Algorithms and complexity for the almost equal maximum flow problem

Rebekka Haese1 | Till Heller2 | Sven O. Krumke1

1Department of Mathematics, Optimization Research Group, RPTU Kaiserslautern-Landau, Kaiserslautern, Germany
2Department of Optimization, Fraunhofer Institute for Industrial Mathematics ITWM, Kaiserslautern, Germany

Correspondence
Till Heller, Department of Optimization, Fraunhofer Institute for Industrial Mathematics ITWM, Kaiserslautern, Germany
Email: till.heller@web.de

Abstract
In the equal maximum flow problem (EMFP), we aim for a maximum flow where we require the same flow value on all arcs in some given subsets of the arc set, so called homologous arc sets. In this article, we study the closely related almost equal maximum flow problems (AEMFP) where the flow values on arcs of one homologous arc set differ at most by the valuation of a so called deviation function Δ. We prove that the integer AEMFP is in general \(\mathcal{NP}\)-complete, and show that even the problem of finding a fractional maximum flow in the case of convex deviation functions is also \(\mathcal{NP}\)-complete. This is in contrast to the EMFP, which is polynomial time solvable in the fractional case. Additionally, we provide inapproximability results for the integral AEMFP. For the (fractional) concave AEMFP we state a strongly polynomial algorithm for the linear and concave piecewise polynomial deviation function case for a fixed number of homologous sets using a parametric search approach.

KEYWORDS
computational complexity, equal-flow problem, integral solutions, maximum flows, network flows, parametric search

1 | INTRODUCTION

The maximum flow problem is a well-studied problem in the area of network flow problems (see [1]). Given a directed graph \(D = (V, A)\) with non-negative arc lower and upper bounds \(l, u : A \rightarrow \mathbb{R}\), a source \(s \in V\), a sink \(t \in V \setminus \{s\}\), one searches for a \(s,t\) flow \(f : A \rightarrow \mathbb{R}^{\geq 0}\) such that \(l(a) \leq f(a) \leq u(a)\) holds for all \(a \in A\) (capacity constraints), \(f(\delta^+(v)) - f(\delta^-(v)) = 0\) holds for all \(v \neq s, t\) (flow conservation), and such that the total amount of flow reaching the sink \(\text{val}(f) = f(\delta^-(t)) - f(\delta^+(t))\) is maximized. As in the standard notation from the literature, for a node \(v\) we denote by \(\delta^-(v)\) the set of incoming arcs, by \(\delta^+(v)\) the set of outgoing arcs, and for \(S \subseteq A\) we abbreviate \(f(S) := \sum_{a \in S} f(a)\).

In this article, we study a variant of the family of equal flow problems, which we call the almost equal flow problems (AEFP). In addition to the data for the maximum flow problem, one is given (not necessarily disjoint) homologous subsets \(R_i \subseteq A\) for \(i = 1, \ldots, k\), monotonically increasing deviation functions \(\Delta_i\), and one requires for the flow \(f\) the homologous arc set condition that \(f(a) \in [f_i, \Delta_i(f_i)]\) for all \(a \in R_i\), \(i = 1, \ldots, k\), where \(f_i := \min_{a \in R_i} f(a)\) denotes the smallest flow value of an arc in \(R_i\). In the special case that all deviation functions \(\Delta_i\) are given as the identity function, all arcs contained in the same homologous set must have the same flow value. This problem is known as the equal maximum flow problem (EMFP).

The AEFP has various applications and can be applied to a wide range of real-world problems. For example, overlapping homologous arc sets can be used to model “ramp-up” and “cool-down” processes in a network. Suppose the charging process of a battery consists of 12 periods. We construct a network graph \(D = (V, A)\) in the following way. Let \(s\) be the source of the graph and \(t\) be the sink and for each time period \(t_i\), we add a node \(H_i\) which represents the change of the battery state at period \(t_i\). Furthermore, we add a node \(C\) which represents the charging of the battery. The charging node is connected to each battery...
state node, and the upper capacity on this arc is given by the external technical process constraints. Since the flow value on an arc between two battery state nodes denotes the current battery state, we introduce the homologous sets as pairs of consecutive arcs between these battery state nodes. By introducing now a bound on the difference of the flow values, the internal process constraints can be incorporated. The graph is depicted in Figure 1. Note that this model can be extended in several ways. For example, one can add desired battery states by introducing lower and upper bounds on arcs between battery state nodes. Also initial or end battery levels can be incorporated. By computing an almost equal maximum flow on this underlying graph, one obtains a battery charging strategy regarding the external and internal process constraints.

### 1.1 Related work and article contribution

The EMFP and related problems have been studied for quite some time. Ali et al. (see [3]) considered a variant of the minimum cost flow problem, where \( K \) pairs of arcs are required to have the same flow value, which they called the equal flow problem. An integer version of this problem, where the flow on the arcs must be integer, was also studied by [3] and shown to be \( \mathcal{NP} \)-complete. They also obtained a heuristic algorithm based on a Lagrangian relaxation technique. Meyer and Schulz (see [12]) showed that the integer equal flow problem is not polynomially approximable (unless \( \mathcal{P} = \mathcal{NP} \)), even when the arc sets are of size two. Ahuja et al. (see [2]) considered the simple equal flow problem, where the flow value on arcs of a single subset of the arc set must be equal. Using Megiddo’s parametric search technique (cf. [10, 11]), they present a strongly polynomial algorithm with a running time of \( \mathcal{O}(m(m + n \log n \log n)^2) \).

Recently, an extension to networks defined as duplicates of graphs with equal flow on duplicate arcs was studied by Hassin and Poznaski (cf. [8]). A complexity study of a robust variant of the minimum cost equal flow problem was carried out by [4]. Other work on equal flow problems has been done, for example, by [5, 13]. Application settings for equal flow problems can be found, for example, in [9] where equal flows are used for optimization water supply under consideration of system requirements.

Our contribution is twofold. We present a formal definition of the almost equal maximum flow problem (AEMFP) and give complexity results for different problem variants regarding the deviation function. Furthermore, we present a strongly polynomial combinatorial algorithm based on the parametric search technique (cf. [10, 11, 15]) for the case of linear and concave deviation functions.

The rest of the article is organized as follows. In Section 2, we state the formal definition of the AEMFP and provide complexity and approximation results for the general case. The case of a linear deviation function, that is, a constant deviation, is discussed in Section 3 where also the strongly polynomial algorithm based on the parametric search technique is presented. In Section 4, different problem variants of the AEMFP are discussed, that is, the cases of concave and convex deviation functions. Finally, we then conclude with an overview of the results and a short outlook onto future research.

### 2 Problem definition, complexity, and approximation

In this section, we give a formal definition of the AEMFP. Let \( D = (V, A) \) denote a directed network graph with node set \( V \), a source \( s \), a sink \( t \), an arc set \( A \), and lower and upper bounds \( l, u : A \to \mathbb{R} \). Additionally, one is given homologous arc sets \( R_i \subseteq A \) and a corresponding deviation function \( \Delta \), which takes the minimal flow value on arcs in \( R_i \), as input and determines the upper bound of flow values on arcs in \( R_i \), that is, the allowed deviation between the minimal and the maximal flow value. The AEMFP can be formulated as the following optimization problem in the variables \( f_a \ (a \in A) \):

\[
\text{(AEMFP)} \quad \max \quad f(\delta^+(s)) - f(\delta^-(s)), \\
\text{s.t.} \quad f(\delta^+(t)) - f(\delta^-(t)) \leq 0, \\
f(\delta^+(v)) - f(\delta^-(v)) = 0 \quad \forall v \in V \setminus \{s, t\}, \\
l(r) \leq f(r) \leq u(r) \quad \forall r \in A, \\
f_i \leq f(r) \leq \Delta_i(f_i) \quad \forall r \in R_i, i = 1, \ldots, k,
\]
where \( f_i \) denotes the minimum flow value on arcs from \( R_i \), that is, \( f_i := \min_{a \in R_i} f(a) \). The following example shows the potential of relaxing the equal condition to an almost-equal condition in terms of the maximum flow value. Suppose we are given a directed network graph with a source node \( s \), a sink node \( t \) and additional \( \pi \) nodes. We denote the total number of nodes by \( n := \pi + 2 \), and the total number of edges by \( m \). For each of the additional \( \pi \) nodes there is a ingoing arc from \( s \) and an outgoing arc to \( t \). All these edges have no upper capacities. Furthermore, there exist a direct arc from \( s \) to \( t \) with an upper capacity of 1. For \( i = 0, \ldots, \pi \) we add homologous arc sets \( \{(i, t), (i + 1, t)\} \) if \( i \) mod 2 = 0, \( \{(s, i), (s, i + 1)\} \) if \( i \) mod 2 = 1 and \( \{(s, 0), (s, 1)\} \). Roughly speaking, the arcs on a path from \( s \) to \( t \) are always contained in two disjoint homologous arc sets connecting these sets together. See also Figure 2, where the same-colored arcs belong to the same homologous arc set. If all the arcs in the same homologous arc sets have to carry an equal amount of flow, the total maximum flow value is \( 1 + \pi \), since there is at most 1 unit of flow on the arc \((s, t)\) and therefore on all other arcs. However, if all homologous arc sets have the same linear deviation function \( \Delta : x \mapsto x + c \), the total maximum flow value is given as \( \sum_{i=0}^{\pi-1} (1 + c \cdot i) = \pi + \frac{\pi - 1}{2} \cdot c \) in comparison to the maximum flow value of \( 1 + \pi \) with the equal condition.

Considering the integral version of the AEMFP, we additionally require \( f \) to attain only integral values. Note that, in general the above problem is nonlinear due to the nonlinearity of the deviation functions \( \Delta_i \) in condition (5). However, if each \( \Delta_i \) is a linear function defined by a constant deviation \( c_i \), then (5) becomes \( f_i \leq f(r) \leq f_i + c_i \) for all \( r \in R_i \). Furthermore, the constraint \( f_i = \min f(a) : a \in A_i \) can be replaced by a set of linear constraints \( f_i \leq f(a) \). Therefore, the AEMFP can be formulated as a linear program and, thus, is polynomial time solvable. Due to constraint (4), the values \( f_i \) are bounded and, thus, all entries in the constraint matrix are \pm 1 \text{ and } 0 \). Therefore, the AEMFP can even be solved in strongly polynomial time by Tardos’ algorithm [14]. We will develop a strongly polynomial time combinatorial algorithm in Section 3 which also works for a larger class of deviation functions. The simple AEMFP is defined as the AEMFP with just one homologous arc set \( R \). Note that by subdividing arcs that are contained in several homologous arc sets, we can assume without loss of generality that the homologous arc sets are disjoint.

**Theorem 1.** The integer AEMFP is \( \mathcal{NP} \)-complete, even if all deviation functions are given as the same linear deviation function, the homologous sets are disjoint, the capacities are integral, and the graph is bipartite.

**Proof.** We prove this by a reduction from Exact-3-Set-Cover (X3C) (cf. [6]). An instance of X3C is given by a finite set \( A \) of elements and a collection \( S \) of 3-element subsets of \( A \). The question is, if there exists a subcollection \( S' \subseteq S \) such that every element in \( A \) occurs exactly in one set of \( S' \).

Let \( A = \{a_1, \ldots, a_q\} \) and \( S = \{S_1, \ldots, S_r\} \). Given an instance of X3C, we construct a directed graph \( D \) in the following way. For each of the sets \( S_i \in S \) we add a node \( S_i \) and for each of the elements \( a_j \in A \), we add a node \( a_j \) to \( D \). Further, we add a source node \( s \) and a sink node \( t \). We add arcs \( (s, S_i) \) for \( S_i \in S \) with capacity 5, arcs \( (S_i, t) \) for \( S_i \in S \) with capacity 2, arcs \( (a_j, t) \) for \( a_j \in A \) with capacity 1 and arcs between \( S_i \) and \( a_j \) if \( a_j \in S_i \) with capacity 1. The arcs of the form \( (S_i, t) \) are referred to as *bonus arcs*. A schematic representation of the graph is given in Figure 3. We set \( k := r + 1 := |S| + 1 \). We now define homologous arc sets \( R_i \) as \( \{(S_i, a_j) : a_j \in S_i\} \cup \{(S_i, t)\} \) for \( i = 1, \ldots, r \) and \( R_0 := \{ (a_j, t) : j = 1, \ldots, q \} \) where all these sets have the same linear deviation function \( \Delta : x \mapsto x + 1 \).

**FIGURE 2** Example of a network graph. Arcs of the same color are contained in the same homologous arc set. The only arc with capacity bounds is the direct arc from source \( s \) to sink \( t \).
The network graph used in the proof of Theorem 1. The corresponding instance of X3C is given by $k$ sets $S_i$ and $q$ elements $a_j$. The arcs between the source $s$ and the set nodes $S_i$ have a capacity of 5, whereas the arcs between the element nodes $a_j$ and the sink $t$ have a capacity of 1. The bonus arcs from set nodes $S_i$ to the sink $t$ have a capacity of 2. There are no lower bounds on any of the arcs.

Now we want to show that X3C has a solution if and only if there is an integer almost equal maximum s-t-flow in $D$ with value $\frac{7q}{3}$. Assume first that X3C has a solution $S'$. Then, we define an integer almost equal flow as follows:

$$f(s, S_i) = \begin{cases} 5, & \text{if } S_i \in S' \\ 1, & \text{else.} \end{cases}$$

$$f(S_i, t) = \begin{cases} 2, & \text{if } S_i \in S' \\ 1, & \text{else.} \end{cases}$$

$$f(S_i, a_j) = \begin{cases} 1, & \text{if } S_i \in S' \text{ and } a_j \in S_i \\ 0, & \text{else.} \end{cases}$$

$$f(a_i, t) = 1, \text{ for } i = 1, \ldots, q.$$

By definition of the flow $f$, the homologous arc set constraints, the flow conservation and capacity constraints are fulfilled. Hence, $f$ is an integer almost equal s-t-flow with flow value $\frac{7q}{3}$. Assume there is a flow $f'$ which has greater value than $f$. Due to capacity constraints, this flow must send at least one more unit of flow along an arc of the form $(S_i, t)$ with $S_i \notin S'$. Thus, at least one of the constraints (5) is violated. This is a contradiction to $f'$ being feasible and, hence, $f$ is maximal.

Conversely, assume that the almost equal maximum flow in $D$ has flow value $\frac{7q}{3}$. Due to constraint (5), only using bonus arcs yield in a flow with value $q$. Since all flow must be integral and flow preservation holds, we know that $f(S_i, a_j) \in \{0, 1\}$ for $a_j \in S_i$ and $i = 1, \ldots, r$. For a fixed node $S_i$ we distinguish two cases:

- If $f(S_i, a_j) = 1$ for three such arcs, then the bonus arc $(S_i, t)$ carries $\{0, 1, 2\}$ units of flow, or
- If $f(S_i, a_j) = 0$ for at least one of the three arcs $(S_i, a_j)$, then the flow value $f(S_i, t)$ lies in $\{0, 1\}$.

Suppose $f(a_i, t) = 0$ for at least one arc. By flow preservation, also $f(S_i, a_j) = 0$ and with (5), we get $f(S_i, t) \leq 1$. With the considerations above, we get an upper bound on the maximum flow value. From arcs of the form $(a_j, t)$ we get at most $q - 1$ units of flow in total, while at most $\frac{4q - 3}{3}$ bonus arcs can carry 2 units of flow and $q - \frac{4q - 3}{3}$ bonus arcs carry 1 unit of flow. Hence, we obtain a maximum flow value of $\frac{7q}{3} - 2$.

In order to get the desired flow value of $\frac{7q}{3}$, we need $f(a_i, t) = 1$ for all $j = 1, \ldots, q$. Thus, each $a_j$ receives one unit of flow from some set node $S_i$. Further, we need $f(S_i, t) = 2$ for at least $\frac{q}{3}$ arcs. We denote the corresponding indices as $i_1, \ldots, i_{\frac{q}{3}}$. This can only happen if for each of these $i_l$ case 1 is true. Now consider $V_l := \{a_j : f(S_i, a_j) = 1\}, l = 1, \ldots, \frac{q}{3}$. All these sets are subsets of $\mathcal{A}$ and are pairwise disjoint. Since $|V_l| = 3, l = 1, \ldots, \frac{q}{3}$, we get that

$$\cup_{l=1}^{\frac{q}{3}} V_l = \mathcal{A}. \quad (6)$$
Hence, $f(S_i, a_j) = 0$ for all other $i$. Choosing $S' := \{ S_i : l = 1, \ldots, \frac{q}{3} \}$ gives a X3C solution since each $a_j$ appears exactly once in it.

Given a solution $f$ to the AEMFP, one can verify this solution in polynomial time. Thus, the AEMFP is $\mathcal{NP}$-complete.

**Theorem 2.** Unless $P = \mathcal{NP}$, for any $\epsilon > 0$, there is no polynomial time $(2 - \epsilon)$-approximation algorithm for the integer AEMFP, even if we consider disjoint sets and a linear deviation function $\Delta : x \mapsto x + 1$.

**Proof.** We extend the instance of the proof of Theorem 1 by adding two additional nodes $t', t''$. Further, we add one arc $(t, t')$ with capacity $\frac{7q}{3}$, $\frac{7q}{3}$ parallel arcs $(t', t'')$ with capacity 1 and $\tau$ parallel arcs $(s, t'')$ with capacity 2, which we refer to as *bonus arcs*.

For the arcs $(t', t'')$ and $(s, t'')$ a homologous arc set $R_b$ with a linear deviation function $\Delta_b : x \mapsto x + 1$ is added. The node $t''$ is the new sink, that is, we are asking for a $s\rightarrow t''$-flow.

If there exists a solution of X3C, then flow value is equal to $\frac{7q}{3}$, as proven before, and all of the arcs $(t, t')$ can be fully saturated. Note that there are $\tau$ such arcs as given by the instance of X3C. This means, on every bonus arc $(s, t'')$ two units of flow can be sent. Overall, this yields in a flow value of

$$\text{val}(f_{\text{arc}}) = \frac{7q}{3} + \tau \cdot \frac{7q}{3} = 2. \quad (7)$$

Now assume that there exists no solution of X3C. Then the maximum flow value is at most $\frac{7q}{3} - 1$. Hence, at least one of the arcs $(t', t'')$ carries no flow. But since all of the bonus arcs are in the same homologous set together with the parallel arcs $(t', t'')$, each bonus arc can carry at most one unit of flow. Again, overall we get a flow value of

$$\text{val}(f_{\text{arc}}) \leq \frac{7q}{3} - 1 + \tau \cdot \frac{7q}{3}. \quad (8)$$

Thus, for $\tau \rightarrow \infty$, the approximation factor goes to 2. □

### 3. The Linear Deviation Case

We start with the simple AEMFP, that is, only consider one homologous arc set, but will generalize the result in the end to more than one homologous set. Let $D = (V, A)$ be a directed graph with a single homologous arc set $R$ and linear deviation function $\Delta_R : x \mapsto x + c$. For easier notation, we define $Q := E \setminus R$ as the set of all arcs that are not contained in the homologous arc set $R$. By the homologous arc set condition (5), we know that the flow value on each of the corresponding arcs must lie in an interval $[\lambda^*, \Delta_R(\lambda^*)] = [\lambda^*, \lambda^* + c]$, where $\lambda^*$ is unknown. For a guess value $\lambda$ consider the modified network $D_\lambda$, where we set the upper capacity of every arc in $R$ to $\lambda + c$ and its lower capacity from 0 to $\lambda$. All arcs in $Q$ keep their upper capacities and have lower capacity of 0. By $f_\lambda$ we denote a traditional $s\rightarrow t$-flow which is feasible in $D_\lambda$.

For an $(s, t)$-cut $(S, T)$ let us denote by

$$g_S(\lambda) := u(\delta^+(S \cap Q)) + \sum_{r \in \delta^+(S \cap R)} \min\{u(r), \lambda + c\} - \sum_{r \in \delta^-(S \cap R)} \lambda,$$

its capacity in $D_\lambda$. By the max-flow min-cut theorem we get

$$F(\lambda) := \max_{f_\lambda} \text{val}(f_\lambda) = \min_{(S, T) \text{ is a } (s\rightarrow t)-\text{cut}} g_S(\lambda). \quad (9)$$

Since the function $F(\lambda)$ is the minimum of linear functions in $\lambda$, it is a concave linear function in $\lambda$. By a direct consequence of the max-flow min-cut theorem, the AEMFP can be solved by solving max \( \{ F(\lambda) : 0 \leq \lambda \leq \min_{r \in R_\lambda} u(r) \} \). In the following, we obtain structural results of the function $F$.

**Lemma 3.** The function $F(\lambda)$ has at most $2m$ breakpoints.

**Proof.** Let $d_R(S) := |\delta^+(S \cap R)| - |\delta^-(S \cap R)|$ denote the number of outgoing and ingoing arcs of $R$ in the cut $(S, T)$. A breakpoint of $F(\lambda)$ occurs whenever the cut $(S, T)$ changes in a way that changes $d_R(S)$. As $F(\lambda)$ is concave and $d_R(S)$ counts arcs, this can happen at most $2m$ times. □

**Lemma 4.** The minimum distance between two of these breakpoints is $\frac{1}{m}$.

**Proof.** At a breakpoint, we have

$$u(\delta^+(S \cap Q)) - l(\delta^-(S \cap Q)) + \lambda d_R(S) = u(\delta^+(S' \cap Q)) - l(\delta^-(S' \cap Q)) + \lambda d_R(S').$$
for two cuts \((S, T), (S', T')\). This gives an expression for \(\lambda\) as

\[
\lambda = \frac{(u(\delta^+(S \cap Q)) - l(\delta^-(S \cap Q))) - (u(\delta^+(S' \cap Q)) - l(\delta^-(S' \cap Q)))}{d_R(S') - d_R(S)}.
\]

Note that the denominator is not zero since \(d_R(S) \neq d_R(S')\) by definition of a breakpoints. Therefore, the expression for \(\lambda\) is well-defined. Further, we also know \((u(\delta^+(S \cap Q)) - l(\delta^-(S \cap Q))) \neq (u(\delta^+(S' \cap Q)) - l(\delta^-(S' \cap Q)))\). By denoting \(U := \max\{u(r) : r \in A\}\) and \(L := \min\{l(r) : r \in A\}\), we get

\[
(u(\delta^+(S \cap Q)) - l(\delta^-(S \cap Q))) - (u(\delta^+(S' \cap Q)) - l(\delta^-(S' \cap Q)))
= \frac{(u(\delta^+(S \cap Q)) - u(\delta^+(S' \cap Q))) - (l(\delta^-(S \cap Q)) - l(\delta^-(S' \cap Q)))}{U - L}.
\]

W.l.o.g. we can rearrange the two cuts such that both nominator and denominator are positive, that is, the nominator lies in \([1, \ldots, mU - mL]\) and the denominator lies in \([1, \ldots, |R|]\) since it just counts the arcs. Thus, we get for the breakpoint \(\lambda\):

\[
\frac{1}{m} \leq \frac{1}{|R|} \leq \lambda \leq \frac{m(U - L)}{1}.
\]

Hence, the smallest distance between two breakpoints is

\[
\frac{1}{m(m-1)} > \frac{1}{m}.
\]

Observe that the optimal value \(\lambda^*\) is attained at a breakpoint of \(F\). At this point the slope to the left is positive or the slope to the right is negative. If there exists a cut such that the slope is 0, we simply take the breakpoint to the left or right of the current value \(\lambda\).

Now we apply the parametric search technique by Megiddo (cf. [10, 11]) to search for the optimal value \(\lambda^*\) on the interval \([0, u_R]\), where \(u_R := \min_{a \in \mathbb{R}} u(r)\) denotes the minimum upper bound of arcs in \(R\). We simulate an appropriate maximum flow algorithm, that is, the Edmonds–Karp algorithm, for symbolic lower capacities \(\lambda^*\) and upper capacities \(\lambda^* + c\) on the arcs in \(R\). See Algorithm 1 for a description of the algorithm in pseudo-code.

**Observation 5.** If we run Algorithm 1 to compute a maximum flow with a symbolic input parameter \(\lambda\), all flow values and residual capacities which are calculated during the algorithm steps are of the form \(a + b\lambda\) for \(a, b \in \mathbb{Z}\).

**Proof.** At the start of our algorithm, all flow values are zero. The residual capacities are either integer or of the form \(b\lambda\) for some \(b \in \mathbb{Z}\), thus can be written as \(a + b\lambda\). Whenever we augment flow along a path, we add two values of the form \(a + b\lambda\), resulting in a new value of the same form.

**Algorithm 1.** Symbolic Edmonds–Karp

1. **Input:** A directed graph \(D = (V, A)\), a source \(s\) and a sink \(t\), capacities \(c_a\) for all \(a \in A\).
2. **Initialization:** Set \(f_a \leftarrow 0\) for all \(a \in A\).
3. **while there exist a path** \(p\) in \(D_f\) **do**
4. | Choose shortest path in \(D_f\) w.r.t. the number of arcs.
5. | Compute \(\Delta := \min_{a \in p} c_a\) by using Algorithm 2 for solving symbolic comparisons.
6. **foreach** \((i, j) \in p\) **do**
7. | \(f_{ij} \leftarrow f_{ij} + \Delta\)
8. | \(f_{ji} \leftarrow f_{ji} - \Delta\)
9. **return** \(f\).

**Lemma 6.** Algorithm 1 computes an almost equal maximum flow in time \(\Theta(n^3 m \cdot T_{MF}(n, n+m))\), where \(T_{MF}(n, n+m)\) denotes the time needed to compute a maximum flow on a graph with \(n\) nodes and \(n + m\) arcs.

**Proof.** **Correctness:** In order to resolve a comparison, we need to decide if the current \(\lambda\) is to the left or to the right of the optimal \(\lambda^*\). Since \(\lambda\) might be a breakpoint, we instead check \(\lambda_1 := \lambda - \frac{1}{2m}\) and \(\lambda_2 := \lambda + \frac{1}{2m}\) and denote the corresponding cuts as \(S_1, S_2\). If \(d_R(S_i)\) is positive for \(i = 1, 2\), then \(\lambda^* > \lambda\), since the flow value increases to the right of the current \(\lambda\). If \(d_R(S_i)\) is negative for \(i = 1, 2\), then the flow value increases to the left of the current \(\lambda\), that is, \(\lambda^* < \lambda\). Hence, since the objective function is concave, the only remaining case is \(d_R(S_1) > 0\) and \(d_R(S_2) < 0\). In this case, the flow value decreases in both direction, and we just have to find the unique breakpoint.
in the interval between $\lambda_1$ and $\lambda_2$. This can be done by computing the intersection of $g_S(\lambda_1)$ and $g_S(\lambda_2)$. In total, the comparison is correctly resolved. The proposed algorithm is the usual Edmonds–Karp algorithm except for the comparison which has to be made in order to compute the augmenting path. If the comparison is made correctly, this has no influence on the correctness of the Edmonds–Karp algorithm. However, this is the case, since the question $a_1 + b_1 \lambda < a_2 + b_2 \lambda$ is equivalent to $\lambda < \frac{a_1-a_2}{b_1-b_2}$. Therefore Algorithm 1 is also correct.

Now for the algorithm: we set the lower and upper capacities of all arcs in $R$ to $\lambda$ to ensure that they have the same flow. Then, by applying the Edmonds–Karp algorithm that runs with a symbolic parameter on $D'$, we find a feasible flow. With this, we have a starting flow for the almost equal maximum problem and we can use Algorithm 1 to find an optimal flow since the residual network respects lower capacities.

**Running time:** The Algorithm 1 has at most $O(nm)$ iterations. In each of these, a shortest $s$-$t$-path $P$ w.r.t. the number of arcs is calculated, which can be done with breadth-first-search and therefore needs time $O(n + m)$. Then the algorithm computes the minimum residual capacity on the arcs of $P$. Since $P$ has at most $n - 1$ arcs and each of the residual capacities may depend on the parametric value $\lambda^*$, there are at most $O(n^2)$ comparisons which the algorithm has to resolve. Updating the residual network takes at most $2m$ comparisons of the form $a + b\lambda < u(r)$ and $a + b\lambda > 0$, thus at most $O(m)$ comparisons have to be resolved by using Algorithm 2. For each comparison, a maximum flow and the corresponding cut have to be computed. Since all other operations are done in constant time, the running time of resolving one comparison is $T_{\text{ALG}} \leq O(T_{\text{MF}}(n, m))$. Altogether, Algorithm 1 runs in time $O(nm \cdot (n^3 + m) T_{\text{ALG}}) \leq O(n^3m \cdot T_{\text{MF}}(n, m))$. So, in total the algorithm has a running time of $O(m + (n^3m T_{\text{MF}}(n, n + m)) + (n^3m T_{\text{MF}}(n, n + m))) \leq O(n^3m \cdot T_{\text{MF}}(n, n + m))$. 

**Algorithm 2. Solve comparison**

1. **Input:** A directed graph $D$, lower and upper capacity functions $l, u$, a homologous arc set $R$ and a test value $\lambda$
2. **Initialization:** Set $\lambda_1 := \lambda - \frac{1}{2m} \cdot \lambda_2 := \lambda + \frac{1}{2m}$
3. **for** $i = 1, 2$ **do**
   1. Compute a maximum flow in $D_{\lambda_i}$
   2. Compute $d(S_i)$ for the corresponding cuts $S_i$
4. **if** $d(S_1) > 0$ and $d(S_2) > 0$ **then**
   5. **return** False
5. **else**
   6. **if** $d(S_1) < 0$ and $d(S_2) < 0$ **then**
      7. **return** True
   8. **else**
      9. Compute $\lambda^*$ as the intersection of $g_S(\lambda)$ and $g_S(\lambda)$.
10. **return** $\lambda^*$

The number of comparisons can be decreased by exploiting implicit parallelism (cf. [11]). When building the residual network, the algorithm has to solve $l(r) < f(r)$ and $f(r) < u(r)$ for every arc $r \in E$. Since $f(r) = a + b\lambda$, we have to solve up to $2m$ comparisons. Instead of this, we can first calculate all the values $v$ for which we want to test $\lambda^* < v$ and sort them. This takes time $O(m \log m)$ and afterwards we apply a binary search over these values. In total, we can compute the residual network in time $O(m \log m \cdot T_{\text{ALG}})$. With the same trick, the time needed to find the minimum residual capacity on a path $P$ is $O(n \log n \cdot T_{\text{ALG}})$. This results in a running time of $O((nm(n \log n + m \log m)T_{\text{MF}}(n, n + m))$.

To solve the integer version of the maximum AEMFP, we simply use the optimal value $\lambda^*$ of the non-integer version and compute two maximum flows on the graphs $D_{\lambda^*}$ and $D_{\lambda}$. By taking the $\arg\max\{val(f_{\lambda^*}), val(f_{\lambda})\}$ we get the optimal parameter $\lambda^*_{\text{int}}$ for the integer version. In the general constant deviation AEMFP we consider more than one homologous arc set. By iteratively using the algorithm for the simple constant deviation AEMFP, we obtain a combinatorial algorithm for the general constant deviation AEMFP. We present the algorithm for the case of two homologous arc sets, but it can be generalized to an arbitrary number of homologous arc sets. The idea behind the algorithm is to fix some $\lambda_1$ and then use the algorithm for the simple case to find the optimal corresponding $\lambda_2$. Once we found $\lambda^*_2(\lambda_1)$, we check if $\lambda_1$ is to the left, right or equal to $\lambda^*_1$. Note that the objective function is still a concave function in $\lambda_1$ and $\lambda_2$ since it is the sum of concave functions. Also, like in the simple case, all flow values and capacities both in the network $D$ and the residual network $D_f$ during the algorithm are of...
the form $a + b\lambda_1 + c\lambda_2$. Note that the running time of the algorithm for the general constant deviation AEMFP increases for every additional homologous arc set roughly by a factor of the running time of the algorithm for the simple constant deviation AEMFP. The next theorem summarizes the results above.

**Theorem 7.** Let $T_{mf}(n, m)$ denote the running time of a not specified maximum flow algorithm on a directed graph $D$ with $n$ nodes and $m$ arcs. The AEMFP with $k$ homologous sets can be solved in time $O(n^3 m^3 \log(\log(n))^k \cdot T_{mf}(n, n + m))$ when we use the Edmonds–Karp algorithm as the underlying maximum flow algorithm.

Note that the running time for an arbitrary number of homologous arc sets becomes exponential. Note that using one of the known faster maximum flow algorithms instead of the Edmonds–Karp algorithm does not necessarily yield an improved running time (cf. [7]).

### 4 | PROBLEM VARIANTS

In the previous section, we considered the case of a constant deviation of the flow value on arcs within a homologous arc set. Now we allow the deviation function to be given as either a convex or a concave function.

#### 4.1 | The convex deviation case

If the deviation function is a convex function $\Delta_{\text{conv}} : R \rightarrow \mathbb{R}_{\geq 0}$, we get the convex AEMFP. Note that this problem is (in general) neither a convex nor a concave program due to constraint (5). Hence, standard methods of convex optimization cannot be applied. In fact, the next theorem states that, unless $P = \mathcal{NP}$, one cannot hope to find a polynomial time algorithm that solves the fractional variant of this problem:

**Theorem 8.** The AEMFP with a convex deviation function $\Delta$ is $\mathcal{NP}$-complete, even if all deviation functions are given as $\Delta_R : x \mapsto 2x^2 + 1$ for all homologous sets $R$, the homologous sets are disjoint, the capacities are integral, and the graph is bipartite.

*Proof.* Again we use a reduction from Exact-3-Cover. Given an X3C instance, we construct a network graph in the same way as in the proof of Theorem 1. We now show that there exists an almost equal maximum flow with convex deviation functions and flow value $\frac{3q}{4}$ if and only if there exists a solution of X3C.

Assume first that X3C has a solution $S'$. Then, we define an almost equal maximum flow in the same way as in the proof of Theorem 1. Suppose there is an almost equal maximum flow $x$ with flow value $\text{val}(x) > \frac{3q}{4}$. Since the capacity of arcs $(a_i, t)$ is 1, the summarized amount of flow on these arcs is at most $q$. That means, the sum of flow on arcs $(S_i, t)$ has to be greater than $\frac{3q}{4} + 1$. Suppose a flow $x'$ uses more than $q$ arcs of $(S, S_i)$. Then there must be at least two arcs $(S_i, t), (S_j, t)$ with $x'(S_i, t) \in (1, 3)$ for $k = 1, 2$. By the homologous arc constraint we know that at least one arc of the form $(S_{a_i}, a_j)$ and one of the form $(S_{a_j}, a_i)$ are not fully saturated, that is, $x'(S_{a_j}, a_i) \in (0, 1)$ for $k = 1, 2$. Without loss of generality, let $x'_{S_j, a_j} := x'(S_{a_j}, a_j) \geq x'(S_{a_j}, a_i) =: x'_{S_i, a_i}$. We show that this flow cannot yield the highest possible flow value, since shifting $\varepsilon$ between these two arcs yields in a higher flow value. Note that the sum $\sum_{k=1}^{2} x'_{k}$ remains the same, only the value of the arcs $(S_k, t)$ changes. In the following, we distinguish the following three cases.

**Case 1:** the arcs $(S_{a_j}, a_i)$ have both the strict smallest flow values in their homologous arc set. Then we shift an $\varepsilon$ from $(S_{a_j}, a_i)$ to $(S_{a_j}, a_i)$. This yields

$$2(x'_{a_i} + \varepsilon)^2 + 1 + 2(x'_{a_i} - \varepsilon)^2 + 1 = 2(x'_{a_i})^2 + 2\varepsilon(x'_{a_i} - x'_{a_i}) + 2\varepsilon^2 + 2 + 2(x'_{a_i})^2 > 2(x'_{a_i})^2 + 1 + 2(x'_{a_i})^2 + 1.$$  

Hence, the sum of the flow value on the arcs $(S_{a_j}, t)$ for $k = 1, 2$ is higher which is a contradiction to the maximality of $x'$.

**Case 2:** $x'_{S_i, a_i}$ is the smallest value among the flow values on arcs of the corresponding homologous arc set and $x'_{S_i, a_i}$ is strictly larger than the smallest flow value of arcs of the corresponding homologous arc set. Then, increasing $x'_{a_i}$ by $\varepsilon \leq \min\{u_1 - x'_{a_i}, x'_{a_i} - x'_{2,\text{min}}\}$ with $x'_{2,\text{min}} = \min_{R} R$ yields

$$2(x'_{a_i} + \varepsilon)^2 + 1 + 2(x'_{2,\text{min}})^2 + 1 = 2(x'_{a_i})^2 + 2\varepsilon x'_{a_i} + 2\varepsilon^2 + 2 + 2(x'_{2,\text{min}})^2 + 1 > 2(x'_{a_i})^2 + 1 + 2(x'_{2,\text{min}})^2 + 1.$$  

Again, this is a contradiction to the maximality of $x'$.
Case 3: \( x'_{2} \) is the smallest flow value on arcs of the related homologous arc set, \( x'_{1} \) is strictly larger than the minimum. Then, shifting \( \epsilon \) units of flow from \( x'_{1} \) to \( x'_{2} \) gives us
\[
2(x'_{1\text{min}})^2 + 1 + 2(x'_{2} + \epsilon)^2 + 1 = 2(x'_{1\text{min}})^2 + 1 + 2(x'_{2})^2 + 2\epsilon^2 + 1
\]
\[
> 2(x'_{1\text{min}})^2 + 1 + 2(x'_{2})^2 + 1.
\]
Also in this case the increasing resp. decreasing by \( \epsilon \) yields in a higher flow value—a contradiction to the maximality of \( x' \).

Conversely, assume that the almost equal maximum flow in \( D \) has flow value \( \frac{8q}{3} \). We need to show that this induces a solution to X3C and this can be done similar to the proof of Theorem 1 under consideration of the three cases above. This settles the proof.

**Theorem 9.** Unless \( P = NP \), there is no polynomial time constant factor approximation algorithm for the integer convex AEMFP.

**Proof.** For this, we use the same reduction as in Theorem 2. For the concave deviation function of the set \( R_b \), we choose \( \Delta_{k}^{b} : x \mapsto kx \). Then, one can see that the maximum flow value is 0 if no solution of X3C exists and \( \frac{2}{3} q + k \) if one exists. Thus, unless \( P = NP \), no polynomial time constant factor approximation algorithm can exist.

### 4.2 The concave (polynomial) deviation case

Unlike the convex case, which is \( NP \)-complete even for the fractional case, the concave case is polynomially solvable, since in this case the AEMFP becomes a concave program. In the following, we describe an algorithm for this variant, again using the parametric search technique (cf. [10, 11]) and a refinement by Toledo (cf. [15]). We restrict ourselves to the case of a single homologous arc set \( R \), but the algorithm can be extended to an arbitrary number of homologous arc sets according to [15]. As we have seen, we can solve the AEFMP for fixed lower bounds \( \lambda \), for each homologous arc set \( R_i \) by a maximum flow computation. This is in fact independent on the structure of the deviation function(s). Therefore, one can use the parametric search technique of Megiddo (cf. [10, 11]) with symbolic input parameters \( \lambda_{i}^{*} \) to find the (unknown) minimizer of the function \( F \) defined in (9).

We now consider the changes when the affine deviation function \( x \mapsto x + c \) is replaced by a (piecewise) concave polynomial deviation function \( \Delta \) of degree at most \( q \). The function \( F \) from (9) becomes a (piecewise) concave polynomial function also of degree at most \( q \), since it is the pointwise minimum of a finite number of (piecewise) polynomial functions \( g_{\Delta} \). We also know that \( F(\cdot) \) has no jumps between two breakpoints. So we restrict ourselves to an interval between two breakpoints and find a maximizer \( x_{i}^{*} \) for each such interval \( I \). In a second step, we evaluate all these local maximizers and find the global solution \( x^{*} \). The problem of finding a maximizer in each of the \( m \) intervals can then be solved simultaneously. Using a standard trick in network optimization, we can assume that the directed graph \( D \) has lower bounds 0, and we can apply the Edmonds–Karp algorithm to it. The maximum flow algorithm must answer questions of the form \( f_{x^{*}}(a) = p_{a}(\lambda^{*}) \leq \min \{ c(a), \Delta(\lambda^{*}) \} \). Such a comparison by the algorithm is equivalent to asking what sign a concave polynomial function \( p \) has at a given point \( x \).

Let us now consider the situation when we simulate the Edmonds–Karp algorithm in the (piecewise) polynomial case:

**Observation 10.** During the algorithm, all flow values and all residual capacities can be described by a polynomial \( p \) in \( \lambda \) which is of degree at most \( q \).

**Proof.** We start with the zero flow. In the first step, the residual capacities are either integer or \( \Delta(\lambda) \) for some polynomial \( \Delta \) of bounded degree. Thus, the residual capacities can be written as a polynomial \( p(\lambda) \) of bounded degree. Whenever the algorithm augments the flow along a path, it adds two values of the form \( p_{1}(\lambda), p_{2}(\lambda) \), where \( p_{1}, p_{2} \) are again polynomials with degree at most \( q \). This results in a flow value of the same form.

Since the sign of a polynomial is constant between two roots, it is sufficient to restrict ourselves to the roots \( \{ r_{1}, \ldots, r_{l} \} \) of the polynomial \( p \). For every root, we evaluate \( F \) and test if its evaluation is equal to \( x^{*} \) or else if it is to its left or right. We know that we can determine the relative position of a point \( x \) to \( x^{*} \) by evaluating \( F \) at this point. In the case of a linear deviation function, we did this by computing the slope of \( F \) at \( x \). Here, instead of relying on the slope, we use the idea by Toledo (cf. [15]). This process is presented in Figure 4. Evaluating \( F \) at a point \( x_{1} \) is a maximum flow computation in the graph \( D_{x_{1}} \), that is, the graph where the lower bound on arcs of the homologous arc set is set to \( x_{1} \). Now we distinguish two cases, either \( x_{1} \) is to the left or to the right of the maximum \( x^{*} \). First we check if we have already evaluated \( F \) at a point \( x_{0} \) with \( F(x_{1}) \geq F(x_{0}) \). If this is the case, we know that the maximum \( x^{*} \) lies in the direction of \( x_{0} \). If we have not found a point with larger value in previous

\(^{1}\)In the definition of \( g_{\Delta} \) the term \( \lambda + c \) is replaced by \( \Delta(\lambda) \).
evaluations, we cannot resolve the comparison. The Case 0 of Figure 4 shows this situation. Now, we copy the state of the algorithm and proceed in one copy with the presumption that $x_1$ lies to the left of $x^*$ and in the other copy with the presumption that $x_1$ lies to the right of $x^*$. These two cases are depicted as Case 1 (or Case 2 resp.) in Figure 4. So, on one side, we calculate a maximum flow for some $x_1 < x_0$. If $f(x_1) > f(x_0)$, we can resolve the comparison from above.

During the whole process, we only have two copies running at any given time. These two copies can be run in parallel, since they only need to communicate right before the next branching step in order to know which branches of the tree to cut. This enables us to prove the following result:

**Theorem 11.** The AEMFP with a piecewise polynomial concave deviation function $\Delta$ with maximum degree $q$ can be solved in polynomial time for one homologous arc set in time $O(mq \cdot (nm \cdot (n + 2m + n^2)(TMF(n, n + m))))$ under the assumption that the roots of a polynomial $p$ of maximum degree $q$ can be computed in constant time $O(1)$.

**Proof.** Since we have to compute the maximum for every interval, we have to run the algorithm $O(mq)$ times. In each interval, we run the Edmonds–Karp algorithm which has at most $O(nm)$ iterations. In each iteration, the algorithm needs to find a shortest path $P$ w.r.t. the number of arcs, which can be done in $O(n + m)$ time, for example with a breadth-first-search. To find the minimum residual capacity on $P$, the algorithm needs to do $O(n^2)$ comparisons. For updating the residual network, again $O(m)$ comparisons are needed. In order to resolve a comparison, first the roots of a polynomial $p$ of bounded degree are computed, which can be done in constant time $O(1)$ by assumption. Evaluating $F$ at a root is a maximum flow computation in the graph $D'$ where the lower bounds have been eliminated. Since this graph $D'$ has $n$ nodes and $n + m$ arcs, we write $TMF(n, n + m)$ for the time needed to compute a maximum flow in this graph. Overall, this yields in a running time of $O(mq \cdot (nm \cdot (n + 2m + n^2)(TMF(n, n + m))))$. 

In the worst case, our algorithm yields a better running time than a direct implementation of the Megiddo-Toledo algorithm for maximizing nonlinear concave function in $k$ dimensions, which runs in $O((TMF(n, m))^2)$ (cf. [15]). The integral version of the concave AEMFP turns out to be still hard to solve and hard to approximate.

**Theorem 12.** The integer concave AEMFP is $\mathcal{NP}$-complete.

**Proof.** The proof is similar to the proof of Theorem 1 and thus omitted.

**Theorem 13.** Moreover, unless $P = \mathcal{NP}$, there is no polynomial time constant factor approximation algorithm for the integer concave AEMFP.

**Proof.** For this, we use the same reduction as in Theorem 2. For the concave deviation function of the set $R_b$, we choose $\Delta^k_b : x \mapsto kx$. Then, one can see that the maximum flow value is 0 if no solution of X3C exists and $\frac{q}{2} + k$ otherwise. Thus, unless $P = \mathcal{NP}$, no polynomial time constant factor approximation algorithm can exist.

## 5 CONCLUSION

In this article, we considered a new class of flow problems, which we call almost equal flow problem. These are to be understood as a generalization of the equal flow problem. We proved that the problem of finding such an optimal integer flow turns out to
be hard in general, regardless of whether the function is given by an affine transformation, a concave or a convex function. Furthermore, even finding an optimal maximum fractional flow for a convex deviation function is \( \mathcal{NP} \)-hard to find. Nevertheless, using a parametric search approach, we provide strongly polynomial algorithms when the number of homologous sets is given by a constant and the deviation function is either a linear or a concave function. Here we provide the first complexity results for the AEMFP and several variants. For the AEFMP with \( k \) homologous arc sets and affine deviation functions, we obtain a running time of \( \mathcal{O}(n^k m^k \log(\log(n))^k T_{mf}(n, n + m)) \), where \( T_{mf}(n, m) \) is the running time of a maximum flow algorithm on a directed graph \( D \) with \( n \) nodes and \( m \) arcs. As noted above, general polynomial time solvability of the AEMFP in the case of affine deviation functions also follows from Tardos’ algorithm (cf. [14]). Our main algorithmic contribution is a combinatorial method that works not only in the affine deviation case, but also for concave functions.

While equal flow problems have been studied extensively in the literature, we believe that the generalization to almost equal flows creates new modeling opportunities and application areas. For this purpose, a future research direction is to develop algorithms for special graph classes, such as series-parallel graphs. Furthermore, the concept of an almost-equal property on certain subsets of variables is not only applicable in the context of maximum flows, but in general as an additional constraint of combinatorial optimization problems. For instance, one could consider the integer knapsack problem where the number of times items in certain subsets of the ground set may be chosen are restricted to be “almost equal.”

ACKNOWLEDGMENTS

We are grateful to the anonymous reviewers for their valuable feedback and constructive criticism, which significantly improved the quality of this research article. Open Access funding enabled and organized by Projekt DEAL.

DATA AVAILABILITY STATEMENT

Data sharing not applicable to this article as no datasets were generated or analyzed during the current study.

ORCID

Till Heller © https://orcid.org/0000-0002-8227-9353

REFERENCES

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, Network flows, Alfred P. Sloan School of Management, Cambridge, MA, 1988.
[2] R. K. Ahuja, J. B. Orlin, G. M. Sechi, and P. Zuddas, Algorithms for the simple equal flow problem, Manag. Sci. 45 (1999), no. 10, 1440–1455.
[3] A. I. Ali, J. Kennington, and B. Shetty, The equal flow problem, Eur. J. Oper. Res. 36 (1988), no. 1, 107–115.
[4] C. Büsing, A. M. Koster, and S. Schmitz, Robust minimum cost flow problem under consistent flow constraints, Ann. Oper. Res. 312 (2022), no. 2, 691–722.
[5] H. I. Calvete, Network simplex algorithm for the general equal flow problem, Eur. J. Oper. Res. 150 (2003), no. 3, 585–600.
[6] M. R. Garey and D. S. Johnson, Computers and intractability, Vol 174, Freeman, San Francisco, CA, 1979.
[7] R. Haese, Almost equal flow problems, Master thesis, TU Kaiserslautern, 2019.
[8] R. Hassin and R. Poznanski, Integrality in the multinetwrok min-cost equal-flow problem, Networks 80 (2022), no. 3, 267–273.
[9] A. Manca, G. M. Sechi, and P. Zuddas, Water supply network optimisation using equal flow algorithms, Water Resour. Manag. 24 (2010), 3665–3678.
[10] N. Megiddo, “Combinatorial optimization with rational objective functions,” Proceedings of the Tenth Annual ACM Symposium on Theory of Computing, ACM, New York, 1978, pp. 1–12.
[11] N. Megiddo, “Applying parallel computation algorithms in the design of serial algorithms,” 22nd Annual Symposium on Foundations of Computer Science (SFCS 1981), IEEE, New York, 1981, pp. 399–408.
[12] C. A. Meyers and A. S. Schulz, Integer equal flows, Oper. Res. Lett. 37 (2009), no. 4, 245–249.
[13] K. Srinathan, P. R. Goundan, M. Kumar, R. Nandakumar, and C. P. Rangan, “Theory of equal-flows in networks,” International Computing and Combinatorics Conference, O. H. Ibarra and L. Zhang (eds.), Springer, Berlin, 2002, pp. 514–524.
[14] E. Tardos, A strongly polynomial algorithm to solve combinatorial linear programs, Oper. Res. 34 (1986), no. 2, 250–256.
[15] S. Toledo, “Maximizing non-linear concave functions in fixed dimension,” Complexity in numerical optimization, World Scientific, Singapore, 1993, pp. 429–447.

How to cite this article: R. Haese, T. Heller, and S. O. Krumke, Algorithms and complexity for the almost equal maximum flow problem, Networks. 83 (2024), 642–652. https://doi.org/10.1002/net.22209