Efficient Models for Special Types of Non-Linear Maximum Flow Problems

Marina Tvorogova
Department of Computer Science
TU Braunschweig, Germany
Email: m.tvorogova@tu-bs.de

***Abstract*—In this paper, we consider the maximum flow problem on networks with non-linear transfer functions. We consider special types of transfer functions, which are particularly relevant for applications. For concave transfer functions, we reduce the NL-flow problem to the generalized flow problem and solve it using a polynomial-time approximation scheme. For convex, s-shaped and monotonically growing piecewise linear (PWL) transfer functions (the latter can always be divided into s-shaped fragments), we present an equivalent network representation that allows us to build a MILP model with a better performance than if we were using standard MILP formulations of PWL functions. The latter requires additional variables and constraints to force the correct (depending on the amount of flow) linear segment of PWL functions to be taken. In our model, the correct segment in an s-shaped fragment is chosen automatically due to the network's structure. For the case when transfer functions are non-linear, we provide an error estimation for the approximated solution.**

I. Introduction

IN THIS paper, we consider flows in networks with non-linear losses (NL-flow problem). We have a directed graph $D = (V, A)$, where V is a set of vertices and A is a set of edges. The maximum flow that we can send through edge $a \in A$ is bounded by the edge's capacity $u_a \in R_+$. Each edge a of graph D has an associated non-linear function $F_a(f_a)$ that defines how outflow depends on inflow: we assume that if we send f_a units of flow into edge $a = (v, w)$, then $F_a(f_a)$ units of flow arrive at the head of edge a (that is node w).

In the classical maximum flow problem, the goal is to send as much flow as possible from the source node(s) to the target node(s), taking capacity and flow conservation constraints into account. *Transfer function* $F_a(f_a)$ defines the outgoing flow from edge a depending on the incoming flow f_a to edge a, $a \in A$. For the classical case, flow does not change while going through an edge, i.e. $F_a(f_a) = f_a$. This problem is well-studied, see e.g. Korte and Vygen [4].

The generalized flow problem (GFP) is a step closer to the NL-flow problem. The goal of the generalized maximum flow problem is to maximize flow at the target node. In a generalized graph, flow changes its value while going through edges. Outgoing flow $F_a(f_a)$ from edge a linearly depends on the incoming flow f_a to edge a, $a \in A$. Transfer functions corresponding to this type of flow are linear, i.e. $F_a(f_a) = \gamma_a \cdot f_a$, where γ_a is the proportionality coefficient corresponding to edge a. The GFP has already been studied by Onaga [6] and Truemper [9]. There are fully polynomial-time approximation schemes for GFP, see e.g. Fleischer and Wayne [2] and Oldham [5], and polynomial algorithms for GFP with assumptions about transfer coefficients, see Radzik [7] and Tardos and Wayne [8].

In this paper, we introduce flows with affine-linear transfer functions, i.e. $F_a(f_a) = \gamma_a \cdot f_a + b_a$, where k_a and b_a are constants corresponding to edge a, $a \in A$. We call optimization problems that correspond to this type of flow, *affine generalized flow problems* (AGFP).

For non-linear PWL functions, there exist standard ways to present them in mixed-integer formulations (see Vielma et al. [10]). Our representation is more efficient than representations applied to the general type of PWL functions. Standard ways use extra variables to force the use of the right segment of the PWL functions. We modify the network in such a way that by flow maximizing, the right segment will be chosen automatically. This allows us to get a problem formulation of significantly reduced size. The main contribution of this paper is that for a wide range of applications, we propose a solution, which deals with large MILP formulations arising by modeling problems with non-linearities. We propose efficient formulations of the maximum flow problem for networks with transfer functions of special types.

The remainder of this paper is organized as follows. In Section II-C, we introduce the required definitions and terms. We describe how to build a residual network for flows with NL functions and provide flow decomposition theorem for flows with NL losses. In Section III, motivated by the applications areas, we distinguish three special types of transfer functions: *concave, convex* and *s-shaped*. In Section IV, we design an equivalent problem representation of the original instance for the considered types of transfer functions. We show that the optimal solution of the maximization problem for this problem representation and the optimal solution of the maximization problem for the original instance are the same. For the problem with concave transfer functions, we propose a fully polynomial-time approximation scheme. For convex and s-shaped transfer functions we replace edges with NL transfer functions by network structures, where transfer functions of the edges are affine-linear. In Section V, we introduce MILP models for AGFP. In Section VI, we evaluate our MILP model. In section VII we consider the case of non PWL transfer functions and estimate the solution error, which arises by approximating the original transfer functions by PWL functions. Section VIII completes our paper with the conclusions.

II. Preliminaries

A. Problem Formulation

We can formulate the maximization problem for flows with NL as follows:

Problem 1:

Given $D = (V, A)$, $u_a \; \forall a \in A$, $F_a \; \forall a \in A$.
Find an $s - t-$flow, that
maximizes

$$\sum_{a \in \delta^-(t)} F_a(f_a)$$

subject to

$$\sum_{a \in \delta^+(v)} f_a - \sum_{a \in \delta^-(v)} F_a(f_a) \leq 0, \forall v \in V \backslash \{s, t\} \quad (1)$$

$$0 \leq f_a \leq u_a, \forall a \in A. \quad (2)$$

Equation (1) describes the flow conservation law, inequality (2) the edge capacity constraints.
Here, we allow positive excess at nodes. This does not contradict flow maximization at the target node.

B. Assumptions

Inspired by the applications of the NL-flow models (see Section III for details), we make the following assumptions on the type of transfer functions. The considered transfer functions are loss functions ($\gamma_a(f_a) \leq 1$). Increasing flow f_a incoming to edge a increases the outgoing flow $F_a(f_a) = \gamma_a(f_a) \cdot f_a$. Thus, the transfer functions are strictly monotonically growing functions. The next natural assumption is $F_a(0) = 0$.

The considered networks have no flow generating-cycles. In the context of our applications, flow-generating cycles would refer to perpetual flow sources, which is not possible in practice.

C. Definitions

Definition 1: The transfer multiplier $\gamma_a(f_a)$: $R_+ \to R_+$ is a quotient of the edge's transfer function $F_a(f_a)$ and flow f_a entering the edge:

$$\gamma_a(f_a) = \frac{F_a(f_a)}{f_a}, \text{ for } f_a \neq 0.$$

If $f_a = 0$, $\gamma_a(f_a)$ may be assigned an arbitrary number. Let's assume for this paper that if $f_a = 0$, then $\gamma_a(f_a) = 0$.
$\gamma_a(f_a)$ can be interpreted as the efficiency of sending flow f_a through edge a.

Definition 2: A transfer function is called *loss function* (flow decreases) if the transfer multiplier corresponding to this function is less than or equal to one, i.e.

$$\gamma_a(f_a) \leq 1 \; \forall f_a \in R_+.$$

Proposition 1: The optimization problem on a graph with multiple source nodes and multiple target nodes can always be transferred to the optimization problem on a graph with one source node and one target node.

Proof: The solution of the flow maximization problem (also for flows with NL-losses) is not influenced by the following two operations.

T is a set of the target nodes and S is a set of the source nodes.
g_i is the flow at node i. $g_i = 0$ for $g_i \in V \backslash \{S, T\}$. $g_i < 0$ for $g_i \in S$. $g_i > 0$ for $g_i \in T$.

- We can add an integrated source node s and an extra edge $a = (s, w)$ with $\gamma_a(\cdot) = 1$, $u_a = g_w$ to all nodes $w \in S$; node w becomes a transfer node, $g_s = \sum_{w \in S} g_w$ and g_w is assigned to zero.
- We can add an integrated target node t and an extra edge $a = (v, t)$ with $\gamma_a(\cdot) = 1$, $u_a = \infty$ to all nodes $v \in T$; node v becomes a transfer node, $g_t = \sum_{v \in T} g_v$ and g_v is assigned to zero.

■

Using Proposition 1 without loss of generality, we assume that the graph D has only one target node and only one source node.

Definition 3: An $s - t-$flow is a flow from the supply nodes s to target nodes t.

Definition 4: Let us call *flow at node* v the difference between flow outgoing from node v ($\sum_{a \in \delta^+(v)} f_a$) and flow incoming to node v ($\sum_{a \in \delta^-(v)} F_a(f_a)$), where $\delta^-(v) := \{(u, v) \in A\}$ is a set of edges entering node v and $\delta^+(v) := \{(v, w) \in A\}$ is a set of edges leaving node v.

Definition 5: The *affine-generalized network* is the network, in which edges have transfer functions of type $F_a(f_a) = f_a \cdot k_a + b_a$.

Definition 6: A network flow distribution $X : R_+^A \to R_+$ defines for all edges $a \in A$ *partition coefficients* $x_a : A \to [0, 1]$, which denote the ratio of the flow leaving node v and the flow sent through edge $a = (v, w)$, i.e.

$$x_a = \begin{cases} 0, \text{ if } \sum_{a \in \delta^+(v)} f_a = 0, \\ \frac{f_a}{\sum_{a \in \delta^+(v)} f_a}, \text{ otherwise.} \end{cases}$$

A feasible flow through the network f determines a flow distribution X.

Definition 7: A flow distribution X (for the given supply) is *feasible* if the corresponding flow is feasible, i.e. the flow does not violate flow feasibility constraints (1) and (2).

Lemma 1: The flow at the source node together with the feasible flow distribution X uniquely determine the flow at each edge of the given graph.

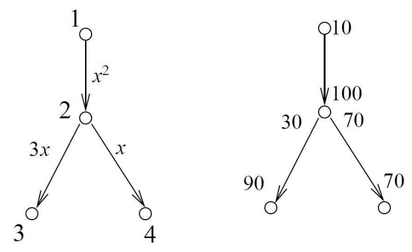

Fig. 1. Graph G and flow on it.

D. Decomposition theorem for NL flows

Here, we introduce the decomposition theorem for flows with NL transfer functions. This theorem will be used in the proof of Theorem 3.

Definition 8: We define the *residual capacity* for flow function $f_a : V \longrightarrow R_+$ as

$$u_a^R = u_a - f_a, a \in A, \tag{3}$$

$$u_{\overline{a}}^R = F_a(f_a), \overline{a} \in \overline{A}, \tag{4}$$

where $\overline{A}$ is a set of backward edges.

Definition 9: The residual function for backward edges can be found as:

$$F_{\overline{a}}^R(l) = f_a^* - F_a^{-1}(F_a(f_a^*) - l), \tag{5}$$

where f_a^* is the current flow on edge a, $l = f_{\overline{a}}$, we use notation l for better observability.

Definition 10: The residual function for forward edges can be found as:

$$F_a^R(g) = F_a(g + f_a^*) - F_a(f_a^*), \tag{6}$$

where f_a^* is the current flow on edge a, $g = f_a$, we use notation g for better observability.

Theorem 1: For every feasible flow f on a graph $D = (A, V)$, there exists a collection of $k \leq m = |A(D)|$ elementary residual flows $\mathcal{F} = \mathcal{F}(1), \ldots, \mathcal{F}(k)$ such that $f_a = \sum_{i=1..k} \mathcal{F}_a(i)$ for all $a \in A$. Elementary residual flows are residual flows on a path, on a unit-gain cycle, on a cycle-path, on a path-cycle or on a bicycle.

We omit the prove of the theorem, but give an example of decomposition.

Given graph G and flow on it, see Figure 1. We want to decompose the current flow to flow $\mathcal{F}(1)$ (through path $1-2-3$) and to flow $\mathcal{F}(2)$ (through path $1-2-4$) so that $f_a = \sum_{i=1,2} \mathcal{F}(i)$. Consider the graph and flow $f = (10, 30, 70)$ on it. Calculate the residual transfer function and the residual capacities for backward edges using formulas 4 and 5. $F_{21}^R(f) = f_{12} - \sqrt{(f_{12})^2 - f}$, where $f_{12} = 10$, $F_{32}^R(f) = \frac{1}{3}$, $F_{42}^R(f) = x$. $u_{21}^R(f) = 100$, $u_{32}^R(f) = 90$, $u_{42}^R(f) = 70$.

a is a part of the flow at the source node, that forms $\mathcal{F}(1)$, b is a part of the flow at the source node, that forms $\mathcal{F}(2)$, $a, b \in [0, 1]$, $a + b = 1$.

There are two ways to define $\mathcal{F}(1)$ and $\mathcal{F}(2)$ depending on the order of augmentation. Let us call the flow after the first augmentation $f(1)$, after the second augmentation $f(2)$.

1: We first augment $a \cdot f_{12}$ units of flow along path $1-2-3$ and then $b \cdot f_{12}$ units of flow along path $1-2-4$. Augment the flow along path $4-2-1$ by $F_{24}(f_{24})$ units. $F_{24}(f_{24}) = u_{1-2-4} = 70$. After augmenting, $f_{24}(1) = 0$, $f_{12}(1) = 10 - F_{21}^R(F_{42}^R(70)) = 10 - 10 + \sqrt{10^2 - 70} = \sqrt{30}$. $\mathcal{F}_{24}(1) = f_{24} = 70$, $\mathcal{F}_{12}(1) = f_{12} - f_{12}(1) = 10 - \sqrt{30}$.

We augment the current flow along path $3-2-1$ by $F_{23}(f_{23})$ units of flow. $F_{23}(f_{23}) = u_{1-2-3} = 90$. After augmenting, $f_{23}(2) = 0$, $f_{12}(2) = \sqrt{30} - F_{21}^R(F_{32}^R(90) = \sqrt{30} - \sqrt{30} + \sqrt{\sqrt{30}^2 - \frac{1}{3} \cdot 90} = 0$.

$\mathcal{F}_{23}(2) = f_{23} = 30$, $\mathcal{F}_{12}(2) = f_{12}(1) - f_{12}(2) = \sqrt{30}$.

Thus, we obtain $\mathcal{F}(1) = (10 - \sqrt{30}, 30, 0)$ and $\mathcal{F}(2) = (\sqrt{30}, 0, 70)$.

2: We first augment $a \cdot f_{12}$ units of flow along path $1-2-3$ and then $b \cdot f_{12}$ units of flow along path $1-2-4$. Augment the flow along path $3-2-1$ by $F_{23}(f_{23})$ units of flow. $F_{23}(f_{23}) = u_{1-2-4} = 90$. After augmenting, $f_{23}(1) = 0$, $f_{12}(1) = 10 - F_{21}^R(F_{42}^R(70)) = 10 - 10 + \sqrt{10^2 - \frac{1}{3} \cdot 90} = \sqrt{70}$. $\mathcal{F}_{24}(1) = f_{23} = 30$, $\mathcal{F}_{12}(1) = f_{12} - f_{12}(1) = 10 - \sqrt{70}$.

We augment the current flow along path $4-2-1$ by $F_{24}(f_{24})$ units of flow. $F_{24}(f_{24}) = u_{1-2-4} = 70$. After augmenting, $f_{24}(2) = 0$, $f_{12}(2) = \sqrt{70} - F_{21}^R(F_{32}^R(70) = \sqrt{70} - \sqrt{70} + \sqrt{\sqrt{70}^2 - 70} = 0$.

$\mathcal{F}_{24}(2) = f_{24} = 70$, $\mathcal{F}_{12}(2) = f_{12}(1) - f_{12}(2) = 10 - \sqrt{70}$.

Thus, we obtain $\mathcal{F}(1) = (\sqrt{70}, 0, 70)$ and $\mathcal{F}(2) = (10 - \sqrt{70}, 30, 0)$.

III. Special types of transfer functions and their applications

In this section, we consider three types of PWL transfer functions, which are especially interesting from the application point of view. We list them together with the application examples of the models, which use those types of transfer functions. Further, we use the special properties of these functions to find the solution of the maximum flow problem.

A. Convex transfer functions

In this case, the slope of the transfer function (and, thus, the transfer efficiency) grows as the amount of flow we send increases.

One application of convex transfer functions is modeling of information flows with "learning" effects. The flow passing along the graph represents information. Transfer functions model information transmission processes with "learning" effect, e.g. handwriting recognition, face recognition, speech

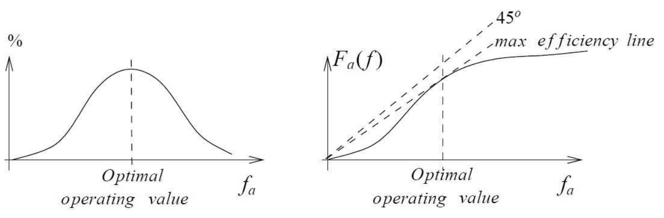

Fig. 2. Efficiency around optimal operating value and the corresponding S-shaped transfer function.

recognition. Efficiency of the information recognition/transmission process increases together with the amount of information.

B. Concave transfer functions

For this case, the slope of the transfer function (and thus, the transfer efficiency) shrinks as the amount of flow we send increases.

We can use concave transfer functions, e.g., in traffic flow modeling. It is a maximum NL-flow problem to find a flow routing with minimal overall flow-losses (deceleration). Edges represent road segment. Flow corresponds to the number of vehicles passing a reference point per unit of time. Transfer functions describe the flow-deceleration effect, which increases as flow approaches the capacity of a road segment.

C. S-shaped transfer functions

S-shaped transfer functions are used to model processes with optimal operating value. The slope of the transfer function grows until it reaches the optimal operating value. As we increase the amount of flow beyond the optimal operating value, the slope decreases.

S-shaped transfer functions are interesting for modeling energy flows. Nodes represent different types of energy (e.g. raw materials, electricity or heat energy), flow is energy (in different forms), edges represent transformation of one type of energy into another. Technical equipment that enables energy transformations usually has an optimal operating value, at which this equipment reaches its maximum efficiency. Thus, transfer functions show how the efficiency of the energy transformation first grows until the optimal operating value and then shrinks (see Fig. 2).

IV. Design of equivalent problem representations

In this section, we modify the underlying network in such a way that *Problem 2* from subsection IV-C has the same optimal solution for the original (*Problem 1*) and for the modified network. Further, in the next section, we use this modified network for establishing an efficient MILP formulation.

In the procedure described below, we replace a single edge $a = (v, w)$ by a set of p parallel edges $(a_1, a_2, \dots a_p)$.

Proposition 2: Edges $((v, w)_1, (v, w)_2, \dots (v, w)_p)$ are parallel edges between nodes v and w. Flow f is a flow that optimally solves the maximum flow problem (non-linear, generalized or classical). If flow f between nodes v and w is positive, then the capacity of the edges $(v, w)_i$, $i \in 1..p$, with higher efficiency is exhausted first.

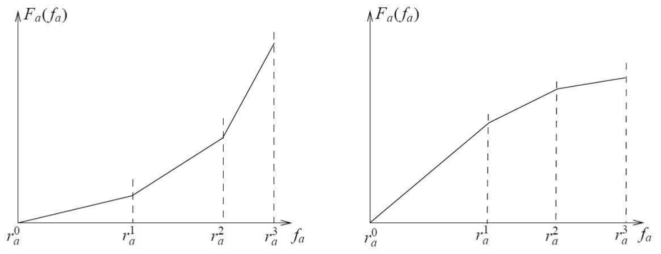

Fig. 3. PWL convex and concave functions.

A. Convex transfer functions

Let function F_a be a PWL convex function consisting of p segments defined by breakpoints: $0 = r_a^0 < r_a^1 < r_a^2 < r_a^3 < \dots < r_a^p$, for all $a \in A$ (see Fig. 3, left). Let us denote the function in the interval $[r_a^k, r_a^{k+1}]$ as F_a^k. F_a^k is a linear function of type $F_a^k = \gamma_a^k \cdot f + b$, where γ_a^k is the slope of function F_a^k. Since function F_a is convex, the slopes of the segments are related as follows:

$$\gamma^1 < \gamma^2 < \dots < \gamma^p. \tag{7}$$

We replace edge a by p parallel edges. The transfer function of the k-th edge is F_a^k, $a \in [1..n]$. The capacity of the k-th edge is equal to u_a.

Let us show that this replacement does not influence the optimal solution.

Function F_a^* is convex. Thus, the efficiency of sending flow through edge a grows as the amount of flow through edge a increases.

Since we maximize flow at the target node and, according to Proposition 2, prefer higher efficiency, only one edge of p parallel edges will be used.

$$F_a(f_a) = \begin{cases} \gamma_a^i \cdot f_a^i + b_a^i, & \text{if } f_a \in [r_a^{i-1}, r_a^i], \\ 0, & \text{if } f_a = 0 \end{cases} \tag{8}$$

If we send an amount of flow f_a through edge a and $f_a \in [r_a^k, r_a^{k+1}]$, the k-th edge will be used, because transfer functions F_a^i, $i > k$ or $i < k$ provide lower efficiency for f_a.

Thus, the replacement of edge a by p parallel edges in the way described above does not influence the solution of the maximization problem.

Now, the transfer functions of the edges are of type $\gamma f + b$ and the problem can be transformed to MILP-form (see Section V).

B. Concave transfer functions

Let function F_a be a PWL concave function consisting of p segments defined by breakpoints: $0 = r_a^0 < r_a^1 < r_a^2 < r_a^3 < \dots < r_a^p$, for all $a \in A$. Let us denote the slope of a PWL function in the interval $[r_a^k, r_a^{k+1}]$ as γ_a^k. Since function F_a is concave, the slopes of the segments are related as follows:

$$\gamma^1 > \gamma^2 > \dots > \gamma^p. \tag{9}$$

Flow f_a along edge a is the sum of flows along its segments.

$$f_a = \gamma^1 \cdot f_a^1 + \gamma^2 \cdot f_a^2 + \cdots + \gamma^p \cdot f_a^p = \sum_{k \in \{1..p\}} \gamma_a^k \cdot f_a^k \quad (10)$$

The capacity of the edge representing the k-th segment of edge a is equal to $r_a^k - r_a^{k-1}$. The flow along the k-th segment can be found as follows:

$$f_a^k = \begin{cases} 0, & \text{if } f_a \leq r_a^{k-1} \\ f_a - r_a^{k-1}, & \text{if } r_a^{k-1} \leq f_a \leq r_a^k \\ r_a^k - r_a^{k-1}, & \text{if } f_a \geq r_a^k \end{cases}$$

Every segment a^k can be represented as an edge with transfer function $F_a^k = f_a \cdot \gamma_a^k$ and assigned capacity $(r_a^k - r_a^{k-1})$. An edge a can be replaced by p parallel edges corresponding to the segments of F_a, for all $a \in A$. This replacement is an equivalent replacement, because the edges with the higher γ will be used (exhausted) first when flow is maximized.

The transfer function corresponding to the edges of the modified network are linear. Now we can transform the problem to LP-form and use standard LP-solvers or apply algorithms for the GFP-case.

For example, we can use the algorithm from [2], which takes $O(\epsilon^{-2} m^* (m^* + n \log m^*) \log n)$ time to compute the ϵ-optimal flow on a network with no flow-generating cycles. If we model energy flows, flow generating cycles refer to perpetual energy sources, which is not possible in practice. Flow is ϵ-optimal if $SOL(F) \geq (1-\epsilon) OPT(F)$, $m = |A|$, $n = |V|$ and m^* is the amount of edges in the modified network.

There are two ways to reduce the algorithm's complexity:

1: The generalized shortest-path problem is a subroutine of the GFP. On a graph with no flow-generating paths, it takes $O(m + n \log m)$ time to find a shortest path. We search for the shortest path on the residual graph. We can speed up this algorithm by a special way to set a residual graph. We use the idea described in [1] for convex cost functions. In the residual network, we do not need to consider all $2p$ (p forward and p backward) copies of edges between nodes v and w; it is sufficient to maintain only two edges: the first is for increasing flow through edge (v, w), the second edge is for decreasing flow on it.

Thus, it takes only $O(\epsilon^{-2} m^* (m + n \log m) \log n)$ time to compute the ϵ-optimal flow.

2: The second way to improve the computational performance of the algorithms is to use scaling of the transfer functions. The polynomial-time algorithm for convex cost (with integer costs) flow problems based on this method is presented in [1]. In the first scaling phase, we linearize a concave function $F_{(v,w)}$ by a PWL function consisting of 2 segments of length $\frac{u_{(v,w)}}{2}$. In the second phase, we linearize a concave function $F_{(v,w)}$ by a PWL function consisting of 4 segments of length $\frac{u_{(v,w)}}{4}$, and so on until we reach a segment length of $\frac{u_{(v,w)}}{2^n} \leq \epsilon$. Thus, we conduct n scaling phases, $n \geq \lceil \log \frac{U}{\epsilon} \rceil$, where $U = \max_{(v,w) \in A} u_{(v,w)}$. After each scaling phase, we make sure that the flow on the linearized residual network is maximum. It can be shown (analogously to the proof from [1]) that to keep the flow on the network maximum in k-th scaling phases, it is enough to increase or decrease the flow on every edge by $\frac{u_{(v,w)}}{2^k}$ units. There can be at most $O(m)$ augmentations in each scaling phase. The overall algorithm takes $O(m \lceil \log \frac{U}{\epsilon} \rceil)$ augmentations and requires $O(m + n \log m)$ time to find a shortest path, thus, runs in $O(m \lceil \log \frac{U}{\epsilon} \rceil (m + n \log m))$ time.

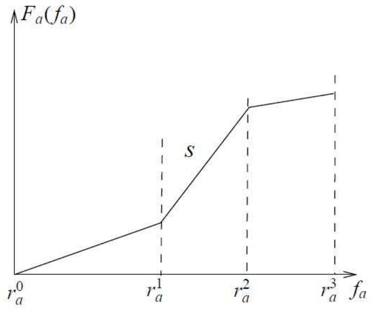

Fig. 4. S-shaped transfer function of edge a.

Algorithm 1 How to find inflection segment

```
Require: S-shaped PWL function
Ensure: Number of the inflection segment, s
    lower bound of s $s_b$:=1;
 2: upper bound of s $s_e$:=p;
    while $s_b \neq s_e$ do
 4:     $s_m := \lfloor \frac{s_e - s_b}{2} \rfloor$;
        if $\gamma_a^{s_b+s_m} > \gamma_a^{s_b+s_m+1}$ then
 6:         slope decreases, continue search on the left part
            $s_e := s_b + s_m$;
        else
 8:         slope grows, continue search on the right part
            $s_b := s_b + s_m + 1$;
        end if
10: end while
    s:=$s_e$; ($s_b$=$s_e$, thus s=$s_b$).
```

C. S-shaped transfer functions

Let function F_a be an s-shaped PWL function consisting of p segments defined by breakpoints: $0 = r_a^0 < r_a^1 < r_a^2 < r_a^3 < \cdots < r_a^p$, for all $a \in A$. Let us denote the slope of a PWL function in the interval $[r_a^k, r_a^{k+1}]$ as γ_a^k.

First, we divide the s-shaped function in two parts: a concave part and a convex part. With the help of Algorithm 1, which uses the fact, that $\gamma_a^j \neq \gamma_a^{j+1}$, $a \in A$, $j \in [1..p-1]$, we can easily find the number of the *inflection segment* s for edge a in $O(\lceil \log p \rceil)$ time. s is the number of the first segment, where the slope of the following segment is lower than the slope of the current segment. The inflection segment corresponds to the highest slope.

We use Algorithm 2 to build an equivalent representation of an s-shaped PWL transform function, see Fig 5. The new representation fits into MILP framework. The solution of the

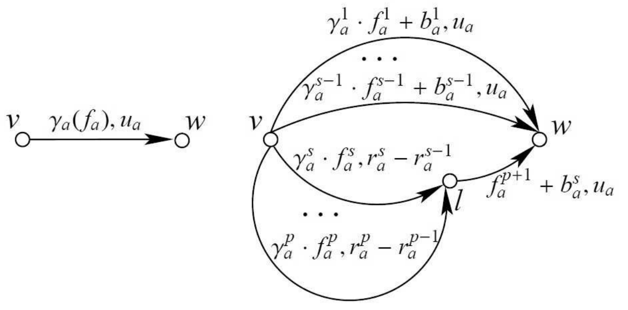
v
$\gamma_a(f_a), u_a$
w
$\gamma_a^1 \cdot f_a^1 + b_a^1, u_a$
$\gamma_a^{s-1} \cdot f_a^{s-1} + b_a^{s-1}, u_a$
$\gamma_a^s \cdot f_a^s, r_a^s - r_a^{s-1}$
$f_a^{p+1} + b_a^s, u_a$
l
$\gamma_a^p \cdot f_a^p, r_a^p - r_a^{p-1}$

Fig. 5. An equivalent representation of edge a with an s-shaped transfer function.

Algorithm 2 How to find the equivalent representation for s-shaped PWL transfer functions

```
Require: Initial underlying graph G=(V,A)
Ensure: Equivalent representation of graph G
    m:=| V |
 2: for $a = 1 \to m$ do
        apply Algorithm 1 for function $F_a$ to find the number of inflection segment for transfer function of edge $a = (v, w)$
 4:     for $i = 1 \to s-1$ do {the convex part of the function}
            add edge $(v, w)$ with
            $F_a^i = \begin{cases} \gamma_a^i \cdot f_a^i + b_a^i, \text{if} f_a^i > 0, \\ 0, \text{otherwise} \end{cases}$ and $u_a^i = u_a$;
 6:     end for
        for $i = s \to p$ do {the concave part of the function}
 8:         add edge $(v, l)$ with $F_a^i = \gamma_a^i \cdot f_a^i$ and $u_a^i = r_a^i - r_a^{i-1}$;
        end for
10:     add edge $(l, w)$ with $F_a^{p+1} = f_a^{p+1} + b_s$ and $u = \infty$.
    end for
```

flow maximization problem on the new representation (by summing over edges a^i, $i \in [1..p]$) and on the initial underling network will be equivalent.

The transfer function of edge a can be found as

$$F_a(f_a) = \begin{cases} 0, \text{if } f_a = 0, \\ \gamma_a^i \cdot f_a^i + b_a^i, \text{if } f_a \in [r_a^{i-1}, r_a^i] \text{ and } i < s, \\ \gamma_a^s \cdot f_a^s + b_a^s, \text{if } f_a \in [r_a^{i-1}, r_a^i] \text{ and } i = s, \\ \sum_{j \in [s..i-1]} \gamma_a^j \cdot u_a^j + \gamma_a^i \cdot f_a^i + b_a^s, \\ \qquad \text{if } f_a \in [r_a^{i-1}, r_a^i] \text{ and } i > s. \end{cases}$$

Capacity limitation edge. To maintain the capacity limit on edge a, $a \in A$, we introduce a total capacity limitation on edges a_i, $i \in [1..p_a]$.

$$\sum_{i \in [1..p]} u_{a_i} \leq u_a.$$

We can either integrate this constraints into the MILP formulation for all $a \in A$ or add a bottleneck edge with capacity u_a to node v so that all flow incoming to node v must first pass through this bottleneck edge. For convex and concave cases, the number of edges on the expanded graph per edge on the original graph p_a^+ becomes $p_a + 1$. For s-shaped transfer functions, p_a^+ becomes $p_a + 2$.

We can formulate the maximization problem on the modified network as follows:

Problem 2:

Given $D = (V, A)$, $u_a \; \forall a \in A$, $F_a \; \forall a \in A$.
Find an $s-t-$flow, that
maximizes

$$\sum_{a \in \delta^-(t)} \sum_{i \in 1..p_a^+} F_a^i(f_a)$$

subject to

$$\sum_{a \in \delta^+(v)} \sum_{i \in 1..p_a^+} f_a^i - \sum_{a \in \delta^-(v)} \sum_{i \in 1..p_a^+} F_a^i(f_a^i) \leq 0, \forall v \in V \backslash \{s, t\}$$

$$0 \leq f_a^i \leq u_a^i, \; \sum_{i \in [1..p_a]} u_a^i \leq u_a, \forall a \in A, i \in 1..p_a^+.$$

Theorem 2: The optimal solution of the flow maximization problem for *Problem 1* with s-shaped transfer functions can be generated from the optimal solution of the flow maximization problem for *Problem 2*.

Proof: If we send an amount of flow f_a through edge a and $f_a \in [r_a^{k-1}, r_a^k]$ and $k < s$, then (like for the standard convex case) the k-th edge will be used, because transfer functions F_a^i, $i > k$ or $i < k$, provide lower efficiency for f_a. And if we send flow $f_a < r_a^{s-1}$, maximization of the flow leads us to choose one of the edges in new representation, and send the whole flow f_a through this edge.

If we send $f_a \in [r_a^k, r_a^{k+1}]$ and $k \geq s$, then we are on the concave part of the function. Segment s is less efficient as any segment of the concave part. The function, which describes the s-shaped PWL function of segment s, can be written as $F_a^s = \gamma_a^s + b_a^s$. Per definition of concavity, for $i > s$ any following segment is less efficient than the previous and, thus, $\gamma_a^i > \gamma_a^{i+1}$ and $\gamma_a^i + b_a^s > \gamma_a^{i+1} + b_a^s$ for $i > s$. In the other words, if we send flow $f_a \geq r_a^{s-1}$, we first fulfill edge s, and only then, if $u_s < f_a$, the following edges. Then, $f_a = \sum_{i=s...p_a} f_a^i$.

■

Parallel edges. By allowing parallel edges, we meet some notational difficulties, because an edge cannot be uniquely specified by its tail and its head. We can use the following to build the equivalent network without using parallel edges. If there are parallel edges between node v and w, we replace all but one of these edges by a pair of series-connected edges. The first edge in the pair stays as original, the second one gets a transfer function equal to one and capacity equal to ∞. For implementation, instead of ∞, we use a big number, which guarantees no capacity limitation on this edge. Flow at any edge of our network is less than $\sum_{a \in \delta_-(s)} f_a$. Thus, in order not to create capacity limitations, we can use any number greater than $\sum_{a \in \delta_-(s)} f_a$. Thus, we can easily replace a graph with parallel edges by an equivalent graph without parallel edges. A negative effect of this replacement is that it expands the underlying network. For convex and concave cases, the number of edges on the expanded graph per edge on the original graph p_a^+ becomes $p_a + 1 + p_a - 1 = 2 \cdot p_a$.

For s-shaped transfer functions, in the subgraph replacing edge a, there are two sets of parallel edges, thus, p_a^+ becomes $p_a + 2 + p_a - 2 = 2 \cdot p_a$.

Graph expansion. If the PWL function of any edge $a \in A$ consists of p segments, the number of edges in the expanded graph becomes $m \cdot 2p$, where m is the amount of edges in the initial graph (with NL PWL transfer functions). If the number of segments in the approximation function p_a is different for every edge a, the number of edges in the modified network becomes $\sum_{a\in A} 2 \cdot p_a$ Let us denote the number of edges in the modified network as m^*.

D. Monotonically growing transfer functions

Monotonically growing PWL transfer functions can be divided into s-shaped fragments. For each s-shaped fragment, we establish an equivalent network representation as described in section IV-C. Thus, the correct segment within each s-shaped fragment is chosen automatically due to the network‘s structure. We need to use the extra variables only to force the right (depending on the amount of flow) s-shaped fragment to be taken.

V. MILP-model for max-flow problems on AGFP network

In this section, we present the formulation of the maximum flow problem on affine-linear generalized network in MILP form.

In the model, we have to integrate the following properties of F_a:

$$F_a = \begin{cases} f_a \cdot \gamma_a + b_a, & \text{if } f_a > 0 \\ 0, & \text{if } f_a = 0 \end{cases}$$

To do this, we use binary variables $f_a^* \in \{0,1\}$, $a \in A$. If $f_a^* = 1$, then flow f_a is greater then zero. If $f_a^* = 0$, then flow f_a is zero. The capacity of flow f_a through edge a is set to zero unless $f_a^* = 1$.

Given $D^* = (V, A)$ (D^* is an expanded/modified graph), u_a, k_a, b_a, r_a $\forall a \in A$.
Find an $s-t-$flow, that
maximizes

$$\sum_{a\in\delta^-(t)} (f_a \cdot \gamma_a + b_a \cdot f_a^*)$$

subject to

$$\sum_{a\in\delta^+(v)} f_a - \sum_{a\in\delta^-(v)} (f_a \cdot \gamma_a + b_a \cdot f_a^*) = 0, \forall v \in V\backslash\{s,t\}$$

$$f_a^* \in \{0,1\}, \forall a \in A$$

$$0 \le f_a \le u_a \cdot f_a^*, \forall a \in A$$

VI. Model Evaluation

Vielma et al. [10] study different ways to model PWL functions in MILP form. They consider the disaggregated convex combinations model, the logarithmic model, the logarithmic disaggregated model and other models. These models are characterized by size of the formulation. Since the quantitative measurement of constraints, continuous and binary variables in formulations is ambivalent (e.g. some models require less constraints, but need more binaries), computational experiments were conducted and presented in [10]. The performance of the logarithmic model was considered the best. According to [10], the crucial parameter that defines time performance is the number of additional continuous variables. Our model, which uses the special properties of the considered PWL functions, does not need additional continuous variables to choose the right edge. Remember, the right edge is the edge that corresponds to the segment of a PWL function that would be chosen if we sent $\sum_{i\in[1..p_a]} f_a^i$ through edge a. Thus, we reduce the number of continuous variables, and thus, improve the computational performance.

VII. Error Estimation

Let us consider the situation in which the original transfer functions $F(\cdot)$ are not PWL functions. Non-linear monotonically increasing functions can be approximated by monotonically increasing PWL functions ($F^A(\cdot)$), and the approach described in this paper can be applied. Obviously, the optimal solution for the original transfer functions $OPT(F)$ does not have to be equivalent to the optimal solution for the approximating transfer function $OPT(F^A)$. It is important to be able to estimate how the approximation quality influence dependence between $OPT(F)$ and $OPT(F^A)$.

The approximation error from above, $\epsilon^\uparrow$, can be defined as follows:

$$\epsilon^\uparrow = \max_{a\in A} \max_{0\le f_a \le u_a} \left(\frac{\gamma_a^{A\uparrow}(f_a) - \gamma_a(f_a)}{\gamma_a(f_a)} \right) =$$

$$= \max_{a\in A} \max_{0\le f_a \le u_a} \left(\frac{\gamma_a^{A\uparrow}(f_a)}{\gamma_a(f_a)} - 1 \right),$$

i.e. $\epsilon^\uparrow \ge \frac{\gamma_a^{A\uparrow}(f_a)}{\gamma_a(f_a)} - 1, \forall a \in A, 0 \le f_a \le u_a$.
The latter can be reformulated as follows:
$(\epsilon^\uparrow + 1)\gamma_a(f_a) \ge \gamma_a^{A\uparrow}(f_a), \forall a \in A, 0 \le f_a \le u_a$.

The approximation error from below, $\epsilon^\downarrow$, can be defined as follows:

$$\epsilon^\downarrow = \max_{a\in A} \max_{0\le f_a \le u_a} \left(\gamma_a(f_a) - \frac{\gamma_a^{A\downarrow}(f_a)}{\gamma_a(f_a)} \right) =$$

$$= \max_{a\in A} \max_{0\le f_a \le u_a} \left(\frac{1 - \gamma_a^{A\downarrow}(f_a)}{\gamma_a(f_a)} \right),$$

i.e. $\epsilon^\downarrow \ge 1 - \frac{\gamma_a^{A\downarrow}(f_a)}{\gamma_a(f_a)}, \forall a \in A, 0 \le f_a \le u_a$.
The latter can be reformulated as follows:
$(1 - \epsilon^\downarrow)\gamma_a(f_a) \le \gamma_a^{A\downarrow}(f_a), \forall a \in A, 0 \le f_a \le u_a$.

Theorem 3:

$$\frac{OPT(\gamma^{A\uparrow})}{(1+\epsilon^\uparrow)^z} \le OPT(\gamma) \le \frac{OPT(\gamma^{A\downarrow})}{(1-\epsilon^\downarrow)^z},$$

where z is the amount of edges in the longest $s-t$-path, at most $|A|$.

Proof: We split the proof into two parts.

Part 1. We prove that $OPT(\gamma) \geq \frac{OPT(\gamma^{A\uparrow})}{(1+\epsilon^{\uparrow})^{z}}$.

Suppose we know the optimal solution[1] $f_{OPT}^{A\uparrow}$ for $\gamma^{A\uparrow}(\cdot)$. According to the flow decomposition theorem (Theorem 1), we can decompose $f_{OPT}^{A\uparrow}$ into l paths, $l \leq |A|$. The amount of flow we send through path $P_i = a_1, a_2, \ldots, a_{ki}$, $i \in [1, l]$ is f_i. The amount of flow that reaches the target node through path P_i is $\gamma_{P_i}^{A\uparrow}(f_i)$[2].

Let us take the same amount of flow at the source node as we use to obtain $OPT^{A\uparrow}$ and send it through the network with transfer functions $\gamma(\cdot)$ according to distribution $X(f_{OPT}^{A\uparrow})$ (Def. 6). This yields flow f. Now, let us apply the same path representation (which was obtained by flow decomposition) in its distribution form X to flow f. Thereby, we do a valid decomposition of flow f into l paths. The flow we send through path P_i stays f_i, flow arriving at the target node through path P_i is $\gamma_{P_i}(f_i) = \gamma_{ki}(\gamma_{ki-1} \ldots \gamma_1(f_i))$.

Assume that we have a limit on the rate of transfer function growth, $\gamma_a^{'}(f_a)$, $0 \leq f_a \leq u_a$, $a \in A$. Then, $\gamma_k(\gamma_j(x)(1+\epsilon^{\uparrow})) \leq \gamma_k(\gamma_j(x))(1+\epsilon^{\uparrow})$, $\epsilon^{\uparrow} \geq 0$, $j, k \in A$.

Then, $\gamma_{P_i}^{A\uparrow}(f_i) \leq \gamma_{ki}(\gamma_{ki-1}(\ldots\gamma_1(f_i)(1+\epsilon^{\uparrow}))(1+\epsilon^{\uparrow}))(1+\epsilon^{\uparrow}) \leq \gamma_{ki}(\gamma_{ki-1}(\ldots\gamma_1(f_i)))(1+\epsilon^{\uparrow})^{ki} = \gamma_{P_i}(f_i)(1+\epsilon^{\uparrow})^{ki}$. Taking into account that $OPT(\gamma) \geq \sum_{i\in[1..l]} \gamma_{P_i}(f_i)$, we can state the following:

$OPT(\gamma^{A\uparrow}) = \sum_{i\in[1..l]} \gamma_{P_i}^{A\uparrow}(f_i) \leq \sum_{i\in[1..l]} \gamma_{P_i}(f_i)(1+\epsilon^{\uparrow})^{ki} \leq OPT(\gamma)\cdot(1+\epsilon^{\uparrow})^{z}$. q.e.d.

Part 2. We prove that $\frac{OPT(\gamma^{A\downarrow})}{(1-\epsilon^{\downarrow})^{z}} \geq OPT(\gamma)$.

This part of the proof is analogous to the first part.

First, we suppose to know the optimal solution for $\gamma(\cdot)$. Let us take a flow decomposition for this solution and apply it to $\gamma^{A\downarrow}(\cdot)$. By this, no capacity constraint is broken. The amount of flow at the target node for the second solution is greater than for the first. The optimal solution for $\gamma^{A\downarrow}(\cdot)$ is the solution that provides the greatest amount of flow at the target node for the graph with $\gamma^{A\downarrow}(\cdot)$. Thus, we can compare $OPT(\gamma^{A\downarrow})$ and $OPT(\gamma)$.

$OPT(\gamma) = \sum_{i\in[1..l]} \gamma_{P_i}(f_i) \leq \sum_{i\in[1..l]} \frac{\gamma_{P_i}^{A\downarrow}(f_i)}{(1+\epsilon^{\downarrow})^{ki}} \leq \frac{OPT(\gamma^{A\downarrow})}{(1+\epsilon^{\downarrow})^{z}}$. q.e.d.

■

[1]Remember, that the solution is the amount of flow at all edges a, $a \in A$, and $OPT(\cdot)$ is the amount of flow at the target node.

[2]Here, we do not consider transfer functions, but residual transfer functions. We omit writing next to each transfer function that it is a residual function to avoid overloading notations. Remember, if flow on an edge is zero, then the residual transfer function of this edge is equal to the transfer function of this edge, i.e. $F^{R}(0) = F$.

VIII. Conclusions

Better performance of the model introduced in this paper is based on the usage of the characteristics of special types of PWL functions.

For the maximum flow problem on networks with concave transfer functions, we propose a polynomial-time approximation scheme.

For the maximum flow problem on networks with monotonically growing transfer functions (this case can be reduced to convex and s-shaped transfer functions), we propose to split the transfer functions into s-shaped segments and then, with the help of extra variables, force only the right s-shaped fragment to be taken. We establish such a network representation that the correct segment within each s-shaped fragment is chosen automatically due to the network‘s structure. This yields to MILP model with better performance than if we were using standard MILP formulations of PWL functions, e.g. the logarithmic model. Moreover, it is worth to mention that our model is simple and transparent, which makes implementation easy.

References

[1] Ahuja, R.K., Magnanti, T.L. and Orlin, J.B. "Network flows: Theory, Algorithms and Applications." Prentice Hall, NJ, 1993.

[2] Fleischer, L. and Wayne, K.D. "Fast and simple approximation schemes for generalized flow." *Mathematical Programming*, Vol. 91, pp. 215-238, 2002.

[3] Keha, A.B., de Farias, I.R. and Nemhauser, G.L. "A branch-and-cut algorithm without binary variables for non-convex piecewise linear optimization." *Operations Research*, Vol. 54, pp. 847-857, 2005.

[4] Korte, B. and Vygen, J. *Combinatorial Optimization: Theory and Algorithms.* Algorithms and Combinatorics, 21 Springer, 2006.

[5] Oldham, J. "Combinatorial Approximation Algorithms for Generalized Flow Problems", In *Proceedings of ACM/SIAM*, 1999, pp. 135-169.

[6] Onaga, K. "Dynamic programming of optimum flows in lossy communication nets." *IEEE Transactions. Circuit Theory*, Vol. 13, pp. 308-327, 1966.

[7] Radzik, T. "Faster algorithms for the generalized network flow problem." *Mathematics of Operations Research*, Vol. 23, pp. 69-100, 1998.

[8] Tardos, E. and Wayne, K. "Simple Generalized Maximum Flow Algorithms." In *Integer Programming and Combinatorial Optimization,* Lecture Notes in Computer Science, Vol. 1412, pp. 310-324. Springer, 1998.

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

[10] Vielma, J.P., Ahmed, S. and Nemhauser, G. "Mixed-Integer Models for Nonseparable Piecewise Linear Optimization." *Discrete Optimization*, Vol. 5, pp. 467-488, 2008.