks if the reduction given in the proof of Theorem 4 is modified accordingly[2]. Therefore, we now focus on relaxing the feasibility, i. e., given some $\alpha > 1$ and a problem instance $I$ with a time horizon $T$, we ask for a feasible solution to $I$ with time horizon $\alpha \cdot T$ whose value is at least that of an optimal solution to $I$ with time horizon $T$. We can use the concept of *condensed time-expanded networks* from Fleischer and Skutella [6] to show the following theorem.

**Theorem 5.** *Let $OPT$ be the value of an optimal solution to a generalized maximum flow over time problem instance $I = (G, u, \tau, \gamma, s, t, T)$ on a lossy network. For any $\epsilon > 0$, there is an algorithm with running time polynomial in the input size and $1/\epsilon$ that computes a solution of value at least $OPT$ for the problem instance $I' = (G, u, \tau, \gamma, s, t, (1 + \epsilon) \cdot T)$.*

The proof of this theorem follows along the lines of Fleischer and Skutella's proof for minimum cost flows over time [6]. The main idea is to round up transit times of arcs to multiples of some large number $\Delta$ and then to construct a

[2] Instead of rewarding the use of the positive length arcs by exponentially large gains, we punish the use of zero length arcs by exponentially small gains.

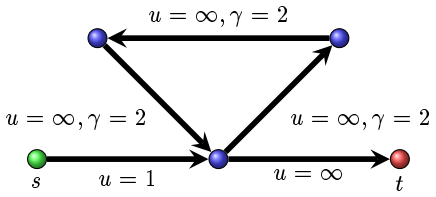


**Fig. 3.** An instance where flow visits nodes multiple times.

*condensed time-expanded network* containing roughly $T/\Delta$ time layers only. A suitable choice of $\Delta$ is $\varepsilon^2 T/n$ resulting in a condensed time-expanded network of size polynomial in the input size and $1/\epsilon$.

It can be shown that one can make up for the imprecision due to the rounding of transit times by simply increasing the time horizon $T$ by a factor $1 + O(\epsilon)$. One essential insight of the proof is that the rounding of transit times delays flow particles by no more than $n \cdot \Delta = \epsilon^2 T$ since we can restrict to solutions with only simple flow paths, i. e., no node is visited more than once. For the case of lossy networks, this approach works for the generalized maximum flow problem as well since we can restrict ourselves to simple paths, which is crucial for applying this technique.

Unfortunately, it is not clear whether Theorem 5 can be generalized to the case of arbitrary gain factors on the arcs. The main problem is that the assumption of simple flow paths no longer holds in the general setting. Taking a closer look at the analysis in the proof of Theorem 5, it suffices to bound the number of times a flow particle visits a node by a polynomial in the input size. But even this relaxed condition does in general not hold for arbitrary gain factors, as can be seen from the example depicted in Figure 3. We assume unit transit times and gains, unless specified otherwise. Then in a generalized maximum flow over time each flow particle has to use the cycle as many times as possible to generate as much flow as possible behind the bottleneck edge leaving $s$. Note, however, that this cycle cannot be used to generate flow from scratch as its transit time is not zero – i. e., we can amplify flow by spending time in the cycle, but we cannot generate flow from scratch.

## 5 Proportional Losses

In this section, we consider the special case of $\gamma \equiv 2^{c \cdot \tau}$, for some constant $c < 0$. This means that in each time unit the same percentage of the remaining flow value is lost. This is motivated by problems where goods cannot be transported reliably, e. g., due to leakage or evaporation. In many applications, this loss crucially depends on the time spent in the transportation network as many processes of growth or decay in nature evolve over time according to an exponential function.

In Section 5.1 we show that the maximum generalized flow over time problem can be solved on the static network by a variant of the Successive Shortest Path Algorithm. This is particularly remarkable as so far only the most basic maximum flow over time problem and the closely related earliest arrival flow problem were known to be solvable to optimality by static flow computations on the static network (i. e., not requiring the use of time-expanded networks).

In Section 5.2 we show how this algorithm can be turned into an FPTAS which is considerably more efficient and uses much less space than the more general FPTAS based on condensed time-expanded networks discussed in Section 4. Furthermore, our FPTAS approximates the flow value instead of the time horizon like Fleischer and Skutella's FPTAS. That is, we approximate optimality instead of feasibility.

Finally, in Section 5.3 we elaborate on an interesting uniqueness property of maximum generalized flows.

### 5.1 A Variant of the Successive Shortest Path Algorithm

Due to the work of Onaga [25, 26] it is known that augmenting flow successively along highest gain $s$-$t$-paths solves the generalized maximum $s$-$t$-flow problem. More precisely, Onaga's algorithm for lossy networks proceeds as follows. Begin with the zero-flow and the corresponding residual network. If no source-sink path exists in this residual network, terminate. Otherwise, augment flow along a source-sink path of maximum gain and continue with the resulting flow and residual network. Thus, applying Onaga's algorithm in the time expanded network solves the generalized maximum flow over time problem – at the cost of potentially requiring pseudo-polynomially many augmentations in the pseudo-polynomially large time expanded network.

We will now present an algorithm capable of solving the special case described above using only the original – not time expanded – network. The idea of this algorithm is to employ a strategy similar to Onaga's in the original network and use this as a foundation to construct a flow over time solving the special case.

We begin by introducing some notations; more precisely, we introduce a slightly non-standard way of building a time-expanded network from copies of the original network. Let $(G, u, \tau, \gamma, s, t, T)$ be a (residual) network, let $\gamma_{v\to w}$ be the maximum gain of a $v$-$w$-path in $G$ and $\tau_{v\to w}$ the length of a shortest $v$-$w$-path (with respect to transit times) in $G$. Notice that $\tau_{v\to w} := \frac{1}{c}\log\gamma_{v\to w}$. Initially, we introduce our construction for the special case of unique gain networks (i. e., a network where all paths from a node $v \in V(G)$ to a node $w \in V(G)$ have the same gain) only; this special case has the advantage of allowing for a simpler and more concise definition. The *$\theta$-copy* $\theta G$ of $G$ for $\theta \in \{0, \dots, T - \tau_{s\to t} - 1\}$ is then:

$$V(\theta G) := \left\{ v_\xi \in V(G^T) \;\middle|\; v \in V(G), \xi = \theta + \tau_{s\to v} \right\}$$
$$E(\theta G) := \left\{ e_\xi \in E(G^T) \;\middle|\; e = (v, w) \in E(G), \xi = \theta + \tau_{s\to v} \right\}$$

More generally, we define *$\theta$-copy* $\theta G$ of $G$ for some $\theta \in \{0, \dots, T - \tau_{s\to t} - 1\}$ to be the following subgraph of the time expanded network $G^T$:

$$E(\theta G) := \left\{ e_\xi \in E(G^T) \;\middle|\; e = (v,w) \in E(G), \xi = \theta + \tau_{s\to v}, \xi + \tau_e + \tau_{w\to t} < T \right\}$$
$$V(\theta G) := \bigcup_{e=(v,w)\in E(\theta G)} \{v, w\}.$$

For the special case mentioned above these two definitions coincide. Similarly, we define the *$[\theta,\theta']$-copies* $[\theta,\theta']G$ of $G$, $0 \le \theta < \theta' \le T - \tau_{s\to t} - 1$ as

$$V([\theta,\theta']G) := \bigcup_{\xi=\theta}^{\theta'} V(\xi G),$$
$$E([\theta,\theta']G) := \bigcup_{\xi=\theta}^{\theta'} E(\xi G) \cup \bigcup_{v\in V(G)} \bigcup_{\xi=\theta+\tau_{s\to v}}^{\theta'+\tau_{s\to v}-1} \{(v_\xi, v_{\xi+1})\}.$$

For brevity, we also define $\overline{G} := [0, T - \tau_{s\to t} - 1]G$ if $\tau_{s\to t} < T - 1$ and as the empty graph otherwise. Note that $\overline{G}$ is the subnetwork of $G^T$ containing exactly the nodes and edges of $G^T$ that can be part of $s'$-$t'$-paths (with $s'$, $t'$ being the source and sink of the time-expanded network, see Definition 2). For our purposes, it is clearly sufficient to work with $\overline{G}$ instead of $G^T$.

Furthermore, if we consider an $s'$-$t'$-flow $f$ in a time-expanded network $G^T$, it can happen that flow is sent through holdover edges at source and sink. In this case, the residual network $G^T_f$ corresponding to such a flow $f$ can have reverse holdover edges at source and sink. These reverse holdover edges do not help to construct new $s'$-$t'$-paths in the time-expanded-network or new flow-generating cycles with a path to $t'$. By Theorem 2 it follows that they can be omitted as well. We write $\widetilde{G^T_f}$ for the subnetwork of $G^T_f$ created by removing nodes and edges not on $s'$-$t'$-paths and reverse holdover edges at source and sink. Figures 4, 5, and 6 show a network, its time-expansion as well as selected $\theta$- and $[\theta,\theta']$-copies.

Analogously, we define for a flow $x$ in such a unique gain network $G$ the *$\theta$-flow* $\theta x$ of $x$ in $\theta G$ for some $\theta \in \{0, \dots, T - \tau_{s\to t} - 1\}$ by $(\theta x)_{e_\xi} := x_e$ for all $e_\xi \in E(\theta G)$. Furthermore, we define the *$[\theta,\theta']$-flow* $[\theta,\theta']x$ of $x$ in $[\theta,\theta']G$ for some $\theta, \theta' \in \{0, \dots, T - \tau_{s\to t} - 1\}$ with $\theta < \theta'$ by setting for all $e \in E([\theta,\theta']G)$:

$$([\theta,\theta']x)_e := \begin{cases} 0 & e = (v_\xi, v_{\xi+1}), v \in V(G) \setminus \{s,t\}, \\ (\theta' - \theta + 1)|x| & e = (s_\xi, s_{\xi+1}), \xi < \theta, \\ (\theta' - \xi)|x| & e = (s_\xi, s_{\xi+1}), \theta \le \xi < \theta', \\ 0 & e = (s_\xi, s_{\xi+1}), \theta' \le \xi, \\ 0 & e = (t_\xi, t_{\xi+1}), \xi \le \theta + \tau_{s\to t}, \\ (\xi - \theta - \tau_{s\to t} + 1)|x| & e = (t_\xi, t_{\xi+1}), \theta + \tau_{s\to t} < \xi < \theta' + \tau_{s\to t}, \\ (\theta' - \theta + 1)|x| & e = (t_\xi, t_{\xi+1}), \xi \ge \theta' + \tau_{s\to t}, \\ x_e & e = e_\xi. \end{cases}$$

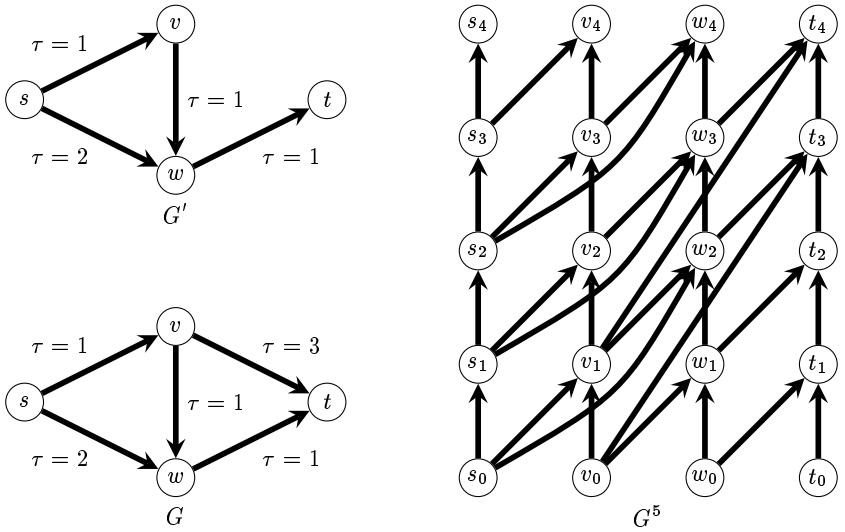


**Fig. 4.** A network $G$, its highest-gain network $G'$ and its time-expanded network $G^5$ (note that gains are defined implicitly by the transit times)

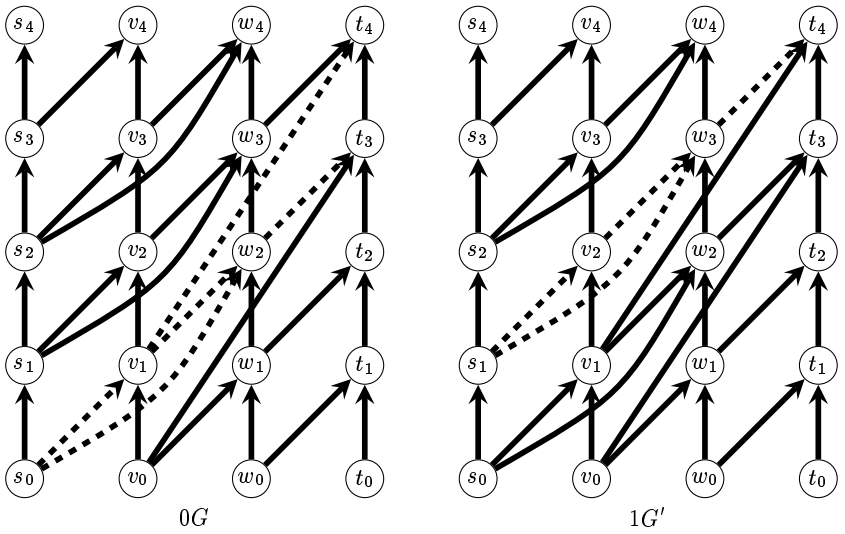


**Fig. 5.** $0G$ and $1G'$ (dashed) as subnetworks of $G^5$

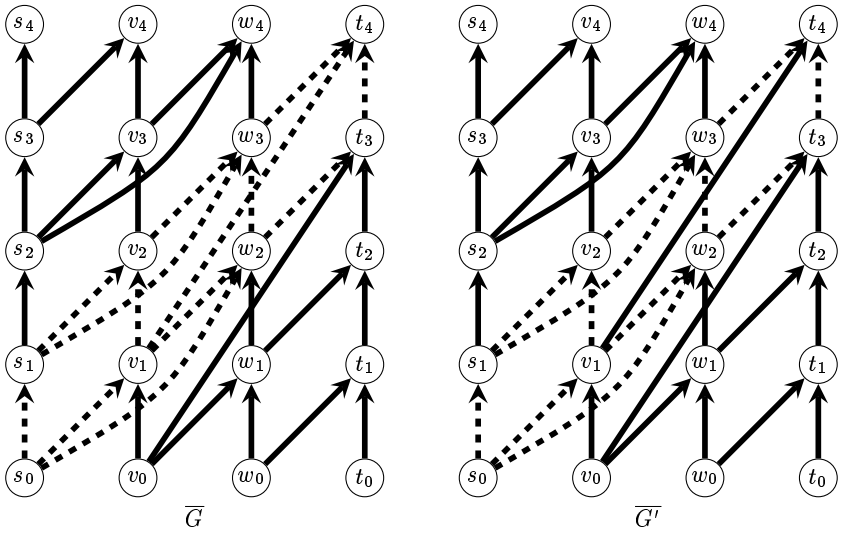


**Fig. 6.** $\overline{G}$ and $\overline{G'}$ (dashed) as subnetworks of $G^5$

Again, we define for brevity $\overline{x} := [0, T-\tau_{s\to t}-1]x$ for a flow $x$ in $G$, if $\tau_{s\to t} < T-1$ and as the zero flow otherwise. Informally spoken, the idea of our algorithm is to start with the zero flow, compute a maximum-flow in the highest-gain / shortest-path subnetwork of the static residual network, augment this flow and repeat this process until no $s$-$t$-path exists in the static residual network. We then use the augmented maximum-flows to construct an optimal solution to our problem, by sending each flow as long as possible into the network (i. e., temporally repeated). We will use the notations introduced above to describe the construction of the flow over time in the last step of our algorithm, and show its optimality.

**Algorithm 1** *Let $I = (G, u, \tau, \gamma, s, t, T)$ be an instance of the generalized maximum flow over time problem.*

1. *Begin with $i := 0$ and the static zero-flow $x_0 :\equiv 0$.*
2. *If no $s$-$t$-path exists in $G_{x_i}$ or if $\tau_{s\to t} \geq T$ in $G_{x_i}$, then set $k := i-1$ and go to step 6.*
3. *Restrict the static residual network $G_{x_i}$ to the network $G'_{x_i}$ containing only paths of maximum gain and compute a generalized maximum flow $x'_i$ in $G'_{x_i}$.*
4. *Define $x_{i+1}$ by adding $x'_i$ to $x_i$ as follows: $(x_{i+1})_e := (x_i)_e + (x'_i)_e + \gamma_e^{-1}(x'_i)_{\overleftarrow{e}}$ for all $e \in E(G)$. Notice that $x_{i+1}$ is a feasible flow in $G$, since $x'_i$ is a feasible flow in a restricted residual network of $x_i$.*
5. *Set $i := i+1$ and go to step 2.*
6. *Construct a generalized flow over time $f$ defined by $f = \sum_{j=0}^{k} \overline{x'_j}$.*

We will prove the correctness of Algorithm 1 by comparing it to Onaga's algorithm applied to the time expanded network. For $i = 0, \ldots, k+1$, define a

generalized flow over time $f_i := \sum_{j=0}^{i-1} \overline{x'_j}$. In particular, $f_0$ is the zero flow over time and $f = f_{k+1}$. The strategy for our proof is to show that in every iteration $\overline{x'_i}$ is a flow along highest gain paths in $G_{f_i}^T$. Then successively adding $\overline{x'_0}, \ldots, \overline{x'_k}$ produces the same result as Onaga's algorithm applied to the time expanded network, showing correctness of our algorithm. The following claim turns out to be helpful in proving the correctness of Algorithm 1.

*Claim.* For each $i = 0, \ldots, k+1$, it holds that $\overline{G_{x_i}} = \widetilde{G_{f_i}^T}$, i. e., the copied static network is equal to the pruned time-expanded network after each iteration of the algorithm.

*Proof.* The proof uses induction on $i$. For $i = 0$, we get $\overline{G_{x_0}} = \overline{G} = \widetilde{G^T} = \widetilde{G_{f_0}^T}$. Now we assume that $\overline{G_{x_i}} = \widetilde{G_{f_i}^T}$ holds for some $i$. In iteration $i$, the algorithm adds $\overline{x'_i}$ to $f_i$ and $x'_i$ to $x_i$, resulting in $f_{i+1}$ and $x_{i+1}$, respectively. Due to $x'_i$ being a maximum flow in the highest gain network of $G_{x_i}$ it follows that $\overline{x'_i}$ is a maximum flow in the highest gain network of $\overline{G_{x_i}}$.

Sending $x'_i$ in $G_{x_i}$ causes all copies $\theta G_{x_i}$ of $\overline{G_{x_i}}$ to be changed as well. These changes due to sending flow are the same that sending $\overline{x'_i}$ in $\overline{G_{x_i}}$ causes. In contrast to $\overline{G_{x_i}}$, $\overline{G_{x_{i+1}}}$ might also gain or lose arcs that either become again part of an $s'$-$t'$-path or are no longer part of an $s'$-$t'$-path. However, note that both $x'_i$ and $\overline{x'_i}$ are flows in the highest gain network of $G_{x_i}$ and $\overline{G_{x_i}}$, respectively. This implies that sending these flows does not increase the gain of source-node- or node-sink-paths. By proportionality of gains and transit times it follows that the transit time of source-node- or node-sink-paths does not decrease; thus, no arcs are gained this way. Similiarly, all arcs lost due to no longer being on an $s'$-$t'$-path in $\overline{G_{x_{i+1}}}$ are by definition also removed in $\widetilde{G_{f_{i+1}}^T}$. Thus, $\overline{G_{x_{i+1}}} = \widetilde{G_{f_{i+1}}^T}$. This concludes the proof. □

**Theorem 6.** *Algorithm 1 computes a generalized maximum flow over time.*

*Proof.* Making use of Claim 5.1 we show that, for $i = k+1$ (i. e., after the final iteration), there is no $s'$-$t'$-path in $G_f^T$. Since the algorithm terminates, there is no $s$-$t$-path in $G_{x_i}$ and therefore no $s'$-$t'$-path in $\overline{G_{x_i}} = \widetilde{G_f^T}$. As $G_f^T$ has no more $s'$-$t'$-paths then $\widetilde{G_f^T}$ the result follows. Note that our instance has no flow-generating cycles to begin with, due to $c < 0$, and sending flow along highest gain paths does not generate new flow-generating cycles (see Onaga [25, 26]).

Thus, our algorithm starts with the zero flow over time $f_0$, similar to Onaga's algorithm. In every iteration, flow is only being augmented along highest gain paths in $\overline{G_{x_i}}$ and by Claim 5.1 this equals $\widetilde{G_{f_i}^T}$ as well. Since this is exactly what Onaga's algorithm does, the correctness of our algorithm follows. Note that Claim 5.1 also guarantees that $f$ is a feasible generalized flow over time, due to each $\overline{x'_i}$ being a flow in $\overline{G'_{x_i}}$ which is a restricted residual network of $\widetilde{G_{f_i}^T}$. This concludes the proof. □

We conclude this section by examining the running time of Algorithm 1. Let $n := |V(G)|$, $m := |E(G)|$ and $U := \max_{e \in E(G)} u_e$ for a problem instance $(G, u, \tau, \gamma, s, t, T)$. In our case, a highest gain path can be found in $O(nm)$ time using Moore-Bellman-Ford's algorithm (see Bellman [2], Ford [9], Moore [24]) or in $O(m + n \log n)$ by applying Dijkstra's algorithm [5] with Fibonacci heaps (see Fredman and Tarjan [12]) and reduced costs. For both algorithms, $\tau_e = \frac{1}{c} \log \gamma_e$ is being used as a cost function. Both algorithms are capable of computing the highest-gain network as well. A generalized maximum flow in the highest-gain network can then be computed by a standard maximum flow algorithm. Since there are at most $T$ time steps, there can be at most $T$ iterations. The running time of an iteration is dominated by the maximum flow computation, yielding a running time of $O(maxflow \cdot T)$, where $O(maxflow)$ is the running time of the maximum flow algorithm. King, Rao, and Tarjan[21] describe a maximum flow algorithm with a running time of $O(nm \log_{m/(n \log n)} n)$, resulting in a running time of $O(nm \log_{m/(n \log n)} n \cdot T)$ for our algorithm.

For special cases, this runtime can be improved further. Beygang, Krumke, and Zeck [3] recently studied static generalized maximum flows in series-parallel networks and discovered that a greedy-strategy that chooses always the highest-gain path in the original – not residual – network is sufficient for finding an optimal solution. This can be carried over to our setting and can be used for bounding the number of paths used. Since each augmentation saturates an arc, there can be at most $m$ iterations, yielding a polynomial time algorithm.

### 5.2 Turning the Algorithm into an FPTAS

In this section, we will see that Algorithm 1 can be terminated early to obtain an approximate solution. In fact, the algorithm can be turned into an FPTAS.

**Theorem 7.** *Let $OPT$ be the value of an optimal solution to a generalized maximum flow over time problem instance $I = (G, u, \tau, \gamma, s, t, T)$, $\varepsilon > 0$, and $U := \max_{e \in E} u_e$. Algorithm 1 has found a solution of value at least $OPT - \varepsilon$ after all paths of length $\leq \tau^*$ with*

$$\tau^* := -\tfrac{1}{c}(\log \tfrac{1}{\varepsilon} + \log m + \log U + 2 \log T)$$

*have been processed (recall that the lengths of the paths used by the algorithm are monotonically increasing). For a constant $c < 0$ and using a maximum flow based approach as proposed in Section 5.1, this leads to a running time of $O(maxflow \cdot (\log \varepsilon^{-1} + \log U + \log T))$ for a solution of value at least $OPT - \varepsilon$.*

*Proof.* Algorithm 1 computes an optimal solution for the generalized maximum flow over time problem. For $i \in \{0, \ldots, T-1\}$, let $a_i$ denote the amount of flow sent into paths of length (transit time) $i$ in a time step $\theta \in \{0, \ldots, T-i-1\}$ by the algorithm. Note that the algorithm does not use holdover; the gain factor of a path of length $i$ is therefore $2^{c \cdot i}$ and flow is sent into a path of length $i$ for $T - i$ time steps. The length of the paths used by Algorithm 1 increases monotonically.

After all paths of length $\leq i$ have been found, only paths of length $i+1, \ldots, T-1$ remain. The amount of flow arriving at the sink by such paths equals

$$\sum_{j=i+1}^{T-1} a_j \cdot 2^{c \cdot j} \cdot (T-j) \ .$$

Due to $c < 0$ it follows that $\max\left\{2^{c(i+1)}, \ldots, 2^{c(T-1)}\right\} = 2^{c(i+1)}$. Furthermore, it is clear that $T - j \leq T$. The amount of flow sent into paths in one time step is bounded by $m \cdot U$. This yields

$$\sum_{j=i+1}^{T-1} a_j \cdot 2^{c \cdot j} \cdot (T-j) \leq \sum_{j=i+1}^{T-1} m \cdot U \cdot 2^{c(i+1)} \cdot T \leq m \cdot U \cdot 2^{c(i+1)} \cdot T^2 \ .$$

For $i = -\frac{1}{c}(\log \frac{1}{\varepsilon} + \log m + \log U + 2 \log T) - 1$, the right hand side equals $\varepsilon$. □

The above theorem allows an approximation within a constant value $\varepsilon$. For an FPTAS, we need to approximate $OPT$ within a factor of $(1-\varepsilon)$ or a value of $\varepsilon OPT$. This can be done by a slight modification of Theorem 7.

**Theorem 8.** *Let $OPT$ be the value of an optimal solution for a generalized maximum flow over time problem instance $I = (G, u, \tau, \gamma, s, t, T)$, $\varepsilon > 0$ and $U := \max_{e \in E} u_e$. Algorithm 1 has found a solution of value at least $(1-\varepsilon)OPT$ after $\tau^*$ iterations with*

$$\tau^* := \lceil -\tfrac{1}{c}(\log \tfrac{1}{\varepsilon} + \log m + \log U + 2 \log T) \rceil$$

*For a constant $c < 0$ and using a maximum flow based approach as proposed in Section 5.1, this leads to a running time of $O(maxflow \cdot (\log \varepsilon^{-1} + \log U + \log T))$ for a solution of value at least $(1-\varepsilon)OPT$.*

*Proof.* We use the arguments and notations of the proof of Theorem 7. Let $i_0$ be the length of the first iteration's paths. We assume w.l.o.g that the minimum capacity in the network is 1. Consequently, at least $2^{ci_0}$ flow is sent in the first iteration. We now want to determine $\tau^*$ such that the amount of flow sent to the sink by later iterations is $\leq \varepsilon OPT$.

$$\sum_{j=i_0+\tau^*}^{T-1} a_j \cdot 2^{c \cdot j} \cdot (T-j) \leq m \cdot U \cdot 2^{c(i_0+\tau^*)} \cdot T^2 \leq \varepsilon 2^{ci_0} \leq \varepsilon OPT$$

This inequality can be transformed to

$$m \cdot U \cdot 2^{c\tau^*} \cdot T^2 \leq \varepsilon \quad \Leftrightarrow \quad \tau^* \geq -\tfrac{1}{c}(\log \tfrac{1}{\varepsilon} + \log m + \log U + 2 \log T)$$

This concludes the proof. □

### 5.3 Arrival Pattern and Connections to Earliest Arrival Flows

In Section 5.1, we have seen that there is always an optimal solution that uses no holdover at any node except for $s$ and $t$. In this section, we will show that all such optimal solutions share the same arrival pattern.

**Theorem 9.** *Let $I = (G, u, \tau, \gamma, s, t, T)$ be an instance of the generalized maximum flow over time problem such that $\gamma \equiv 2^{c\tau}$ for some constant $c < 0$. Let $f$, $f'$ be generalized maximum flows over time for $I$ that do not use holdover in any node except for $s$ and $t$. Then $f$ and $f'$ have the same arrival pattern.*

*Proof.* Assume that $f$ and $f'$ fulfill the requirements of the theorem but have different arrival patterns. Consider the time-expanded network $G^T$ and treat $f$ and $f'$ as generalized static flows in $G^T$. We can define $f - f'$ to be a generalized flow in the residual network $G^T_{f'}$ of $f'$ as follows:

$$(f - f')_e := \begin{cases} \max\{f_e - f'_e, 0\} & e \in E(G^T), \\ \max\left\{(f'_{\overleftarrow{e}} - f_{\overleftarrow{e}}) \cdot \gamma_{\overleftarrow{e}}, 0\right\} & \overleftarrow{e} \in E(G^T), \end{cases} \quad \text{for all } e \in E(G^T_{f'}).$$

Generalized flows can be decomposed into five types of elementary generalized flows according to Theorem 2. Such an elementary flow $g$ in $G^T_{f'}$ can be added to $f'$ in $G^T$ in the following way:

$$(f' + g)_e := f'_e + g_e - g_{\overleftarrow{e}}\gamma_{\overleftarrow{e}} \quad \text{for all } e \in E(G^T).$$

Thus we can conclude that the decomposition does not contain Type I or II elementary flows since they could be added to $f'$ increasing the total flow sent to the sink $t'$, which would yield a contradiction to the maximality of $f'$. Similarly, the decomposition cannot contain Type III or V flows routing flow through a copy $t_\xi$ of sink $t$ since such a flow would enable a Type I or II flow. All other kinds of Type III or V flows can obviously not affect the arrival pattern when added to $f'$. Due to our choice of $f$ and $f'$, one of these flows needs to change the arrival pattern, however. This leaves Type IV flows as the only possibility left. Once again, we can confine ourselves to Type IV flows sending flow along holdover or reverse holdover edges at $t$ (if no form of holdover is used, the arrival pattern does not change due to flow conservation). Notice that such a Type IV flow does not use holdover at $s$ since it would otherwise enable a Type I flow.

We now partition the edges used by a Type IV flow into the set of all used holdover edges $H$, the set of all used reverse holdover edges $R$ and the set of all other edges $O$. Type IV flows send flow along a unit-gain-cycle $C$. This leads to:

$$1 = \gamma_C = \underbrace{\prod_{e \in H} \gamma_e}_{=1} \cdot \underbrace{\prod_{e \in R} \gamma_e}_{=1} \cdot \prod_{e \in O} \gamma_e \quad \Longrightarrow \quad \prod_{e \in O} \gamma_e = 1$$

Each edge $e \in O$ corresponds to an edge at a certain point in time, each edge $e \in H$ corresponds to waiting one time unit at a node and each edge $e \in R$

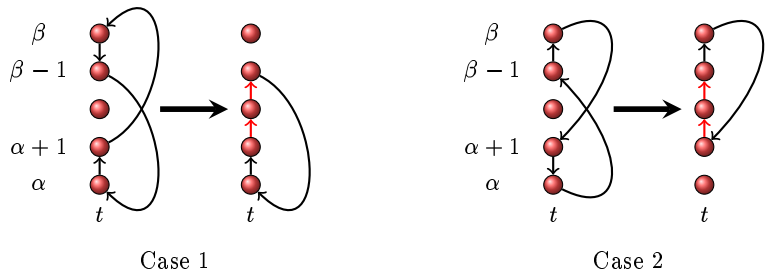


**Fig. 7.** Rerouting to construct flow-generating cycles

corresponds to canceling waiting time at a node for one time unit. The unit-gain cycle $C$ itself corresponds to a cycle with transit time 0, yielding

$$0 = \tau_C = |H| - |R| + \sum_{e \in O} \tau_e \ .$$

Proportionality of transit times and gain factors give us

$$1 = \prod_{e \in O} \gamma_e = \prod_{e \in O} 2^{c\tau_e} = 2^{c \sum_{e \in O} \tau_e} \quad \Longrightarrow \quad \sum_{e \in O} \tau_e = 0 \quad \Longrightarrow \quad |H| = |R| \ .$$

Thus we know now that Type IV flows using holdover / reverse holdover at $t$ must use an equal number of holdover and reverse holdover arcs. We will complete the proof by showing that a type IV flow using an equal number of holdover and reverse holdover arcs at $t$ and none at $s$ allows us to reroute flow along a flow-generating cycle in contradiction to the maximality of $f'$. Figure 7 depicts this rerouting.

Let $f^*$ be such a type IV flow. By assumption, there is a holdover arc used by $f^*$ to let flow wait in $t$ from time layer $\alpha$ to $\alpha + 1$ and a reverse holdover arc used by $f^*$ to cancel waiting in $t$ from time layer $\beta$ to $\beta - 1$. There are two possible cases, depending on which happens earlier in time. We will refer to the case where $\alpha < \beta$ as Case 1, the other as Case 2. Figure 7 shows these two cases. Due to the fact that holdover arcs at the sink have unlimited capacity, we can reroute the flow by letting the flow wait at $t$ from $\alpha$ to $\beta - 1$ in the first case and $\alpha + 1$ to $\beta$ in the second case. The resulting cycles are flow-generating, since they use holdover but no reverse holdover; the rest of the cycle must therefore have negative transit time, which implies flow-generation by proportionality of transit times and gain factors. This shows the contradiction and completes the proof. □

If we consider a slightly different model of flow-conservation, where flow can only be safely stored at source and sink and is subject to the same proportional loss when waiting in a node that occurs when traversing arcs, we can give a stronger version of the previous theorem.

**Theorem 10.** *Let $I = (G, u, \tau, \gamma, s, t, T)$ be an instance of the generalized maximum flow over time problem such that $\gamma \equiv 2^{c\tau}$ for some constant $c < 0$ and holdover at a node $v \in V(G) \setminus \{s, t\}$ has a loss factor of $2^c$ for each time unit spent waiting. Let $f$, $f'$ be generalized maximum flows over time for $I$. Then $f$ and $f'$ have the same arrival pattern.*

*Proof.* The arguments of the previous proof hold here as well, if we make a single modification: instead of partioning the arcs of the Type IV flow into holdover, reverse holdover and normal arcs we partition the arcs now into holdover arcs at the sink, reverse holdover at the sink and all other arcs (i.e. normal arcs and holdover arcs at intermediate nodes). This is due to the fact, that holdover arcs in nodes other than source and sink are now allowed, in contrast to the previous theorem. These holdover arcs behave the same as normal arcs in terms of proportional loss and are therefore grouped with them; the following arguments of the previous proof show then this claim as well. □

## 6 Conclusion

We have introduced the generalized maximum flow over time problem that, for the first time, combines important features captured by flows over time and generalized flows in one network flow model. While the generalized maximum flow over time problem cannot be approximated in polynomial time, unless P=NP, we have presented an efficient FPTAS for the special case of lossy networks with proportional gain factors. The algorithm relies on the successive shortest paths algorithm and reveals an interesting connection to earliest arrival flows.

The generalized flow over time model presented in this paper raises numerous interesting questions and directions for future research. The most natural generalizations of the considered network flow problem seem to be generalized minimum cost flows over time and generalized multicommodity flows over time. An interesting approach to these flow problems is the concept of *condensed time-expanded networks* introduced by Fleischer and Skutella [6]. However, as mentioned in Section 4, the analysis of these condensed time-expanded networks crucially relies on the assumption that, in an optimum solution, flow particles travel along simple paths from the source to the sink. This assumption, however, is no longer valid for generalized flows over time in networks containing flow-generating cycles. The same holds for multi-commodity flows over time without holdover at intermediate nodes. With respect to practical applications, it is an important open problem and a big theoretical challenge to make condensed time-expanded networks usable and, in particular, analyzable for such flow over time problems.

*Acknowledgements.* The authors wish to thank the anonymous referees whose valuable comments helped to improve the presentation of the paper.

## References


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

[2] R. E. Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16:87–90, 1958.

[3] K. Beygang, S. O. Krumke, and C. Zeck. Generalized max flow in series-parallel graphs. Report in Wirtschaftsmathematik 125, TU Kaiserslautern, 2010.

[4] G. B. Dantzig. *Linear programming and extensions.* Princeton University Press, 1962.

[5] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

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

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

[8] L. K. Fleischer and K. D. Wayne. Fast and simple approximation schemes for generalized flow. *Mathematical Programming*, 91:215–238, 2002.

[9] L. R. Ford. Network flow theory. Paper P-923, The Rand Corporation, 1956.

[10] L. R. Ford and D. R. Fulkerson. *Flows in Networks.* Princeton University Press, 1962.

[11] L. R. Ford and D. R. Fulkerson. Constructing maximal dynamic flows from static flows. *Operations Research*, 6:419–433, 1987.

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

[13] D. Gale. Transient flows in networks. *Michigan Mathematical Journal*, 6:59–63, 1959.

[14] A. V. Goldberg, S. A. Plotkin, and É. Tardos. Combinatorial algorithms for the generalized circulation problem. *Mathematics of Operations Research*, 16:351–379, 1991.

[15] D. Goldfarb and Z. Jin. A faster combinatorial algorithm for the generalized circulation problem. *Mathematics of Operations Research*, 21:529–539, 1996.

[16] D. Goldfarb, Z. Jin, and J. B. Orlin. Polynomial-time highest gain augmenting path algorithms for the generalized circulation problem. *Mathematics of Operations Research*, 22:793–802, 1997.

[17] M. Gondran and M. Minoux. *Graphs and Algorithms.* Wiley, 1984.

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

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

[20] L. V. Kantorovich. Mathematical methods of organizing and planning production. Technical report, Publication House of the Leningrad State University, 1939. Translated in Management Science, 6:366–422, 1960.

[21] V. King, S. Rao, and R. Tarjan. A faster deterministic maximum flow algorithm. *Journal of Algorithms*, 17:447–474, 1994.

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

[23] E. Minieka. Maximal, lexicographic, and dynamic network flows. *Operations Research*, 21:517–527, 1973.

[24] E. F. Moore. The shortest path through a maze. In *Proceedings of the International Symposium on Switching, Part II*, pages 285–292. Harvard University Press, 1959.

[25] K. Onaga. Dynamic programming of optimum flows in lossy communication nets. *IEEE Transactions on Circuit Theory*, 13:282–287, 1966.

[26] K. Onaga. Optimal flows in general communication networks. *Journal of the Franklin Institute*, 283:308–327, 1967.

[27] W. B. Powell, P. Jaillet, and A. Odoni. Stochastic and dynamic networks and routing. In M. O. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser, editors, *Network Routing*, volume 8 of *Handbooks in Operations Research and Management Science*, chapter 3, pages 141–295. North–Holland, Amsterdam, The Netherlands, 1995.

[28] T. Radzik. Faster algorithms for the generalized network flow problem. *Mathematics of Operations Research*, 23:69–100, 1998.

[29] T. Radzik. Improving time bounds on maximum generalised flow computations by contracting the network. *Theoretical Computer Science*, 312:75–94, 2004.

[30] M. Restrepo and D. P. Williamson. A simple gap-canceling algorithm for the generalized maximum flow problem. *Mathematical Programming*, 118:47–74, 2009.

[31] M. Skutella. An introduction to network flows over time. In *Research Trends in Combinatorial Optimization*, pages 451–482. Springer, 2009.

[32] K. Truemper. On max flows with gains and pure min-cost flows. *SIAM Journal on Applied Mathematics*, 32:450–456, 1977.

[33] K. D. Wayne. *Generalized Maximum Flow Algorithms*. PhD thesis, Cornell University, 1999.

[34] K. D. Wayne. A polynomial combinatorial algorithm for generalized minimum cost flow. *Mathematics of Operations Research*, 27:445–459, 2002.

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

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