The Budgeted Minimum Cost Flow Problem with Unit Upgrading Cost

Christina Büsing* Sarah Kirchner† Arie Koster* Annika Thome†

October 6, 2015

Abstract

The budgeted minimum cost flow problem (BMCF(K)) with unit upgrading costs extends the classical minimum cost flow problem by allowing to reduce the cost of at most K arcs. In this paper, we consider complexity and algorithms for the special case of an uncapacitated network with just one source. By a reduction from 3-SAT we prove strong $\mathcal{NP}$-completeness and inapproximability, even on directed acyclic graphs.

On the positive side, we identify three polynomial solvable cases: on arborescences, on so-called tree-like graphs, and on instances with a constant number of sinks. Furthermore, we develop dynamic programs with pseudo-polynomial running time for the BMCF(K) problem on (directed) series-parallel graphs and (directed) graphs of bounded treewidth.

Keywords: Minimum cost flow, bilinear problem, budgeted optimization, complexity

1 Introduction

Motivation In this paper we present a bilinear minimum cost flow optimization problem which is an extension of the minimum cost flow (MCF) problem. Consider a traffic network with users travelling through the network and an operator (e.g., government, railway company) who wants to maintain and/or improve the infrastructure/connections. Both the users and the operator are interested in minimizing the total travelling time of the users. The route a user chooses is obviously affected by the improvements the operator selects to reduce the travelling time. The operator has limited means to do so. For example, adding another lane

*RWTH Aachen University, Lehrstuhl II für Mathematik, Pontdriesch 14-16, D-52062 Aachen, Germany, {buesing, koster}@math2.rwth-aachen.de

†RWTH Aachen University, Lehrstuhl für Operations Research, Kackertstraße 7, D-52072 Aachen, Germany, {kirchner, thome}@or.rwth-aachen.de

to a highway results in shorter traveling times. The limited budget for such infrastructure measurements, however, prohibits the operator from an unlimited number of improvements.

The collective choice of the users to take the fastest route through the network can be modelled as a classical minimum cost flow problem, where one unit of flow corresponds to one user, the nodes represent cities and the arcs road sections between cities. The cost on each arc is the time one needs to travel the corresponding road. The additional choice of the operator to upgrade a limited number of road sections to reduce travel times is incorporated by allowing a lower cost, i.e. a faster travel time, to be applied to the flow on a limited number of arcs. This means that each arc is associated with a second cost which represents the travel time after the upgrade on this section. This extension to the MCF problem results in an $\mathcal{NP}$-hard problem.

More formally, we are given a directed graph with several sources and sinks each having a supply or demand respectively. This graph represents the infrastructure. The sources and sinks represent start and destination points of the users. Furthermore, each arc is associated with a capacity, a regular and lower cost, where the regular cost is the travel time before the upgrade and the lower cost after the upgrade. The goal of the operator is finding a selection of at most K arcs such that the cost of the minimum cost flow problem after upgrading these arcs is minimized. We call this problem the budgeted minimum cost flow problem with unit upgrading costs (BMCF(K)) where K denotes the maximum number of arcs to be upgraded.

Related Work There are several extensions to the MCF problem considered in literature. In this short overview we focus on the extensions that are closely related to our problem. Coene et al. [2013] and Maya Duque et al. [2013] introduce the Budget Constrained Minimum Cost Flow (BC-MCF) problem and as a special case the accessibility arc upgrading problem (AAUP). In the BC-MCF problem, besides the flow dependent cost, there is a second cost associated with each arc that occurs if flow is sent via this arc. In the AAUP the vertex set is divided into regular vertices and centres. The arcs connecting the vertices are associated with several upgrading levels and corresponding upgrading costs. An upgrade on an arc reduces the travel cost on this arc. There is a budget for upgrading arcs in the network. The goal is to find an upgrading level for every arc in the network such that the total cost to connect all regular vertices to the nearest center is minimized and that the budget is not exceeded by the selected upgrading strategy. They show that the problem is weakly $\mathcal{NP}$-hard by a reduction from the knapsack problem. This result holds even if the problem is restricted to star graphs. Our problem can be seen as a special case of the AAUP where there is only one upgrading level associated with each arc and unit upgrading costs. We prove that even this special case of the problem is strongly $\mathcal{NP}$-complete by a reduction from 3-SAT. But when

restricted to star graphs or even trees we propose a polynomial time algorithm to solve the BMCF(K) problem. Both results extent the complexity results of Maya Duque et al. [2013].

Demgensky et al. [2002] propose a flow cost lowering problem (FCLP). In the FCLP there is a single source and a single sink. A specified demand has to be sent from the source to the sink through the given network. Every arc in the network is associated with a cost and a capacity. The cost on the arcs can be lowered up to a minimal cost. To do so a price has to be paid to receive a predetermined discount on the cost of an arc. It is possible to buy more than one discount per arc. The amount invested in upgrading arcs is bounded by a budget. The goal is to find an upgrading strategy and a routing for the flow through the network such that the flow cost is minimized. The authors show the non-approximability of the problem on series-parallel and on bipartite graphs. However, they give a pseudo-polynomial time approximation algorithm for the FCLP on series-parallel graphs. The arc upgrading policies are more restricted in our setting as we consider one arc upgrading level only and unit upgrading costs. Our problem is polynomially solvable in the case of a single source and a single sink. We therefore focus on problems with more than one sink.

Holzhauser et al. [2015] describe a minimum cost flow problem with the extension of a second cost per arc. This second cost applies as soon as there is positive flow on the arc. It can be interpreted as increasing the capacity by one unit. The capacity can be increased up to a maximum. The cost invested on increasing capacities must meet a budget constraint. The authors investigate the cases where the capacity can be increased continuously, in integral steps or either to its maximum or not at all. For the third case, the authors prove strongly $\mathcal{NP}$-hardness. This case is related to our problem in the sense that in the BMCF(K) only one upgrade/increase is possible per arc. However, in the BMCF(K) problem the upgrade reduces the cost on an arc and does not increase the capacity. Also, the upgrading cost is independent of the benefit.

Schwarz and Krumke [1998] introduce the budget-constrained flow improvement problem of increasing capacities on arcs such that the flow from a source to a sink is maximized. They show that this problem can be solved in polynomial time, but becomes $\mathcal{NP}$-hard on series-parallel graphs if for each arc the decision is between increasing the capacity to its maximum or no increase at all. With this restriction on possible increases the problem is strongly $\mathcal{NP}$-hard on bipartite graphs. The authors propose an FPTAS for series-parallel graphs.

Ahuja and Orlin [1995] define the constrained maximum flow problem as the problem to send as much flow as possible from a source to a sink such that the cost of the flow does not exceed a given budget. They investigate two different cost structures: a linear and a convex cost function of the amount of flow. For both functions the authors propose weakly

polynomial algorithms that are modifications of the capacity scaling algorithms for the MCF.

Contribution To the best of our knowledge, the particular setting of the budgeted minimum cost flow problem has not been studied in the literature before. We show that the BMCF(K) problem is already strongly $\mathcal{NP}$-complete and (by the reduction) inapproximable if we restrict to a single source and directed acyclic graphs. But, if restricted to directed trees (arborescenses) or tree-like graphs, the BMCF(K) problem is solvable in polynomial time. As star graphs are a special case of trees, this extends the complexity results of Maya Duque et al. [2013].

In case capacities on the arcs are not restrictive, we investigate the structure of optimal solutions. This structure is exploited to obtain polynomial time algorithms for so-called tree-like graphs. Although the problem at hand is strongly $\mathcal{NP}$-complete in general, we further propose pseudo-polynomial algorithms to solve the BMCF(K) problem on (directed) series-parallel graphs. We extend this result to the more general graph class of graphs of bounded tree-width.

Outline In Section 2, we formally define BMCF(K) and show that it is strongly $\mathcal{NP}$-complete. Sections 3 and 4 investigate, respectively, polynomial and pseudopolynomial solvable cases of the problem. We conclude the paper with an outlook in Section 5.

2 Problem Definition and Complexity

In this section we give a mathematical problem description and prove that the considered problem is $\mathcal{NP}$-complete.

2.1 Problem Definition

The budgeted minimum cost flow problem with unit upgrading costs (BMCF(K)) extends the classical minimum cost flow (MCF) problem by allowing to change the cost of at most K arcs. In the BMCF(K) problem, we consider instances defined by a directed graph $G = (V, A)$ where V is the set of vertices and A is the set of arcs. Each vertex $v \in V$ is associated with a number $b(v) \in \mathbb{Z}$. We call any $v \in V$ with $b(v) < 0$ a *demand vertex* and the corresponding value $b(v)$ the *demand* of vertex v. Equivalently, we call any $v \in V$ with $b(v) > 0$ a *supply vertex* and the corresponding value $b(v)$ the *supply* of vertex v. If $b(v) = 0$, then v is a *transshipment vertex*. W.l.o.g. we assume that the total supply equals the total demand. Each arc $a \in A$ is associated with a cost per unit of flow, denoted by $\bar{c}(a)$ and a capacity denoted by $u(a) \in \mathbb{N}^+$. The classical MCF problem deals with finding a minimum cost flow

that satisfies the demand of all demand vertices and does not exceed the capacity on any arc. This flow is also called a *b-flow*. For more insight on the MCF problem we refer the reader to Ahuja et al. [1993].

In addition to an MCF instance, an instance of the BMCF(K) problem contains a budget $K \in \mathbb{N}$ and a second cost per unit of flow for each arc, denoted by $\underline{c}(a)$. We refer to this second cost as the *lower cost* of an arc, since we assume $\underline{c}(a) \leq \bar{c}(a)$ for all $a \in A$. The budgeted minimum cost flow problem with unit upgrading cost (BMCF(K)) is to find a b-flow with minimum cost, where we are allowed to pay the lower cost on at most K arcs. More formally, the problem is defined as follows.

Definition 1 (**The BMCF(K) Problem**). *Let $G = (V, A)$ be a digraph. Let $b(v) \in \mathbb{Z}$ be demand/supply of any $v \in V$, $\bar{c}(a)$, $\underline{c}(a)$ be upper and lower costs for all $a \in A$ such that there are no negative cost cycles, and $u(a)$ be capacities for all $a \in A$. Finally, let $K \in \mathbb{N}$ be the budget parameter. A solution to the BMCF(K) problem is a b–flow f^* and a set $A_K^* \subseteq A$ with $|A_K^*| \leq K$. The objective is to find such a pair (f^*, A_K^*) that minimizes the cost function*

$$c(f^*, A_K^*) = \sum_{a \in A_K^*} \underline{c}(a) \cdot f^*(a) + \sum_{a \in A \setminus A_K^*} \bar{c}(a) \cdot f^*(a)$$

The parameter K can be interpreted as a budget used for upgrading arcs and the cost of the upgrade is one per arc. Also note that for $K = 0$ or $K = |A|$ the BMCF(K) problem is the standard MCF problem.

In this paper we focus on the special case of unlimited capacities and exactly one source. Note that in this case we can assume that G is a simple graph. When referring to the BMCF(K) problem, we therefore refer to the uncapacitated single source BMCF(K) problem. Furthermore, n denotes the number of vertices and m the number of arcs in the considered graph.

2.2 Complexity

The classical minimum cost flow problem is solvable in polynomial time (e.g. Orlin [1993]). Coene et al. [2013] already proved that the budgeted minimum cost flow problem is weakly $\mathcal{NP}$-hard by a reduction from the knapsack problem, if the upgrading costs are not uniform. We strengthen the result by showing that even if the upgrading cost are 1 for every arc, the problem is strongly $\mathcal{NP}$-hard.

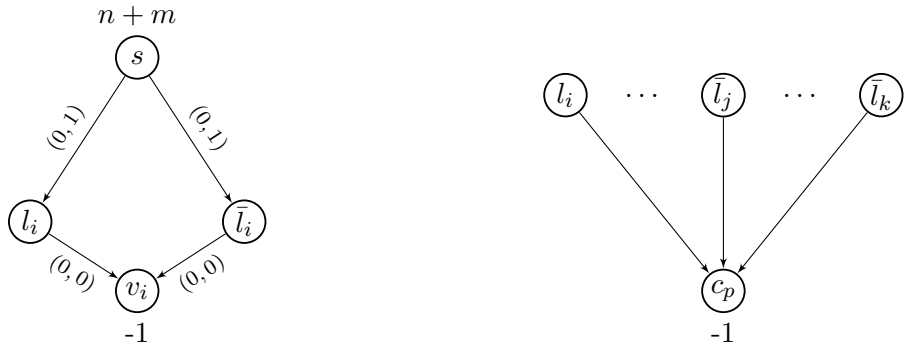
Theorem 2. *The decision version of the BMCF(K) problem is strongly $\mathcal{NP}$–complete.*

Proof. We prove the $\mathcal{NP}$–hardness with a reduction from 3-SAT. The 3-SAT problem consists of a finite set X of n boolean variables $X = \{x_1, \ldots, x_n\}$ and a collection of m clauses

$C = \{C_1, \ldots, C_m\}$. If x_i is a variable in X, then x_i and $\bar{x}_i$ are literals over X. Each clause consists of three literals, i.e. $C_j = \{y_{j1} \vee y_{j2} \vee y_{j3}\}$ where $y_{jk} \in \{x_1, \bar{x}_1, \ldots, x_n, \bar{x}_n\}$ for $k \in \{1, 2, 3\}$. The 3-SAT decision problem is to determine whether there is an assignment of x that satisfies all clauses.

Given such a 3-SAT instance I, we create a corresponding instance I' of the BMCF(K) problem. The graph $G = (V, A)$ in I' has one supply vertex s with $b(s) = n+m$, one vertex v_i for each variable $x_i \in X$ with $b(v_i) = -1$, vertices ℓ_i and $\bar{\ell}_i$ for each literal over X with $b(\ell_i) = 0$ and $b(\bar{\ell}_i) = 0$ and one vertex c_j for each clause $C_j \in C$ with $b(c_j) = -1$. We refer to these vertices as variable–vertices, literal–vertices, and clause–vertices respectively. Note that there are $1 + n + 2n + m$ vertices. There is one arc $a \in A$ connecting s to each literal–vertex ℓ_i and $\bar{\ell}_i$ for all $i \in \{1, 2, \ldots, n\}$ with an upper cost $\bar{c}(a) = 1$ and a lower cost $\underline{c}(a) = 0$. Furthermore, there are arcs connecting the literal–vertices with the corresponding variable–vertex, i.e. (ℓ_i, v_i) and $(\bar{\ell}_i, v_i)$, each with $\underline{c}(a) = \bar{c}(a) = 0$. We refer to Figure 1a) for clarification.

Finally, if $x_i \in C_j$ (respectively $\bar{x}_i \in C_j$), then G contains the arc $a = (\ell_i, c_j)$ (respectively $a = (\bar{\ell}_i, c_j)$) with $\underline{c}(a) = \bar{c}(a) = 0$. We refer to Figure 1b) for an example.



(a) Example for variable x_i

(b) Example for $C_p = \{x_i \vee \bar{x}_j \vee \bar{x}_k\}$

Figure 1: Construction of the graph

Note that in the case where $\underline{c}(a) = \bar{c}(a)$ we speak of 'cost of arc a' while making no distinction between lower and upper cost. The budget K is set to n. The entire reduction graph is depicted in Figure 2.

We prove that I is a yes–instance for the 3-SAT problem if and only if there is a solution for I' of the BMCF(K) problem with cost 0.

Let I be a yes–instance and let x^* be a corresponding truth assignment. Then we send one unit of flow from s to each variable–vertex v_i via the literal–vertex ℓ_i if x_i is true and $\bar{\ell}_i$ if x_i is false. Furthermore, assume w.l.o.g. that $y_{j1} \in C_j$ with $y_{j1} = \bar{x}_i^*$ satisfies C_j. Then

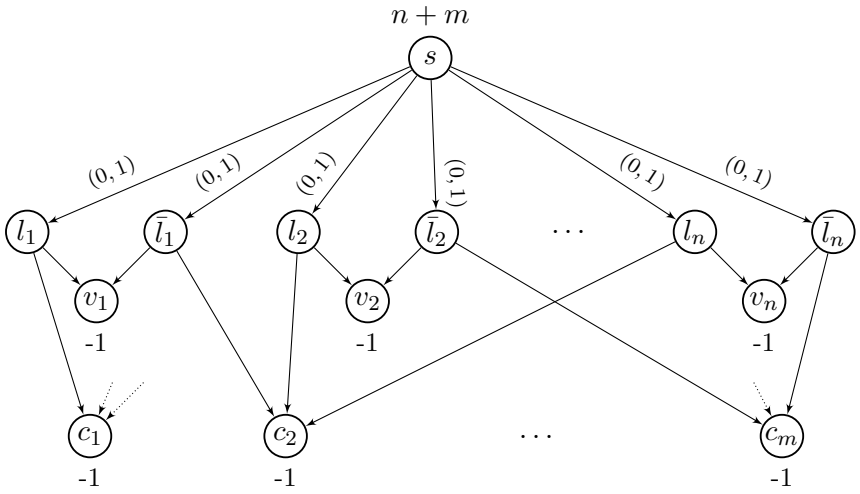


Figure 2: Graph of 3SAT instance as BMCF(K) problem

we send one unit of flow from s to c_j via the path $\{s, \bar{l}_i, c_i\}$. The case $y_{j1} = x_i^*$ is handled analogously.

Finally, we assign lower costs to the arc (s, ℓ_i) if x_i^* is true for all $i \in \{1, 2, \ldots, n\}$ while the arc $(s, \bar{\ell}_i)$ is associated with the upper cost. Similarly, if x_i^* is false, we assign lower costs to the arc $(s, \bar{\ell}_i)$ while the arc (s, ℓ_i) is associated with the upper cost.

It is easy to see that the described flow satisfies all demands. Since we send flow only along arcs that are associated with a cost of 0, the total cost of the flow is also 0. Thus the BMCF(K) problem has a solution with cost 0. Note that the occurring cost is never lower than 0.

Now we prove that if there is a solution for I' of the BMCF(K) problem with cost 0, then I is a yes–instance for the 3-SAT problem.

Let there be a solution for I' of the BMCF(K) problem with cost 0 and let f^* be a corresponding optimal flow in G. Furthermore, let A_K^* be the set containing the arcs associated with their lower cost. Since the total cost is 0, flow can only be sent along an arc that is associated with a cost of 0. Hence, if $f(a) > 0$ for any $a \in A$, then either $\underline{c}(a) = \bar{c}(a) = 0$ or $a \in A_K^*$. Since each variable–vertex can only be reached via a path that either contains $a = (s, \ell_i)$ or $a' = (s, \bar{\ell}_i)$, it holds that $a \in A_K^*$ or $a' \in A_K^*$ for all variable–vertices v_i. Since there are n variable–vertices and $K = n$, it holds that $a \in A_K^*$ if and only if $a' \notin A_K^*$. This accounts for n units of flow (satisfying the demand for all v_i) and also completely describes A_K^* (the latter because $|A_K^*| \leq K$).

The flow satisfying the demand of each c_j is then sent via an arc $a \in A_K^*$ to a literal–vertex and then to c_j. Such a path must exist as we assumed there is a solution to the problem with

total cost 0.

This flow can now be interpreted as a yes–instance for the 3-SAT problem in the following way:

If $a \in A_K^*$ for $a = (s, \ell_i)$ or $a = (s, \bar{\ell}_i)$, then $x_i =$ true or $x_i =$ false, respectively. This is a truth assignment as the flow satisfying the demand of each c_j – w.l.o.g. assume it is sent via $\{s, \ell_i, c_j\}$ – corresponds to the fact that clause C_j is satisfied by the assignment of ℓ_i of the variable $x_i =$ true. This holds for all $j \in \{1, 2, \ldots, m\}$.

Since we can verify that a flow f and an arc set A^* is a feasible solution and compute the corresponding costs in polynomial time, this concludes the proof. □

Note, that the reduction implies that there is no approximation algorithm for the BMCF(K) problem. Furthermore, the cost structure of the graph in the reduction is quite simple. In order to find polynomial solvable cases a restriction to special cost structures seems unwise. We will therefore consider in the following sections the BMCF(K) problem on special graph classes and for the case when the number of demand vertices is constant.

3 Polynomially solvable Cases

In the following section we discuss polynomial solvable special cases of the BMCF(K) problem. In particular, the BMCF(K) problem on graphs with a constant number of demand vertices, on directed trees (arborescences) and on so called tree-like graphs. Note that the BMCF(K) problem with one demand vertex can easily be transformed into a weight constrained shortest path problem. This problem is known to be polynomial solvable, [see Aneja and Nair, 1978]. Hence, the single demand vertex BMCF(K) problem is polynomially solvable. Note that the BMCF(K) problem with multiple demand vertices is equivalent to a BCMF(K) problem with one demand vertex and multiple supply vertices. We just need to replace each arc (u, v) by its anti-parallel arc (v, u) and transform each supply vertex to a demand vertex and vice versa.

3.1 The BMCF(K) on Arborescences and tree-like Graphs

Arborescences present a simple graph structure and have the nice property that there exists a unique path between the root/supply node and all other vertices. As a consequence the b–flow of an MCF instance on an arborescence is unique. We will use that property and show that the BMCF(K) problem is solvable in polynomial time.

Theorem 3. *The BMCF(K) problem is solvable in polynomial time on an arborescence.*

Proof. First, we solve the classical minimum cost flow problem on an arborescence T. If the MCF problem has a solution for the given graph, so does the BMCF(K) problem. Further-

more, the optimal flow f^* is unique as T is an arborescence. Since the flow is unique, we know how many units of flow are sent via each arc. This flow is independent of A_K^*. Therefore, we calculate a score for each arc a defined as $\big(\bar{c}(a) - \underline{c}(a)\big) \cdot f^*(a)$. This score is the amount that the objective value decreases if $a \in A_K^*$. We then order the arcs non-increasingly according to this score and choose the first K arcs to be upgraded, and hence to be in A_K^*. The flow f^* is optimal as it is unique. The set A_K^* is optimal as we have chosen for those arcs to be in A_K^* which decrease the objective value the most. It is easy to see that all steps taken are achieved in time polynomial in the input size. □

The above result can be easily generalized to multiple sources on (bi)directed trees. Next, we extend the class of arborescences and present so-called tree-like graphs. Tree-like graphs are arborescences on a macro perspective whereas zooming into one arc we see some general graph. More formally, we define tree-like graphs as follows.

Definition 4. *A graph $G = (V, A)$ is a tree-like graph if it can be obtained by the following procedure.*

1. *Let $T = (V, A)$ be an arborescence, where V is the set of arborescence-vertices and A the set of arborescence-arcs with $|A| = m_T$. Let the $G_i = (V_i, A_i)$ be a digraph with two distinguished vertices $v_{i,1}$ and $v_{i,2}$ for all $i \in \{1, 2, \ldots, m_T\}$.*
2. *Each arc $a_i = (u, v) \in A$ with $i \in \{1, 2, \ldots, m_T\}$ is replaced with G_i by identifying u with $v_{i,1}$ and v with $v_{i,2}$. We call G_i the arc-graph of arc a_i*
3. *The new graph now defines G.*

The idea for solving the BMCF(K) problem on arborescences can be extended to tree-like graphs, if only the arborescence-vertices are supply or demand vertices and the root is the unique supply vertex. All other vertices are transshipment vertices. We call such a placement of the demand a *arborescence-demand*. In a tree-like graph with arborescence-demand the amount of flow that needs to be send through each arborescence-arc is fixed as the flow in the underlying arborescence is unique. We now need to decide how many arcs obtain lower costs in each arc-graph. This can be done by solving the single demand vertex BMCF(k) problem for every arc-graph G_i, every budget $k \in \{0, \ldots, K\}$ and a demand d_i equal to the unique amount of flow that is sent through G_i. According to these cost values, we define again a score $c_i(k)$. This score represents the cost of d_i units of flow in G_i by using k upgrading arcs. An optimal placement of the budget K to each arc-graph can finally be computed by a special shortest path problem. Algorithm 1 presents this idea more formally.

We now show that the proposed algorithm solve the BMCF(K) problem on tree-like graphs to optimality. Furthermore, we show that it is a polynomial time algorithm.

Algorithm 1: Tree-like

input : tree-like graph G with underlying arborescence T and arborescence-demand, budget K, upper arc cost $\overline{c}$, lower arc cost $\underline{c}$

output: (f^*, A_K^*)

for $a_i \in A(T)$ **do** initialize flow d_i to 0;
for *demand vertices* t *of* G **do**
    find unique path p from s to t in underlying T;
    if $a_i \in p$ **then** $d_i \leftarrow d_i + |b(t)|$;
// computation of score $c_i(k)$
for $i = 1, \ldots, m_T$ **do**
    for $k \leq K$ **do**
        solve single demand vertex BMCF(k) problem with demand d_i on G_i;
        $c_i(k) \leftarrow c(f^*, A_k^*)$; $A_{i,k}^* \leftarrow A_k^*$; $f_{i,k} \leftarrow f^*$;
// create a grid-like graph H
Let $H = (V_H, A_H)$ be a graph with $V_H, A_H = \emptyset$;
for $i = 0, \ldots, m_T$ **do**
    for $j = 0, \ldots, K$ **do**
        $V_H \leftarrow V_H \cup \{v_{ij}\}$
        if $i > 0$ **then**
            for $l = 0, \ldots, K$ *with* $j + l \leq K$ **do**
                $a = (v_{i-1,j}, v_{i,j+l})$;
                $c(a) = c_i(l)$;
                $A_H \leftarrow A_H \cup \{a\}$;
Find shortest path p from v_{00} to $v_{m_T K}$ in H;
// obtain a solution in the original graph G
Let $f(a) = 0$ for all $a \in G(A)$, $A_K^* = \emptyset$;
for $(v_{i-1,j}, v_{i,l}) \in p$ **do**
    for $a \in G_i$ **do**
        $f^*(a) = f_{i,l-j}(a)$;
        $A_K^* \leftarrow A_K^* \cup A_{i,l-j}^*$;

Theorem 5. *Algorithm 1 solves the BMCF(K) problem on tree-like graphs to optimality and runs in time polynomial in the input size.*

Proof. An optimal solution of the BMCF(K) problem on a tree-like graph with k_i upgraded arcs in subgraph G_i is composed of optimal solutions of the single demand vertex BMCF(k_i) problem on the subgraphs G_i. Since Algorithm 1 solves the single demand vertex BMCF(k) problem on line 7 for all $k \leq K$, we only need to show that the algorithm determines an optimal allocation of K upgraded arcs to the subgraphs. Note that an arc $a = (v_{i-1,j}, v_{i,j+l})$ in the grid-like graph H as described at line 15 represents an optimal solution to the single demand vertex BMCF(l) problem on subgraph G_i. A shortest path from v_{00} to $v_{m_T K}$ containing arc $a = (v_{i-1,j}, v_{i,j+l})$ therefore represents the allocation of l upgraded arcs to the subgraph G_i when j arcs already have been upgraded in $G_0, \ldots, G_{i-1}$. Since $l + j \leq K$ in line 14, there are at most K arcs upgraded in total. Hence, the flow composition of the optimal flows of the single demand vertex problems and all upgraded arcs determine an optimal solution to the BMCF(K) problem on the original graph.

Obviously, all the steps in Algorithm 1 are polynomial in the input size. □

We now discuss the BMCF(K) problem on graphs with a constant number of demand vertices and propose a polynomial time algorithm for these instances.

3.2 Constant number of demand vertices

Now we discuss the BMCF(K) problem on graphs with a constant number of demand vertices. We start by introducing some notation followed by properties of an optimal solution of the BMCF(K) problem. Finally, we develop the polynomial algorithm for the BMCF(K) problem on graphs with a constant number of demand vertices that relies on these properties.

Let $G = (V, A)$ be a digraph and let f be a flow in G. The subgraph F that is induced by the arcs with positive flow $A(f)$ is called the *flow induced subgraph* of G. The following corollary states that we can focus our attention to solutions where the flow induces an arborescence.

Lemma 6. *There exists an optimal solution (f^*, A_K^*) to the BMCF(K) problem with the set of demand vertices L such that the flow induced subgraph is an arborescence.*

Proof. It is well known that an arc flow can be decomposed into a path and cycle flow with at most $n+m$ paths and cycles (see e.g. Ahuja et al. [1993]). As we have unlimited capacities on all arcs and G contains no negative cycles we know there exists an optimal solution with no cyclic flows and exactly one path from the supply vertex to every demand vertex. If there is more than one path in an optimal solution with positive flow between the supply vertex and

a single demand vertex then all of these paths must have the same cost. Since the capacities are unlimited, we can choose just one path and send all flow along this path. Hence, there exists an optimal solution that can be decomposed into $|L|$ paths, one from the supply vertex to every demand vertex.

It is obvious that two of those paths split up at least once. Next, we show that there exists an optimal solution in which these paths separate exactly once (meaning they are node-disjoint afterwards). Suppose they separate in more than one vertex, in a so called *separation vertex*. Then the two paths must merge at some other vertex, at a so called *merge vertex*. That means that there are two partial paths of the same cost between the first separation vertex and the merge vertex. If that is the case, all flow can be routed along one of the paths entirely using the same argument as before. Therefore, there is exactly one separation vertex for any pair of paths. This means, there exists an optimal solution where the flow induced subgraph is an arborescence. □

The concept of separation vertices mentioned in the previous lemma becomes important for the algorithm to solve the BMCF(K) problem on graphs with a constant number of demand vertices. We therefore introduce the notation of *separation vertices*: Let the separation vertices of an arborescence $T = (V, A)$ be the vertices $S \subseteq V(T)$ having out-degree greater one. We show now that the number of separation vertices in the flow induced subgraph of an optimal solution is bounded.

Corollary 7. *There exists an optimal solution (f^*, A_K^*) to the BMCF(K) problem with the set of demand vertices L such that the flow induced subgraph is an arborescence with at most $|L| - 1$ separation vertices.*

Proof. From Lemma 6 we know that there exists an optimal solution to the BMCF(K) problem such that the flow induced subgraph is an arborescence. As there a $|L|$ demand vertices the arborescence has at most $|L|$ leaves. Thus paths are separated at most $|L| - 1$ times. □

Later on, the algorithm considers arborescences whose vertices are either its root, leaves or separation vertices. Note that we can assume that the demand vertices correspond to leaves in the flow induced arborescence and vice versa: for each demand vertex d that is not a leaf, introduce a dummy demand vertex d' with $b(d') = b(d)$, add the arc (d, d') and set $b(d) = 0$. We call these arborescences *separation arborescences*.

Definition 8 (Separation Arborescence)**.** *Let $T = (V, A)$ be an arborescence with root s, leaves $L \subseteq V(T)$ and separation vertices S. The arborescence T' is called the separation arborescence of T if it is obtained via the following contraction procedure. Contract two*

vertices $u, v \in V(T)$ *if* $(u, v) \in A(T)$ *and at most one of them is in* $\{s\} \cup L \cup S$*. Hence,* $V(T') = \{s\} \cup L \cup S$*.*

Given an optimal solution to the BMCF(K) problem whose flow induced subgraph is an arborescence, we show that this optimal solution is composed of solutions to certain single demand vertex BMCF(k_a) problems.

Theorem 9. *Let* (f^*, A_K^*) *be an optimal solution to the BMCF(K) problem on* G *such that the flow induced subgraph* F *is an arborescence. Let* $T = (V(T), A(T))$ *be the separation arborescence of* F*. For all* $a = (v_1, v_2) \in A(T)$*, let* f_a^* *be the amount of flow sent from* v_1 *to* v_2 *in* F *as specified by* f^**. Furthermore, let* $A_{a,K}^*$ *where* $|A_{a,K}^*| = k_a$ *be the set of upgraded arcs in path* $p \subseteq A(F)$ *with start vertex* v_1 *and end vertex* v_2*. Then it holds that*

$$c(f^*, A_K^*) = \sum_{a \in A(T)} c_a(f^*, A_{a,K}^*)$$

where $c_a(f^*, A_{a,K}^*)$ *is the optimal solution value of the single demand vertex BMCF(k_a) problem on* G *with supply vertex* v_1*, demand vertex* v_2*,* $b(v_1) = f_a^*$ *and* $b(v_2) = -f_a^*$*.*

Proof. The partial flow between v_1 and v_2 in F with upgraded arc set $A_{a,K}^*$ is a feasible solution to the single demand vertex BMCF(k_a) problem as specified in the corollary. Suppose it is not an optimal solution to the single demand vertex BMCF(k_a) problem, then there exists a different flow and set of upgraded arcs that is optimal to the single demand vertex problem. By replacing the partial flow between v_1 and v_2 with this solution, one can improve the solution value of the BMCF(K) problem on the original graph. This contradicts the the choice of (f^*, A_K^*) as an optimal solution. □

Next we present an algorithm that solves the BMCF(K) problem and runs in polynomial time if the number of demand vertices is constant. The idea of the algorithm is based on the previous results. According to them, we know that there is an optimal solution where the flow induced subgraph is an arborescence. In order to find such an arborescence induced by optimal flow, we enumerate over all possible sets of separation vertices and all possible spanning trees T on these sets. The cost of an arborescence $c(T)$ can be computed by constructing a tree-like graph as described in Definition 4 and applying Algorithm 1. The entire procedure is described in Algorithm 2.

Theorem 10. *Algorithm 2 yields a feasible solution to the BMCF(K) problem on graphs with a constant number of demand vertices* $|L|$*.*

Proof. It is obvious that $|A_{G,K}^*| \leq K$. It is also easy to see that all demands are satisfied. It remains to show that the flow is feasible, meaning that the flow conservation constraints are

Algorithm 2: Constant number of demand vertices

input : graph G with a constant number of demand vertices $|L|$

output: $(f^*_G, A^*_{G,K})$

set $c(f^*_G, A^*_{G,K}) = \infty$;
for $S \subseteq V$ *with* $|S| \leq |L| - 1$ **do**
  for *spanning trees* T^s *on* $V(T^s) = \{s\} \cup L \cup S$ *where* S *is the set of separation vertices of* T^s **do**
    `// construct tree-like graph`
    for *arcs* $a_i = (v_1, v_2)$ *in* $A(T^s)$ **do**
      let $G_i = G$ with supply vertex $s = v_1$ and demand vertex $t = v_2$;
      replace a_i with G_i;
    apply Algorithm 1 and obtain solution (f^*, A^*_K);
    `// obtain a solution in the original graph` G
    for $a \in A(G_i)$ **do** initialize flow $f_{T^s}(a)$ to 0;
    forall the G_i **do**
      for $a \in A(G_i)$ **do**
        $f_{T^s}(a) \leftarrow f_{T^s}(a) + f^*(a)$;
        if $a \in A^*_K$ **then** $A^*_{T^s,K} \leftarrow A^*_{T^s,K} \cup \{a\}$;
    if $c(f_{T^s}, A^*_{T^s,K}) < c(f_G, A^*_{G,K})$ **then**
      `// update current best solution`
      $f^*_G = f_{T^s}$;
      $A^*_{G,K} = A^*_{T^s,K}$;

met. Since flow conversation constraints are fulfilled in each G_i, they are still fulfilled after the flow composition step line 11. □

We now show that Algorithm 2 always obtains an optimal solution to the BMCF(K) problem and that it runs in polynomial time when applied to graphs with a constant number of demand vertices.

Theorem 11. *Algorithm 2 solves the BMCF(K) problem on graphs with a constant number of demand vertices $|L|$ to optimality and runs in time polynomial in the input size.*

Proof. Corollary 7 yields that there exists an optimal solution where the flow induced subgraph is an arborescence with at most $|L| - 1$ separation vertices. Algorithm 2 enumerates over all possible sets of separation vertices and possible spanning trees. Hence, at some point Algorithm 2 considers the set of separation vertices and the spanning tree of an optimal solution to the BMCF(K) problem. We claim that the solution value $c(ALG)$ of Algorithm 2 is the same as the solution value $c(OPT)$ of an optimal solution.

Based on Theorem 9, we know that for an optimal separation arborescence T there exists k_a for $a \in A(T)$ such that $\sum_{a \in A(T)} k_a = K$ and it holds that

$$c(OPT) = \sum_{a=(v_1,v_2) \in A(T)} c_a(f^*, A^*_K)$$

where $c_a(f^*, A^*_{k_a})$ is an optimal solution value of the single demand vertex BMCF(k_a) problem on G with supply vertex v_1, demand vertex v_2, and $b(v_1) = f^*_a$ and $b(v_2) = -f^*_a$ where f^*_a is the flow sent from v_1 to v_2 in T. Since solving the single demand vertex BMCF(K) problem in line 7 of Algorithm 1 (which is called in line 7 of Algorithm 2) is being done at some point with $K = k_a$, we know that $c(ALG) = \sum_{a=(v_1,v_2) \in A(T)} c_a(f^*, A^*_K)$ and hence $c(ALG) = c(OPT)$.

Next, we show that Algorithm 2 runs in time polynomial in the input size. We enumerate over all subsets of cardinality $|L| - 1$ in line 2. The number of these subsets is $\sum_{k=1}^{|L|-1} \binom{n}{k} \leq (|L|-1) \cdot n^{|L|-1}$, which is polynomial in the input as $|L|$ is constant. Afterwards we enumerate over all spanning trees on these subsets. The number of spanning trees is bounded by $|L|^{|L|-2}$ as shown in Shor [1995]. All other steps taken in Algorithm 2 are clearly polynomial in the input. □

4 Pseudo-polynomially solvable Cases

In this section, we present two pseudo-polynomial cases: series-parallel graphs and more general, graphs with bounded tree-width.

4.1 The BMCF(K) on series-parallel Graphs

In this section we propose a pseudo-polynomial dynamic program (DP) that solves the BMCF(K) problem on series-parallel digraphs. We start with a formal definition of directed series-parallel (SP) graphs (similar to Kikuno et al. [1983]) and then describe the DP.

Definition 12. *Series-parallel digraph*

1. *An arc $a = (s, t)$ is a SP-digraph with a source s and a target t.*
2. *Any digraph that is obtained by either a series operation or a parallel operation of two SP-digraphs G_1 and G_2 is itself an SP-digraph. A series operation describes the contraction of t_1 and s_2. The source of G is then s_1 and the target is t_2. A parallel operation describes the contraction of both s_1 and s_2 (becoming s) and t_1 and t_2 (becoming t). The source of the newly obtained series-parallel digraph G is s and the target is t.*
3. *Digraphs that are obtained by a finite number of series or parallel operations are SP-digraphs.*

As described in Valdes et al. [1979], given any SP-digraph G an SP-tree T can be constructed that represents the order of series/parallel operations to obtain G. The SP-tree T is a binary tree whose nodes are associated with subgraphs of G. The leaves of T are associated with arcs, whereas all inner nodes of T indicate whether the graphs of the child nodes were conjoined by a series or parallel operation. To each inner node v_i, the subgraph G_i is associated that can be obtained by the operations of the subtree rooted at v_i. The root of T is associated with G. The nodes of T are indexed increasingly such that the root node is v_0.

Let us now consider a BMCF(K) problem on an SP-graph G with a given SP-tree T. We assume w.l.o.g. that the supply vertex corresponds to the source of G: SP-digraphs are acyclic and thus vertices before the supply vertex cannot be reached and can be left out. Let h_i denote the demand in G_i excluding the demand of t_i for each node v_i of T. We call this demand the *inner demand* of G_i. The DP runs on the SP tree T of G. It takes advantage of the following facts: If two subgraphs of G, say G_1 and G_2 are contracted by a series operation, the flow that is present in G_2 must have been sent through G_1 first as there is no other way for the flow to reach G_2 later on. The other feature of a minimum cost flow in a series parallel graph is that if G_1 and G_2 are connected by a parallel operation, then all flow arriving at the target of G is either completely sent through G_1 or completely through G_2. This is valid because any flow of an optimal solution induces a tree (see Corollary 7).

We will use these two properties to define the DP. Let us therefore introduce a cost tuple $c_i(k, f, d)$ with which each node v_i of T is associated. The parameter $k \leq K$ indicates the

number of upgraded arcs in the current subgraph G_i. The value f is the amount of flow exceeding the total demand of G_i that will eventually be passed on to another subgraph of G after a future series operation. Therefore, only cost tuples are considered where f is smaller or equal to the total demand of G, i.e. $f \leq \sum_{v \in V \setminus \{s\}} d(v)$. The value d indicates whether (possible) demand of the current target t_i is met, hence $d \in \{0, |b(t)|\}$. This means especially the inner demand h_i is satisfied. We need this case distinction, since in a parallel operation, the demand of vertex t_i will just be satisfied by flow through one subgraph. However, the inner demand must still be met for the other graph.

Let us now describe the DP and the upgrading procedures of the labels more precisely. The cost tuples are updated based on the values of their children's tuples in a bottom-up procedure.

Let us consider a series operation of v_j, v_l forming v_i. Then the cost $c_i(k, f, d)$ is a composition of cost tuples of v_j and v_l. As mentioned before, the demand of t_j must be satisfied, and the entire flow that is present in G_l, that is $f + h_l + d$, must have been sent through G_j. The budget k is split up between G_j and G_l such that $c_i(k, f, d)$ is minimal. Formally, this leads to

$$c_i(k, f, d) = \min_{0 \leq k' \leq k} \{c_j(k', f + h_l + d, b(t_j)) + c_l(k - k', f, d)\} \tag{1}$$

Let us now consider a parallel operation of v_j, v_l forming v_i. Then the cost $c_i(k, f, d)$ depends on whether the flow that does not satisfy the inner demands h_j and h_l is sent through G_j or G_l. Again, the budget k is split up between G_j and G_l such that $c_i(k, f, d)$ is minimal. This leads to

$$\begin{aligned} c_i(k, f, d) = \min\{ &\min_{0 \leq k' \leq k} \{c_j(k', f, d) + c_l(k - k', 0, 0)\}, \\ &\min_{0 \leq k' \leq k} \{c_j(k', 0, 0) + c_l(k - k', f, d)\}\} \end{aligned} \tag{2}$$

If v_j is a leaf node of the SP-tree T, then the cost tuples are initialized as follows:

$$c_i(k, f, d) = \begin{cases} (f + d) \cdot \bar{c}(a) & \text{if } k = 0 \\ (f + d) \cdot \underline{c}(a) & \text{if } k = 1 \\ \infty & \text{if } k > 1 \end{cases}$$

Now that we described the updating steps of the labels, we show that the label $c_0(K, 0, b(t))$ is the optimal solution value of the BMCF(K) problem on an SP-graph.

Theorem 13. *The value of $c_0(K, 0, b(t))$ equals the optimal solution value of the BMCF(K) problem on a series-parallel graph G. An optimal solution can be obtained by backtracking the steps of the dynamic program that determined $c_0(K, 0, b(t))$.*

Proof. We prove the first statement by induction.

In the base case we show that $c_i(k, f, d)$ is correct for the leaves. Depending on whether the arc a associated with a leaf is upgraded or not, the cost of one unit of flow along this arc is either $\bar{c}(a)$ or $\underline{c}(a)$. Hence, sending $f + d$ units of flow costs either $(f + d) \cdot \bar{c}(a)$ or $(f + d) \cdot \underline{c}(a)$.

Our induction hypothesis is that for two SP nodes v_j and v_l the cost tuples $c_j(k_j, f_j, d_j)$ and $c_l(k_l, f_l, d_l)$ equal the optimal cost of flow satisfying h_j and d_j in G_j (respectively h_l and d_l in G_l) plus sending f_j (and f_l) units of additional flow from s_j to t_j (s_l to t_l) when k_j (and k_l) arcs are upgraded.

Inductive Step: Assuming the induction hypothesis holds, we show equation (1) yields the optimal value in case G_j and G_l are conjoined via a series operation becoming G_i and equation (2) if a parallel operation was performed.

Equation (1): If k arcs can be upgraded in G_i, it is straightforward that if k' arcs are upgraded in G_j only $k-k'$ arcs can be upgraded in G_l. If f units of additional flow are present at t_i, there are also f units of additional flow present at t_l as $t_i = t_l$. The entire flow that is sent through G_l (satisfying demand in G_l or being additional flow) must have been sent through G_j and therefore must have been additional flow at t_j. Since t_j will not be a target node again, its demand must be satisfied by the flow sent through G_j. Hence, for given f and d in G_i we can derive the unique values of f and d both in G_j and G_l. Since equation (1) minimizes over how the budget k is split between G_j and G_l, the value of $c_i(k, f, d)$ gives the optimal total cost of the flow in G_i satisfying h_i and of $f + d$ additional units of flow if k arcs are upgraded.

Equation (2): As in the first case, if k arcs can be upgraded in G_i, it is straightforward that if k' arcs are upgraded in G_j only $k-k'$ arcs can be upgraded in G_l. The entire additional flow in G_i can be either routed through G_j or G_l because of the same reasoning as in the proof of Corollary 7. Hence, equation (2) first minimizes over how the budget k is split between G_j and G_l given that $f + d$ units of flow are sent to t through G_j, and respectively through G_l. Equation (2) then chooses the smaller value (representing the choice of sending $f + d$ units of flow either entirely through G_j or entirely through G_l). Together with the induction hypothesis equation (2) yields the optimal cost of the total flow in G_i satisfying h_i and of $f + d$ additional units of flow if k arcs are upgraded in G_i after a parallel operation.

By backtracking the chosen minima to the SP leaves, we can determine both which arcs are upgraded and how much flow is sent via each arc. This gives us the pair (f^*, A_K^*). □

After having shown the correctness of the dynamic program, we now analyze its running time.

Theorem 14. *The previously described dynamic program runs in time pseudo-polynomial in the input size.*

Proof. The number of nodes of the SP tree is in $\mathcal{O}(m)$. In each SP node v_i, tuples for all combinations of k smaller or equal to K, f smaller or equal to the total demand D of G and $d \in \{0, b(t_i)\}$ are calculated. Hence, for each SP node we need to compute at most $K \cdot D \cdot 2$ different cost labels. The evaluation of equation (1) and (2) are done in polynomial time. Therefore, the dynamic program is pseudo-polynomial in the total demand D of G. □

4.2 Graphs with bounded tree width

Several $\mathcal{NP}$-hard optimization problems, such as vertex cover, independent set, dominating set, graph k-colorability, Hamiltonian circuit, and network reliability, can be solved in polynomial time on graphs with bounded tree width (see Bodlaender [1988] or Arnborg and Proskurowski [1989]). In this section we make use of dynamic programming that is applied to a tree decomposition in order to solve the BMCF(K) problem.

We start by giving a formal definition of a *tree-decomposition* and its *tree-width*.

Definition 15. *Let $G = (V, A)$ be a digraph. A tree-decomposition of G is a tree $T = (V(T), A(T))$ where each $v_i \in V(T)$ is associated with a subset of vertices of G. This subset is called the bag of node v_i and denoted by X_i. The following statements must hold:*

1. *there is at least one bag X_i for each $v \in V$ such that $v \in X_i$*
2. *there is at least one bag X_i for each $a = (u, v) \in A$ such that $u, v \in X_i$*
3. *for all $v \in V$, the set of v_i where $v \in X_i$ induces a connected subgraph of T*

The *width* of a tree-decomposition is then defined as $\max_i |X_i| - 1$ and the *tree-width* of a graph G, denoted by $tw(G)$, is the minimum width over all tree-decompositions.

To improve the readability of the algorithm, we use the common concept of a *nice tree-decomposition* (cf. Bodlaender and Koster [2008]) in what follows. A nice tree-decomposition is a tree T^* with at most two children per node and $|X_l| = 1$ for all leaves l. Further, for any two $v_i, v_j \in V(T^*)$ with $(v_i, v_j) \in A(T^*)$ and $i < j$, i.e. node i is closer to the root than node j, it holds that $|X_i| - |X_j| \in \{-1, 0, 1\}$. If the difference is -1, then $X_i \subseteq X_j$. In this case v_i is called a *forget node* as the corresponding bag X_i contains one less vertex of G than X_j does. We say v_i *forgets* a vertex. If the difference is 1, then $X_j \subseteq X_i$ and v_i is called an *introduce node* as X_i contains one additional vertex of G compared to X_j and v_i *introduces* a vertex. If the difference is 0, then $X_i = X_j$ and v_i is called a *joint node*. In this case, there exists a node v_k that is adjacent to v_i with $k \notin \{i, j\}$ and $X_k = X_i = X_j$. Note that any

tree-decomposition T can be transformed to a nice tree-decomposition with the same width and $\mathcal{O}(n)$ bags (see Kloks [1994]).

Let us consider the BMCF(K) problem on a graph $G = (V, A)$ whose given nice tree-decomposition T^* has a bounded width of tw^*, i.e. $|X_i| - 1 \leq tw^*$ for all v_i. Furthermore, we can assume that T^* is a binary tree. Let $X_i = \{u, v, \dots\}$ be the bag associated with node v_i of T^*. By means of a simple preprocessing step of G we ensure that the supply and demand vertices appear in exactly two bags of T^*: We introduce dummy supply and dummy demand vertices in G, transfer the demand from the original demand vertices to the dummy vertices and associate upper and lower costs of 0 to the arcs connecting the original with the dummy vertices. Note, that we delete the supply in the graph. This is compensated by choosing appropriate labels in the dynamic program as explained later on. Furthermore, the dummy demand vertices appear in bags that are associated with leaves of T^* and the dummy supply vertex is in the bag X_0 of the root v_0 of T^*. Let $A[X_i]$ be the arcs of G that are induced by X_i, i.e. all arcs that are in bag X_i. Let G_i be the subgraph that is induced by all the vertices in X_i and the vertices of the subtree of T^* that is rooted in v_i of T^*.

Let the *in-flow* $f^{I,i} = (f^I_{w_1}, f^I_{w_2}, \dots, f^I_{w_{|X_i|}})$ be a vector that specifies the flow that enters $w_1, w_2, \dots, w_{|X_i|} \in X_i$ from outside of G_i. Let $f^{O,i} = (f^O_{w_1}, f^O_{w_2}, \dots, f^O_{w_{|X_i|}})$ be a vector that specifies the flow that leaves $w_1, w_2, \dots, w_{|X_i|} \in X_i$ and is sent into the remaining part of G, the so-called *out-flow*. Furthermore, let $f^i_a = (f^i_{a_1}, f^i_{a_2}, \dots, f^i_{|A[X_i]|})$ be a vector specifying the flow on arcs $a_1, a_2, \dots, a_{|A[X_i]|} \in A[X_i]$ and let $k_a = (k_{a_1}, k_{a_2}, \dots, k_{|A[X_i]|})$ indicate which arcs in $A[X_i]$ are upgraded. If $k_{a_j} = 1$ then arc a_j is upgraded and 0 otherwise. Let $k^i_f \in \mathbb{N}$ be the number of upgraded arcs in $A(G_i) \backslash A[X_i]$.

In order to define a dynamic program to solve the BMCF(K) problem we associate labels $x_i = (f^{I,i}, f^{O,i}, f^i_a, k^i_a, k^i_f)$ to a node v_i in T^*. For each label x_i we define $c_i(x_i)$ as the cost of a so called x_i-*restricted* BMCF(K^i), denoted by rBMCF(x_i). This rBMCF(x_i) problem is a BMCF(K^i) problem with $K^i = k^i_f + \sum_{a \in A[X_i]} k_a$ on G_i where the flow on arcs in $A[X_i]$ is predetermined by f^i_a. Furthermore, k^i_a determines which arcs are upgraded in $A[X_i]$. All $v \in X_i$ are supply vertices with a supply $\tilde{b}(v) = f^I_v - f^O_v + b(v)$. Note that $b(v) = 0$ for all $v \in X_i$ if v_i is a joint node of T^*. This is due to the described preprocessing to obtain a nice tree decomposition.

If an rBMCF(x_i) problem instance is feasible, then the in-flow to all vertices inside a bag minus the demand of all vertices in the induced subgraph G_i equals the out-flow from all vertices inside a bag. More precisely, if $\sum_{v \in X_i} f^{I,i}_v + \sum_{v \in V_i} b(v) \neq \sum_{v \in X_i} f^{O,i}_v$ then $c_i(x_i) = \infty$ for all i.

Using the proposed structure of labels we can calculate labels of a specific tree-decomposition node v_i by using the labels of its child nodes. We say that the label of the child nodes *build*

the current node's label and that the current node's cost is updated. Before we discuss the different updating procedures, we start with describing the initialization of the costs associated with leaf nodes.

Initialize For all leaves l of T, we initialize all costs as follows:

$$c_l(f^{I,l}, f^{O,l}, \mathbf{0}, \mathbf{0}, k_f^l) = \begin{cases} 0 & \text{if } f_v^I + b(v) = f_v^O \text{ and } k_f^l = 0 \\ \infty & \text{else} \end{cases}$$

for all values $f_v^I, f_v^O \in [0, D], k_f^l \in [0, K]$. Note that $(f_v^{I,l})_{v \in X_i} = f_v^I$ as there is exactly one vertex in the bag X_l. Both values $\mathbf{0}$ stand for the fact that there are no arcs in bag X_i and hence both $(f_a)_{a \in A[X_i]}$ and $(k_a)_{a \in A[X_i]}$ are vectors of size zero.

Lemma 16. *If v_i is a leaf node, then $c_i(x_i)$ is the optimal solution value to the rBMCF(x_i) problem with $K^i = k_f^i$.*

Proof. If $f_v^{I,i} + b(v) = f_v^{O,i}$ and $k_f^i = 0$, then the rBMCF(x_i) to the corresponding label x_i is feasible. The optimal solution value is 0 as the induced subgraph G_i is a single node with no arcs. If $f_v^{I,i} + b(v) \neq f_v^{O,i}$ or $k_f^i > 0$, then the induced problem is infeasible and the solution value unbounded. □

Introduce Node Let v_i be the current tree-decomposition node and let v_j be its only child node. Let u be the vertex that is introduced in v_i, more precisely $\{u\} = X_i \backslash X_j$. Let U be the set of arcs that are in bag X_i but not in X_j, i.e. $U = A[X_i] \backslash A[X_j]$. Furthermore, let $c_a(k_a^i) = \bar{c}(a)$ if $k_a^i = 0$ and $c_a(k_a^i) = \underline{c}(a)$ if $k_a^i = 1$.

Lemma 17. *Let v_i be an introduce node, x_i a label and and x_j a label of the child node v_j with $f_v^{I,j} = f_v^{I,i} + f_{uv}^i$, $f_v^{O,j} = f_v^{O,i} + f_{vu}^i$ both for all $v \in X_j$; $f_a^j = f_a^i$, $k_a^j = k_a^i$ both for all $a \in A[X_j]$ and $k_f^j = k_f^i$. Then for $K^i = K^j + \sum_{a \in U} k_a^i$*

$$c_i(x_i) = c_j(x_j) + \sum_{a \in U} f_a^i \cdot c_a(k_a^i).$$

Proof. Since u is the vertex that is introduced in v_i, u is adjacent to only vertices in X_i or in $V \backslash V_i$. Hence, if flow is sent through u it either comes from a vertex in the bag X_i (or X_j for that matter) or it comes from outside of G_i.

Let $c_i(x_i)$ be the optimal solution value to the rBMCF(x_i) problem. Given an optimal solution $(f^{i,*}, A_K^{i,*})$, one can construct a feasible solution to an rBMCF(x_j) problem with $K^j = K^i - \sum_{a \in U} k_a^i$: copy all relevant values of $(f^{i,*}, A_{K^i}^{i,*})$, i.e. ignore the flow values on any arc adjacent to u (because they are not in G_j) and ignore the information if any arc adjacent to u is upgraded or not. As $f_v^{I,j} = f_v^{I,i} + f_{uv}^i$ and $f_v^{O,j} = f_v^{O,i} + f_{vu}^i$ for all $v \in X_j$

the flow conservation holds for all $v \in X_j$ and as the other flow values are simply copied from a feasible solution the flow conservation also holds for all $v \in V(G_j) \backslash X_j$ as well. Hence, this results in a feasible flow in G_j. Therefore,

$$c_j(x_j) \leq c_i(x_i) - \sum_{a \in U} f_a^i \cdot c_a(k_a^i). \tag{3}$$

Let $c_j(x_j)$ be the optimal solution value to the rBMCF(x_j) problem. Given an optimal solution $(f^{j,*}, A_K^{j,*})$, one can construct a feasible solution to an rBMCF(x_i) problem with $K^i = K^j + \sum_{a \in U} k_a^i$: copy all values of $(f^{j,*}, A_{K^j}^{j,*})$. With the same arguments as before, the flow conservation holds for all $v \in V_j$. Therefore,

$$c_j(x_j) + \sum_{a \in U} f_a^i \cdot c_a(k_a^i) \geq c_i(x_i) \tag{4}$$

which concludes the proof. □

Forget Node Let v_i be the current tree-decomposition node and let v_j be its only child node. Let $u \in V$ be the vertex that v_i forgets. Let U be the set of arcs that were in bag X_j but not in bag X_i, i.e. $U = A[X_j] \backslash A[X_i]$.

Lemma 18. *Let v_i be a forget node, and let $B(x_i)$ be the set of all labels x_j of the child node v_j with $f_v^{I,j} = f_v^{I,i}$ and $f_v^{O,j} = f_v^{O,i}$ for all $v \in X_i$; $f_u^{I,j} = f_u^{O,j} = 0$, $f_a^j = f_a^i$ and $k_a^j = k_a^i$ for all $a \in A[X_i]$, and $k_f^j + \sum_{a \in U} k_a^j = k_f^i$. Then*

$$c_i(x_i) = \min_{x_j \in B(x_i)} \{c_j(x_j)\}$$

Proof. Observe that $G_i = G_j$. The main difference between the rBMCF(x_i) instance I and the rBMCF(x_j) instance I' lies in the set U: in I', the flow and whether an arc a is upgraded is pre-determined for all $a \in U$. Furthermore, vertex u is not a supply vertex in I.

Assume we are given an optimal solution $(f^{i,*}, A_K^{i,*})$ to the rBMCF(x_i) problem. We create a label $\bar{x}_j$ by copying the in- and out-flow values of x_i for all vertices in bag X_i, setting all f_a^j equal to the flow value in $f^{i,*}$ with $a \in A[X_i] \cup U$, setting all k_a^j with $a \in A[X_i] \cup U$ according to whether $a \in A^{i,*}$ or not, and finally setting $k_f^j = k_f^i - \sum_{a \in U} k_a^j$. It is trivial that $(f^{i,*}, A_K^{i,*})$ is also a feasible solution to the rBMCF($\bar{x}_j$) problem. Hence, it holds that $c_j(\bar{x}_j) \leq c_i(x_i)$. It is also clear that an optimal solution $(f^{j,*}, A_K^{j,*})$ to the rBMCF(x_j) problem for any label x_j obeying the preconditions of Lemma 18 is a feasible solution to the rBMCF(x_i) problem. Hence, it also holds that $c_i(x_i) \leq c_j(x_j)$ which concludes the proof. □

Joint Node Let v_i be the parental node of v_j and v_l. Note however, the bags of v_i, v_j, v_l are identical, but G_i, G_j, G_l are not. Let x_i, x_j, x_l be labels of the corresponding nodes. In the case of a joint node, we say that x_j and x_l *build* x_i if the following equalities hold:

$$f_a^j = f_a^l = f_a^i \qquad \forall a \in A[X_i] \tag{5}$$

$$k_a^j = k_a^l = k_a^i \qquad \forall a \in A[X_i] \tag{6}$$

$$k_f^i = k_f^j + k_f^l \tag{7}$$

$$\begin{aligned} f_v^{I,i} - f_v^{O,i} = {} & f_v^{I,j} - f_v^{O,j} + f_v^{I,l} - f_v^{O,l} \\ & - \sum_{a \in \delta^+(v) \cap A[X_i]} f_a^i + \sum_{a \in \delta^-(v) \cap A[X_i]} f_a^i \qquad \forall v \in X_i \end{aligned} \tag{8}$$

Lemma 19. *Let v_i be a joint node, v_j and v_l its child nodes. Let x_j and x_l be labels that build label x_i. Then*

$$c_i(x_i) \leq c_j(x_j) + c_l(x_l) - \sum_{a \in A[X_i] \setminus A[X_j]} f_a \cdot c_a(k_a)$$

Proof. Let $(f^{j,*}, A^{j,*})$ and $(f^{l,*}, A^{l,*})$ be an optimal solution to the rBMCF(x_j), respectively rBMCF(x_l) problem.

Let us define the following edge values h^i on G_i in the following way:

$$\begin{aligned} h_a^i &= f_a^{j,*} && \text{if } a \in A[X_i] \\ h_a^i &= f_a^{j,*} && \text{if } a \in A(G_j) \setminus A[X_i] \\ h_a^i &= f_a^{l,*} && \text{if } a \in A(G_l) \setminus A[X_i] \end{aligned}$$

We need to show that h^i is a feasible flow for the rBMCF(x^i). This is the case if

$$\sum_{a \in \delta^+(v)} h_a^i - \sum_{a \in \delta^-(v)} h_a^i = \tilde{b}_i(v)$$

for all $v \in X_i$. Let $A_j = A(G_j) \setminus A[X_j]$ and let $A_l = A(G_l) \setminus A[X_l]$. Then

$$\begin{aligned} \sum_{a \in \delta^+(v)} h_a^i - \sum_{a \in \delta^-(v)} h_a^i = {} & \sum_{a \in \delta^+(v) \cap A[X_i]} h_a^i + \sum_{a \in \delta^+(v) \cap A_j} h_a^i + \sum_{a \in \delta^+(v) \cap A_l} h_a^i \\ & - \sum_{a \in \delta^-(v) \cap A[X_i]} h_a^i - \sum_{a \in \delta^-(v) \cap A_j} h_a^i - \sum_{a \in \delta^-(v) \cap A_l} h_a^i \\ = {} & \sum_{a \in \delta^+(v) \cap A[X_i]} f_a^{j,} + \sum_{a \in \delta^+(v) \cap A_j} f_a^{j,*} + \sum_{a \in \delta^+(v) \cap A_l} f_a^{l,*} \\ & - \sum_{a \in \delta^-(v) \cap A[X_i]} f_a^{j,*} - \sum_{a \in \delta^-(v) \cap A_j} f_a^{j,*} - \sum_{a \in \delta^-(v) \cap A_l} f_a^{l,*} \end{aligned}$$

$$
\begin{aligned}
&= \tilde{b}_j(v) + \tilde{b}_l(v) - \sum_{a \in \delta^+(v) \cap A[X_i]} f_a^{j,*} + \sum_{a \in \delta^-(v) \cap A[X_i]} f_a^{j,*} \\
&= f_v^{I,j} - f_v^{O,j} + f_v^{I,l} - f_v^{O,l} - \sum_{a \in \delta^+(v) \cap A[X_i]} f_a^{j,*} + \sum_{a \in \delta^-(v) \cap A[X_i]} f_a^{j,*} \\
&\overset{(8)}{=} f_v^{I,i} - f_v^{O,i} \\
&= \tilde{b}_i(v).
\end{aligned}
$$

Therefore h^i is a feasible flow and by setting $A^i = A^{j,*} \cup A^{l,*} \cup \{a \in A[X_i] : k_a^i = 1\}$, we obtain

$$
\begin{aligned}
c_i(x_i) &\leq \sum_{a \in A(G_i) \cap A^i} \underline{c}_a h_a^i + \sum_{a \in A(G_i) \setminus A^i} \overline{c}_a h_a^i \\
&= \sum_{a \in A(G_j) \cap A^i} \underline{c}_a f_a^{j,*} + \sum_{a \in A(G_l) \cap A^i} \underline{c}_a f_a^{l,*} \\
&\quad + \sum_{a \in A(G_j) \setminus A^i} \overline{c}_a f_a^{j,*} + \sum_{a \in A(G_l) \setminus A^i} \overline{c}_a f_a^{l,*} - \sum_{a \in A[X_i]} c(k_a) f_a^{j,*} \\
&= c_j(x_j) + c_l(x_l) - \sum_{a \in A[X_i]} c(k_a) f_a^{j,*}.
\end{aligned}
$$

□

Now we show that the inequality in Lemma 19 holds with equality for some labels.

Lemma 20. *Let v_i be a joint node, v_j and v_l its child nodes. Let x_i be a label of v_i. Then there are always two labels x_j and x_l that build x_i such that*

$$
c_i(x_i) = c(x_j) + c(x_l) - \sum_{a \in A[X_i]} c(k_a) f_a^i.
$$

Proof. Let $(f^{i,*}, A^{i,*})$ be an optimal solution for rBMCF(x_i) and let $A^* = A^{i,*} \cap A(G_i) \setminus A[X_i]$. In the following we first define two labels x_j and x_l such that they build x_i and show that the above equation holds for these labels. For $v \in X_i$, let $\delta_j(v) = \sum_{a \in \delta^+(v) \cap A(G_j)} f_a^{i,*} - \sum_{a \in \delta^-(v) \cap A(G_j)} f_a^{i,*}$ and let us define

$$
f_v^{I,j} = \begin{cases} \delta_j(v) & \text{if } \delta_j(v) > 0 \\ 0 & \text{otherwise} \end{cases}, \quad f_v^{O,j} = \begin{cases} |\delta_j(v)| & \text{if } \delta_j(v) \leq 0 \\ 0 & \text{otherwise} \end{cases}
$$

Similar let $\delta_l(v) = \sum_{a \in \delta^+(v) \cap G_l} f_a^{i,*} - \sum_{a \in \delta^-(v) \cap G_l} f_a^{i,*}$ and let us define

$$
f_v^{I,l} = \begin{cases} \delta_l(v) & \text{if } \delta_l(v) > 0 \\ 0 & \text{otherwise} \end{cases}, \quad f_v^{O,l} = \begin{cases} |\delta_l(v)| & \text{if } \delta_l(v) \leq 0 \\ 0 & \text{otherwise} \end{cases}
$$

Furthermore, let $f_a^j = f_a^l = f_a^i$, $k_a^j = k_a^l = k_a^i$ for $a \in A[X_i]$. Using $A^{j,*} = A^* \cap A(G_j)$ and $A^{l,*} = A^* \cap A(G_l)$ define $k_f^j = |A^{j,*}|$ and $k_f^l = |A^{l,*}|$. This completely defines two labels x_j and x_l that build x_i.

Now we show that we can construct feasible solutions to the resulting rBMCF(x_j) and rBMCF(x_l) problems such that the above equality holds. Therefore define the flow f^j in G_j as $f_a^j = f_a^{i,*}$ for all $a \in A(G_j)$ and the flow f^l in G_l as $f_a^l = f_a^{i,*}$ for all $a \in A(G_l)$. Let $A^{j,*}$ and $A^{l,*}$ be the upgraded arcs in the corresponding problems that are not in $A[X_i]$. These flows are feasible because the flow conservation holds. This can easily be seen due to the definition of $\delta_j(v)$ and $\delta_l(v)$.

Using $A^j = \{a \in A[X_j] : k_a^j = 1\}$ and $A^l = \{a \in A[X_l] : k_a^l = 1\}$, we know

$$\begin{aligned} c_i(x_i) + \sum_{a \in A[X_i]} c(k_a) f_a^{i,*} &= \sum_{a \in G_j \cap (A^{j,*} \cup A^j)} \underline{c}_a f_a^{i,*} + \sum_{a \in G_j \setminus (A^{j,*} \cup A^j)} \overline{c}_a f_a^{i,*} \\ &+ \sum_{a \in G_l \cap (A^{l,*} \cup A^l)} \underline{c}_a f_a^{i,*} + \sum_{a \in G_l \setminus (A^{l,*} \cup A^l)} \overline{c}_a f_a^{i,*} \\ &\geq c_j(x_j) + c_l(x_l) \end{aligned}$$

Together with the result from Lemma 19 this concludes the proof. □

Corollary 21. *Let v_i be a joint node and x_i a label. Let x_j and x_l denote label of the child nodes v_j and v_l. Let $B(x_i) = \{(x_j, x_l) | x_j, x_l \text{ build } x_i\}$. Then*

$$c_i(x_i) = \min_{(x_j, x_l) \in B(x_i)} \left\{ c(x_j) + c(x_l) - \sum_{a \in A[X_i]} c(k_a) f_a^i \right\}$$

Proof. This result directly follows from Lemma 19 and 20. □

Now that we described the updating rules for all three types of tree-decomposition nodes, we show that the proposed dynamic program solves the BMCF(K) problem to optimality.

Theorem 22. *The cost $c_0(x_0)$ with $x_0 = (b(s), 0, \mathbf{0}, \mathbf{0}, K)$ equals the optimal solution value of the BMCF(K) problem on a graph $G = (V, A)$. An optimal solution (f^*, A_K^*) can be found by backtracking the chosen label in the tree-decomposition T^*.*

Proof. By the definition of induced subgraphs, $G_0 = G$. We also know that the dummy supply vertex is the only vertex in X_0. Since the inflow of $b(s)$ into the dummy supply vertex can be interpreted as a supply of the vertex, solving rBMCF(x_0) with the specified x_0 is equivalent to solving BMCF(K) on G.

We proof that $c_0(x_0)$ of the given label is indeed the optimal value of the BMCF(K) problem by induction, where the base is proven in Lemma 16 and the inductive steps in Lemma 17 and 18, and Corollary 21. □

It remains to show that the proposed dynamic program runs in time pseudo-polynomial on graphs with bounded tree-width.

Theorem 23. *The calculations to obtain $c_0(x_0)$ with $x_0 = (b(s), 0, \mathbf{0}, \mathbf{0}, K)$ for a graph $G = (V, A)$ with a bounded tree-width tw^* can be done in time pseudo-polynomial in the input size.*

Proof. First we show that the number of considered labels is pseudo-polynomially bounded by the input size. Afterwards we show that the updating steps are done in time pseudo-polynomial in the input size as well.

We claim that both the total number of considered labels as well as the entries in a label are pseudo-polynomially bounded by the input size. The number of labels $x_i = (f^{I,i}, f^{O,i}, f_a^i, k_a^i, k_f^i)$ associated with each $v_i \in V(T^*)$ depends on the possible combinations of its vectors and their entries. Consider the in- and out-flow vectors $f^{I,i}$,$f^{O,i}$. Both have $|X_i|$ many entries. As G has bounded tree-width tw^* it holds that $|X_i| \leq tw^* + 1$. Also each of their entries is between 0 and D, where D denotes the total demand in G. The sizes of the vectors f_a^i and k_a equal $|A[X_i]|$. The maximum size is obtained when $A[X_i]$ is a clique. Because of the bounded tree-width and as G can be assumed to be simple, the number of arcs in the clique is at most $tw^*(tw^* + 1)$. Each entry of f_a^i is between 0 and D. The entries of k_a are either 0 or 1. Hence, there are $(D+1)^{(tw^*)^2+3tw^*+2} \cdot 2^{(tw^*)^2+tw^*} \cdot K$ labels in each node of T^*. This number is in $\mathcal{O}(D^c \cdot K)$ with c constant.

In the initialization as well as in the updating step for introduce nodes, there are only assignments and no calculations done. Hence, they need constant time. As for the calculations for forget nodes: we minimize over $(k_a)_{a \in U}$ and k_j. Since G has a bounded tree-width, it holds that $|U| \leq 2 \cdot tw^*$. Together with $k \leq K$, we can simply try all possible combinations of $(k_a)_{a \in U}$ and k_j, and then choose the smallest $c_j(x_j)$. Hence the calculations if v_i is a forget node are done in constant time as well. Concerning the joint nodes: we minimize over the set $B(x_i) = \{(x_j, x_l) | x_j, x_l \text{ build } x_i\}$. Its cardinality is bounded by $K \cdot D^{4(tw^*+1)}$ which is in $\mathcal{O}(D^c \cdot K)$. Therefore, the calculations if v_i is a joint node are also done in pseudo-polynomial time. This concludes the proof. □

As the running time of the proposed dynamic program depends on the total demand D, it is pseudo-polynomial in D. Note however, that the running time becomes polynomial in the input size when the instances are restricted to have only unit demands.

5 Conclusion

In this paper, we studied the BMCF(K) problem which is an extension of the classical minimum cost flow problem where at most K arcs can be chosen to be associated with a smaller

cost per unit of flow. We restricted our attention to the case of unlimited capacities. To the best of our knowledge this problem has not been studied in the literature on its own, but can be seen as a special case of the AAUP. We have shown that even this special case is strongly $\mathcal{NP}$-complete which improves the complexity results found by Coene et al. [2013]. Although the problem is strongly $\mathcal{NP}$-complete, we have proposed polynomial time algorithms for the BMCF(K) on arborescenses, tree-like graphs and for instances with a constant number of sinks. For series-parallel graphs and, more general, graphs of bounded tree-width, we proposed pseudo-polynomial time algorithms.

In a next step we are interested in running time improvements for further graph classes such as graphs of bounded pathwidth. Furthermore, it is of interest to what extent our results can be transferred to graphs with capacities, since we heavily rely on the tree-structure of an optimal solution. Unless $\mathcal{P} = \mathcal{NP}$, there is no (pseudo-)polynomial algorithm on directed acyclic graphs as can be directly seen from the reduction in Section 2.2. However, obtaining exact algorithms is not limited to polynomial special cases. By linearizing a bi-linear formulation, one can obtain an IP formulation. In this case, developing valid inequalities and hence describing the solution space more precisely seems promising.

Acknowledgements

We thank our former master's student Luisa Eickmeyer for her work.

References

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

Ravindra K Ahuja and James B Orlin. A capacity scaling algorithm for the constrained maximum flow problem. *Networks*, 25(2):89–98, 1995.

Y. P. Aneja and K. P. K. Nair. The constrained shortest path problem. *Naval Research Logistics Quarterly*, 25(3):549–555, 1978.

Stefan Arnborg and Andrzej Proskurowski. Linear time algorithms for np-hard problems restricted to partial k-trees. *Discrete Applied Mathematics*, 23(1):11 – 24, 1989.

Hans L. Bodlaender. Dynamic programming on graphs with bounded treewidth. In Timo Lepisto and Arto Salomaa, editors, *Automata, Languages and Programming*, volume 317 of *Lecture Notes in Computer Science*, pages 105–118. Springer Berlin Heidelberg, 1988. ISBN 978-3-540-19488-0.

Hans L Bodlaender and Arie M. C. A. Koster. Combinatorial optimization on graphs of bounded treewidth. *The Computer Journal*, 51(3):255–269, 2008.

S. Coene, Goos P., Maya P., Sörensen K., and Spieksma F. C. R. The budget-constrained min cost flow problem. 2013.

I. Demgensky, H. Noltemeier, and H.-C. Wirth. On the flow cost lowering problem. *European Journal of Operational Research*, 137(2):265 – 271, 2002. Graphs and Scheduling.

Michael Holzhauser, Sven O. Krumke, and Clemens Thielen. Budget-constrained minimum cost flows. *Journal of Combinatorial Optimization*, pages 1–26, 2015. ISSN 1382-6905.

Tohru Kikuno, Noriyoshi Yoshida, and Yoshiaki Kakuda. A linear algorithm for the domination number of a series-parallel graph. *Discrete Applied Mathematics*, 5(3):299 – 311, 1983.

T. Kloks. *Treewidth. Computations and Approximations*, volume 842 of *Lecture Notes in Computer Science*. Springer Verlag, 1994.

P.A. Maya Duque, S. Coene, P. Goos, K. Sörensen, and F. Spieksma. The accessibility arc upgrading problem. *European Journal of Operational Research*, 224(3):458 – 465, 2013.

James B Orlin. A faster strongly polynomial minimum cost flow algorithm. *Operations research*, 41(2):338–350, 1993.

S Schwarz and Sven Oliver Krumke. On budget-constrained flow improvement. *Information Processing Letters*, 66(6):291–297, 1998.

Peter W Shor. A new proof of cayley's formula for counting labeled trees. *Journal of Combinatorial Theory, Series A*, 71(1):154 – 158, 1995.

Jacobo Valdes, Robert E. Tarjan, and Eugene L. Lawler. The recognition of series parallel digraphs. In *Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing*, STOC '79, pages 1–12, New York, NY, USA, 1979. ACM.