The Maximum Flow Problem for Oriented Flows*

Stanley Schade[1] and Martin Strehler[2]

1 **Zuse Institute Berlin**
Takustr. 7, 14195 Berlin, Germany,
schade@zib.de
2 **BTU Cottbus - Senftenberg,**
Postfach 101344, 03013 Cottbus, Germany
strehler@b-tu.de

Abstract

In several applications of network flows, additional constraints have to be considered. In this paper, we study flows, where the flow particles have an orientation. For example, cargo containers with doors only on one side and train coaches with 1st and 2nd class compartments have such an orientation. If the end position has a mandatory orientation, not every path from source to sink is feasible for routing or additional transposition maneuvers have to be made. As a result, a source-sink path may visit a certain vertex several times. We describe structural properties of optimal solutions, determine the computational complexity, and present an approach for approximating such flows.

1998 ACM Subject Classification G.2.2 Network problems, G.2.3 Applications

Keywords and phrases network flow with orientation, graph expansion, approximation, container logistics, train routing

Digital Object Identifier 10.4230/OASIcs.ATMOS.2016.7

1 Introduction

1.1 Motivation

In many practical applications of network flows, the entities that are flowing have an orientation. Consider for example a modern container terminal like the container terminal Altenwerder[1] in the port of Hamburg. The containers there are transported by automated guided vehicles, which are centrally controlled and driverless. Thus, it is possible to operate them back and forth without limitations, e.g., the direction of motion could be changed to make a sharp turn. However, it matters to which direction the doors of the containers open.

At sea, to protect the cargo against wave impact and spray, the doors of a container should point astern. At port, doors of containers should point towards the driveway to be easily accessible, e.g., for custom inspections. Loaded on a truck, the doors should be again at the back. Apart from these rules, the desired orientation may depend on other side constraints. For example, refrigerated containers have to be positioned in reach of a power supply. Usually, connectors are only on one side of the container and on one side of the storing position. Furthermore, containers have different sizes, e.g., two twenty foot containers may be placed on a forty foot container. As shown in the example in Figure 1, this

* This work was partially supported by Research Campus MODAL.
[1] https://en.wikipedia.org/wiki/Container_Terminal_Altenwerder

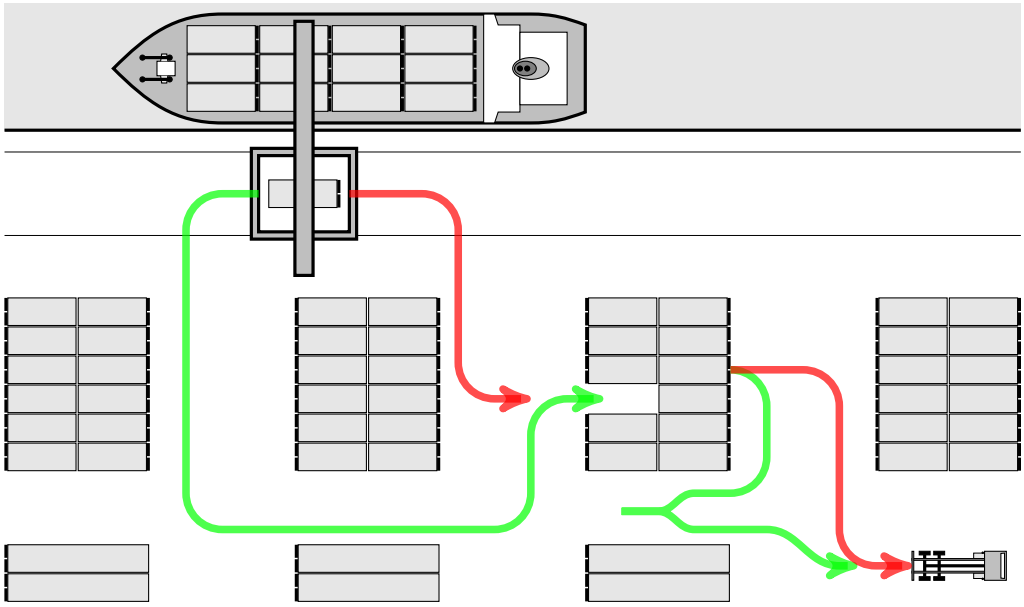

Figure 1 Oriented flows occur, e.g., when handling containers at a terminal. The red paths are shorter, but the desired orientation is not achieved. The green paths fulfill all requirements, but detours or additional moves to change the orientation are needed. Considering several automated guided vehicles carrying containers at the same time, i.e., the corresponding network flow problem, the orientation constraints may cause a significant drop in the network's capacity due to such extra moves.

may lead to detours or transposition maneuvers while handling these containers to achieve the desired orientations.

Another practical example is the orientation of railcars, where the orientation indicates, e.g., whether the first or second class is at the beginning of a car and is relevant considering seat numbering, rest rooms, or luggage compartments. At terminals, trains have to reverse out of the station, thus, changing their orientation. These aspects are particularly relevant in rolling stock rotation planning.

The considered applications surely involve dynamic aspects, but here, we are going to study the underlying static flow problem. Regarding the corresponding static maximum flow problem of this underlying routing problem, e.g., handling several containers at the same time, also yields bounds to the performance of the network and the dual cut problem enables to identify bottlenecks in the current infrastructure.

We can incorporate the orientation of containers by also equipping the flow units with an orientation which indicates whether the doors of the corresponding transported containers head into the direction of motion. It should be noted that this orientation of flow must not be confused with the orientation of arcs. Each arc can only be used in the correct direction, but flow particles on this arc may have both orientations.

1.2 Related work

The maximum flow problem is a well-known linear optimization problem. Even without the use of the sophisticated tools of linear programming, one can easily determine an optimal solution, which can be chosen to be integral if the capacities are integral, in strongly polynomial time. There is a huge number of fast combinatorial algorithms available, see [9] for

an overview. The optimality of a solution can be proven using a minimum source-sink cut as demonstrated by Ford and Fulkerson's max-flow min-cut theorem [6].

In the following, we will distinguish between combinatorial algorithms, e.g., algorithms that are increasing the flow value on augmenting paths, and linear programming based approaches. Of course, also the simplex algorithm for solving linear programs can be considered to be combinatorial, but it may have an exponential running time. Polynomial-time algorithms for solving linear programs like the ellipsoid method or inner points algorithms are rather purely numerical approaches. Nevertheless, combinatorially easy problems and good polyhedral descriptions are closely related [14].

Tardos [15] presents a linear programming algorithm that is independent of the size of numbers in the right-hand side of the constraints and in the objective. She concludes that the maximum-value multi-commodity flow problem is solvable in strongly polynomial time using linear programming. For the undirected case with two commodities a combinatorial algorithm is known and there is a max-flow min-cut theorem which is similar to the one for the single commodity case [10]. Also for this case, it is possible to find an half-integral solution if the capacities are integral [10]. If, additionally, the vertices are even, an integral solution can be found [13]. In general, finding an integral solution of the maximum-value flow problem for two or more commodities is an $\mathcal{NP}$-complete problem [5].

Furthermore, also single-commodity network flow problems tend to become $\mathcal{NP}$-complete when additional constraints are added. For example, if a length bound for the used paths is added, it is even $\mathcal{NP}$-complete to compute a feasible path decomposition out of a feasible edge flow [2]. Confluent flows where the routing options in a vertex are limited cannot be approximated with arbitrary precision in polynomial time [4]. Hence, the computational complexity of the oriented flow problem is not clear a priori.

The routing of automated guided vehicles has been studied especially in a dynamic setting [8]. The cycle embedding problem [3] is a subproblem that appears in rolling stock rotation planning for railways. One first detects cycles (rotations) in a coarse graph, which does not model orientations. The cycle embedding problem is to regain the same cycles in a finer graph that correctly models orientations. If it is restricted to standard arcs, it can be regarded as a special case of the directed oriented maximum flow problem in this paper.

1.3 Our contribution

In this paper, we equip network flows with orientations. In Section 2, we demonstrate how we incorporate the orientations into the networks. Subsequently, we present the problem formulation for the oriented maximum flow problem which uses a graph expansion to keep track of the orientations. In Section 4, we examine the maximum oriented flow problem with respect to properties of maximum-value single-commodity and multi-commodity flow problems. In particular, we establish a dual bound and show that it is unlikely that there is a combinatorial polynomial-time algorithm that solves the problem. Instead, we show that there is a fully polynomial-time approximation scheme in Section 5.

2 Problem Input

As we want to incorporate orientations into the maximum flow problem, we require a flow network $\mathcal{N} = (G, u, s, t)$, where $G = (V, A)$ is a directed graph, $u : A \to \mathbb{R}_+$ is the capacity function and s, t denote the source and the sink, respectively. Additionally, we define a finite set S, the *set of orientations* and o_s, $o_t \in S$, the *orientation of the source* and the *orientation of the sink*. We also have to specify how the orientation changes at a vertex v.

For every pair of an incoming arc e_1 and an outgoing arc e_2 of v, the *orientation transition function* $r_v : \delta^-(v) \times \delta^+(v) \times S \to 2^S \setminus \{\emptyset\}$, $r_v(e_1, e_2, o) \mapsto S^*$ specifies the set of possible orientations S^* of the outgoing flow on e_2 for the fraction of the incoming flow on e_1 with orientation o. Here, $\delta^-(v)$ and $\delta^+(v)$ denote the set of incoming and outgoing arcs of a vertex $v \in V$, respectively. We collect the orientation transitions functions of all vertices in R. The functions can also be given via a matrix representation. For each vertex v, we have a $(\delta^-(v) \times S) \times (\delta^+(v) \times S)$ matrix with binary entries describing whether the transition is possible.

▶ **Definition 1.** Given a flow network $\mathcal{N} = (G, u, s, t)$ and S, o_s, o_t and R as described above, $\mathcal{N}^* = (G, u, s, t, S, o_s, o_t, R)$ is called an *oriented network*.

While the above definition is very flexible, it is also tedious to specify all required parameters. If there are just two orientations, i.e., $|S| = 2$, it is often more convenient to use the following alternative definition of the orientation transition function. Declare a map $\hat{r}_v : \delta^-(v) \times \delta^+(v) \to \{-1, 0, 1\}$ for each vertex $v \in V(G)$ and let $\hat{r}_v(e_1, e_2)$ specify whether the orientation changes when the flow passes from e_1 to e_2. We define that 1 indicates no alteration of the orientation, -1 indicates an alteration, and 0 means that the orientation may be altered.

This is a restriction of the previous definition, e.g., let $S = \{+, -\}$ and $\hat{r}_v(e_1, e_2) = -1$. This can easily be modeled using r_v by setting $r_v(e_1, e_2, +) = \{-\}$ and $r_v(e_1, e_2, -) = \{+\}$. But if the negative flow did not change the orientation when passing v, i.e., $r_v(e_1, e_2, +) = \{-\}$ and $r_v(e_1, e_2, -) = \{-\}$, this could not be modeled using $\hat{r}_v$, because the symmetry is lost. However, such a situation did not occur in our container applications. That is, there is no situation in which an AGV has to perform a motion that will result in a certain orientation regardless of the initial orientation. Hence, the restricted orientation transition may also suffice for modeling many other applications.

3 Graph Expansion Model

We propose a model that makes use of graph expansion to represent oriented flows. In this model, every arc carries only flow of one orientation. Thus, the expanded graph, let us call it $G^\#$, contains several copies of every arc of the original graph G, that stems from the oriented network $\mathcal{N}^* = \{G, u, s, t, S, o_s, o_t, R\}$. Although these arc copies are distinct, they share a common capacity. We shall call them *partner arcs*. A second kind of arcs will model the orientation transition. Their head and tail correspond to the same vertex in G. Therefore, we shall call them *internal arcs*. To avoid confusion, we make the following definition.

▶ **Definition 2.** A digraph is a tuple $G = (V, A, h, t)$ that consists of the finite vertex set V, the finite arcs set A and the functions $\mathrm{h}, \mathrm{t} : A \to V$ that associate each arc $e \in A$ with a head $\mathrm{h}(e)$ and a tail $\mathrm{t}(e)$. We require $\mathrm{h}(e) \neq \mathrm{t}(e)$ $\forall\, e \in A$ and $\mathrm{h}(e_1) = \mathrm{h}(e_2)$, $\mathrm{t}(e_1) = \mathrm{t}(e_2)$, $e_1, e_2 \in A \Rightarrow e_1 = e_2$, i.e., we only consider simple digraphs.

▶ Remark. It is sufficient to consider simple digraphs, as loops or parallel arcs can be replaced by two arcs and an artificial vertex that preserves the orientation.

Let us now specify in detail how to obtain an expanded graph $G^\#$. An example is provided in Figure 2. We start with the set of partner arcs $A_P^\# = A(G) \times S$. We define the projections $\mathrm{proto}_P : A_P^\# \to A(G)$, $(a, o) \mapsto a$ and $\mathrm{orient} : A_P^\# \to S$, $(a, o) \mapsto o$ that map a partner arc to its *prototype* or orientation, respectively. The arcs that share their capacity with a certain

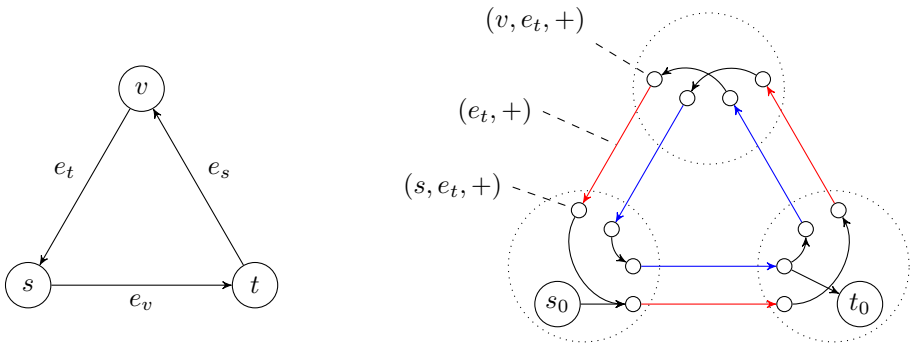

Figure 2 Example of an orientation-expanded graph with two orientations ($S = \{+, -\}$). On the left-hand side, the original graph with all arc capacities equal to 1, source orientation $o_s = +$, sink orientation $o_t = -$, and orientation transition functions $\hat{r}_s(e_t, e_v) = \hat{r}_t(e_v, e_s) = 1$ and $\hat{r}_v(e_s, e_t) = -1$. On the right-hand side the corresponding expanded network is shown where the two orientation layers are visualized by different arc colors (+: red, −: blue). To demonstrate our notation, the red arc on the upper left and its incident vertices are labeled according to our definitions.

arc $a \in A_P^\#$ are called the *partners of* a and we define $\text{partner}(a) = \{e \in A_P^\# \mid \text{proto}(e) = \text{proto}(a)\}$. The vertex set of $G^\#$is defined as

$$V(G^\#) = \bigcup_{a \in A(G)} \big(\{\text{h}(a), \text{t}(a)\} \times \{a\} \times S\big) \cup \{s_0, t_0\},$$

i.e., it consists of a super source and a super sink and the heads and tails of the partner arcs. For $a \in A_P^\#$ the head is given by $(\text{h}(\text{proto}_P(a)), \text{proto}_P(a), \text{orient}(a))$ and the corresponding tail is $(\text{t}(\text{proto}_P(a)), \text{proto}_P(a), \text{orient}(a))$. In Figure 2, the partner arcs are depicted in red and blue, since there are two orientations.

The black arcs in Figure 2 are the internal arcs. Before we can define the set of internal arcs $A_I^\#$, we have to make a few preliminary definitions. The function $\text{proto}_V : V(G^\#) \to V(G)$, $(v, \cdot, \cdot) \mapsto v \ \forall v \in V \setminus \{s, t\}, s_0 \mapsto s, t_0 \mapsto t$ associates every vertex of $G^\#$with its *prototype* in G. Let us denote all vertices of $G^\#$that have the prototype $v \in V(G)$ by a meta vertex $v^\# = \{u \in V(G^\#) \,|\, \text{proto}_V(u) = v\}$. We define $\delta^-(v^\#) := \{e \in A_P^\# \,|\, \text{h}(e) \in v^\#\}$, $\delta^+(v^\#) := \{e \in A_P^\# \,|\, \text{t}(e) \in v^\#\}$. Then, the set of internal arcs is

$$\begin{aligned} A_I^\# = \bigcup_{v \in V(G)} & \{(e^-, e^+) \,|\, e^- \in \delta^-(v^\#),\ e^+ \in \delta^+(v^\#), \\ & \text{orient}(e^+) \in r_v\big(\text{proto}_P(e^-),\ \text{proto}_P(e^+),\ \text{orient}(e^-)\big)\} \\ & \cup \{(s, e^+) \,|\, e^+ \in \delta^+(s^\#),\ \text{orient}(e^+) = o_s\} \\ & \cup \{(e^-, t) \,|\, e^- \in \delta^-(t^\#),\ \text{orient}(e^-) = o_t\}. \end{aligned}$$

It remains to define the heads and tails of the internal arcs. For an arc of the form (e^-, e^+), the head is $\text{t}(e^+)$ and, accordingly, the tail is $\text{h}(e^-)$. Therefore, it enables the flow to transit from e^- to e^+. For an arc of the form (s, e^+) the tail is the super source s_0 and the head is $\text{t}(e^+)$. Analogously, $\text{h}(e^-)$ is the tail of an internal arc of the form (e^-, t) and the super sink t_0 is its head.

▶ **Definition 3.** Given an oriented network $\mathcal{N}^* = \{G, u, s, t, S, o_s, o_t, R\}$, we call the tuple $\mathcal{N}^\# = (G^\# = (V^\#, A_P^\# \cup A_I^\#), u, s_0, t_0)$, whose components are constructed as described above, an *expanded network*. We call G the *underlying graph* of $G^\#$.

The notion of an expanded graph enables us to formulate the maximum flow problem with orientations as a linear optimization problem.

▶ **Definition 4.** Given an expanded network $\mathcal{N}^\# = (G^\# = (V^\#, A^\#), u, s_0, t_0)$ and its underlying graph $G = (V, A)$, the *directed maximum oriented flow problem* is given by the following linear optimization problem:

$$\begin{aligned}
\max \sum_{e\in\delta^+(s_0)} f(e) & \\
\sum_{e\in\delta^-(v)} f(e) \quad &= \sum_{e\in\delta^+(v)} f(e) && \forall\, v \in V(G^\#) \setminus \{s_0, t_0\} \\
\sum_{e^\#:\mathrm{proto}_P(e^\#)=e} f(e^\#) \quad &\leq u(e) && \forall\, e \in A(G) \qquad (1) \\
f \quad &\geq 0.
\end{aligned}$$

Any feasible solution $f \in \mathbb{R}_+^{A^\#}$ of this problem is called an *(oriented) flow of* $\mathcal{N}^\#$. It is *integral* iff., additionally, $f \in \mathbb{Z}^{A^\#}$ is true. The value of the objective function $\mathrm{val}(f) = \sum_{e\in\delta^+(s_0)} f(e)$ is called the *value of* f.

Hence, creating the structure of an expanded graph enables us to write the oriented maximum flow problem as an ordinary maximum flow problem with a *coupling constraint* (1). Although this seems like a small difference, we will see that is significant. To conclude this section, we show that the description provided by our model is efficient.

▶ **Lemma 5.** *The number of arcs of the expanded graph $G^\#$ is bounded by $k^2m^2 + (k+2)m$, where m is the number of arcs of its underlying graph G and $k = |S|$ is the number of orientations.*

Proof. We count the internal arcs within a meta-vertex $v^\#$, which is the set of all vertices in $G^\#$ that have the same prototype $v \in V(G)$, except for the arcs that are incident to s_0 or t_0. Note that every such arc connects an incoming partner arc of $v^\#$ and an outgoing partner arc. Therefore, their number is bounded by $|S||\delta^-(v)||S||\delta^+(v)|$. By the Cauchy-Schwartz inequality and the relationship of the Euclidean norm and the Manhattan norm, we obtain the following estimate:

$$\sum_{i=1}^{|V|} |\delta^-(v)||\delta^+(v)| \leq \sqrt{\sum_{i=1}^{|V|} |\delta^-(v)|^2}\sqrt{\sum_{i=1}^{|V|} |\delta^+(v)|^2} \leq \sum_{i=1}^{|V|} |\delta^-(v)| \cdot \sum_{i=1}^{|V|} |\delta^+(v)| = m^2$$

The vertices s_0 and t_0 are connected to at most m vertices each and the number of partner arcs is $k \cdot m$. Hence, there are no more than $k^2m^2 + (k+2)m$ arcs in $G^\#$. ◀

▶ **Theorem 6.** *For a fixed number of orientations, the directed oriented maximum flow problem can be solved in polynomial time.*

Proof. There is a formulation as a linear optimization problem. The variables of the problem correspond to the arcs of the expanded graph whose number is bounded by a polynomial of the input size. ◀

▶ Remark. If we do not fix the number of orientations k, the problem can still be solved in polynomial time in some cases. This depends on the encoding of the transition functions. The transition functions map to 2^S. If the values of the function are saved as k bits, the

input size is greater than k and the problem can be solved in polynomial time. However, for huge k, one may prefer to save the values of the functions as a list and, conceivably, their number of elements could be bounded by some constant $K \in \mathbb{Z}_+$ for the values of all transition functions. Then, we obtain a pseudo-polynomial running time.

4 Properties Of Oriented Flows

4.1 Integrality of Oriented Flows

▶ **Lemma 7.** *The solution of an instance of the directed oriented maximum flow problem need not be integral, even if the capacities of the underlying oriented network* $\mathcal{N}^* = (G, u, s, t, S, o_s, o_t, R)$ *are integral.*

Proof. Let G be the triangle with the vertices s, t and v and unit capacities. Choose $S = \{+, -\}$, $o_s = +$, $o_t = -$. Use e_w to denote the arc that lies opposite to the vertex w in the triangle for each $w \in \{s, t, v\}$. Then, we choose $\hat{r}_s(e_t, e_v) = 1$, $\hat{r}_t(e_v, e_s) = 1$ and $\hat{r}_v(e_s, e_t) = -1$, that is the orientation only changes at v, see Figure 2. It is easy to verify that there is a flow of value 0.5. Any quantity of flow that leaves s_0 has to pass $(e_v, +)$. So does any quantity of flow that enters t_0 have to pass $(e_v, -)$. Using (1), we make the following estimate: $2\,\mathrm{val}(f) \leq f((e_v, +)) + f((e_v, -)) \leq 1$. Therefore, the oriented flow of value 0.5 is maximum. ◀

▶ Remark. We will later show that the directed two-commodity maximum value flow problem can be solved using a similar instance of the directed oriented maximum flow problem. Even if the capacities for the former problem are integral, there may be no optimal flow which is integral or at least half-integral [11]. In fact, finding an integral maximum-value two-commodity flow is an $\mathcal{NP}$-complete problem [5]. Thus, these properties are also implied for the directed oriented maximum flow problem.

4.2 Flow Decomposition Theorem

The well-known flow decomposition theorem is a very useful tool and it can also be applied to oriented flows.

▶ **Theorem 8** (Flow Decomposition Theorem)**.** *If* $\mathcal{N}^\# = \{G^\#, u, s_0, t_0\}$ *is an expanded network and* G *its underlying graph, the following is true for any oriented flow* f *of* $\mathcal{N}^\#$*: There is a family* $\mathcal{P}$ *of paths and a family* $\mathcal{C}$ *of cycles, both in* $G^\#$*, with associated weights* $w : \mathcal{P} \cup \mathcal{C} \to \mathbb{R}_+$*, such that*

$$f(e) = \sum_{P \in \mathcal{P} \cup \mathcal{C}:\, e \in A(P)} w(P) \quad \forall\, e \in A(G^\#).$$

The number of paths and cycles that are required to obtain the above representation does not exceed the cardinality of $A(G^\#)$*. Moreover, the weight function* w *can be chosen to be integral if* f *is integral.*

The full proof is not given here. However, the strategy is the same as for the proof for ordinary flows (cf. [1]) because the capacity constraints are not used. Find an s_0-t_0 path or a cycle that carries flow in the expanded graph using depth-first search and remove it, then repeat. This procedure completely eliminates the flow on at least one arc or the excess of the source and the sink in every step.

4.3 Augmenting Paths

To check if a flow in an ordinary network is maximum, one usually considers the residual graph and the residual capacity. A path in the residual graph is called an augmenting path and its existence is a necessary and sufficient condition for the maximality of the associated flow. In this section, we will introduce these notions for oriented networks. To make the definition clearer, we will look at the crucial cases first.

Let $\mathcal{N}^\# = (G^\# = (V^\#, A_P^\# \cup A_I^\#), u, s_0, t_0)$ be an expanded network with the underlying graph $G = (V, A)$ and let f denote an oriented flow of $\mathcal{N}^\#$. To make the exposition less repetitive, all definitions in this subsection are referring to this $\mathcal{N}^\#$ and f, unless specified otherwise. We look for a strategy to improve f with respect to the objective function of the directed maximum oriented flow problem (Definition 4). A straightforward idea is to find a path P from s_0 to t_0, such that it contains no arc with with a tight capacity constraint (1). For a partner arc $a^\# \in A_P^\#$ with the prototype $a = \text{proto}_P(a^\#)$, the constraint is tight iff.

$$\sum_{e^\# \in \text{partner}(a^\#)} f(e^\#) = u(a).$$

As we will not consider such arcs, we can remove them from the residual graph. Additionally, it should be possible that an augmenting path may use a partner arc or an internal arc a that has a non-zero flow $f(a) > 0$ in the reverse direction. This models that the flow on a can be decreased. Hence, we introduce a reverse arc a', i.e., $\text{h}(a') = \text{t}(a)$ and $\text{t}(a') = \text{h}(a)$, with the residual capacity $u_f(a') = f(a)$ to model that we can decrease the flow on a using a path. Note that the residual capacity is not shared for reverse arcs, nor do they have partner arcs. To simplify the exposition, let $A'(G^\#)$ be the set of reverse arcs of $A(G^\#)$, i.e., $\exists\, a' \in A'(G^\#) : \text{h}(a') = v, \text{t}(a') = w \Leftrightarrow \exists\, a \in A(G^\#) : \text{h}(a) = w, \text{t}(a) = v$.

▶ **Definition 9.** The *residual graph of $G^\#$ with respect to f* is the graph G_f with the vertex set $V(G_f) = V(G^\#)$ and the arc set

$$A(G_f) = A(G^\#) \setminus \{a \in A_P^\# \mid \sum_{e \in \text{partner}(a)} f(e) = u(\text{proto}(a))\} \cup \{a' \in A'(G^\#) \mid f(a) > 0\}.$$

The *residual capacity function* $u_f : A(G_f) \to \mathbb{R}_+$ is defined as following. For $a \in A(G_f) \cap A(G^\#)$, we have $u_f(a) = u(a) - \sum_{e \in \text{partner}(a)} f(e)$. For $a' \in A(G_f) \cap A'(G^\#)$, we have $u_f(a') = f(a)$, a' being the reverse arc of a. An *augmenting path* is an s_0-t_0 path in G_f.

▶ **Definition 10.** Let G_f be the residual graph of $G^\#$ with respect to f and let u_f denote the residual capacity function. The function $\varphi_f : A(G_f) \to \mathbb{R}_+$ is a *(feasible) residual flow* in $\mathcal{N}^\#$ iff.

$$\sum_{e \in \delta^-_{G_f}(v)} \varphi_f(e) = \sum_{e \in \delta^+_{G_f}(v)} \varphi_f(e) \qquad \forall\, v \in V(G_f) \setminus \{s_0, t_0\}$$

$$\varphi_f(a') \le u_f(a') \qquad \forall\, a' \in A(G_f) \cap A'(G^\#)$$

$$\sum_{e^\# \in A(G_f) :\, \text{proto}(e^\#) = a} \varphi_f(e^\#) \le u_f(a) \qquad \forall\, a \in A(G).$$

The *value* of φ_f is $\text{val}(\varphi_f) = \sum_{e \in \delta^+_{G_f}(s_0)} \varphi_f(e)$.

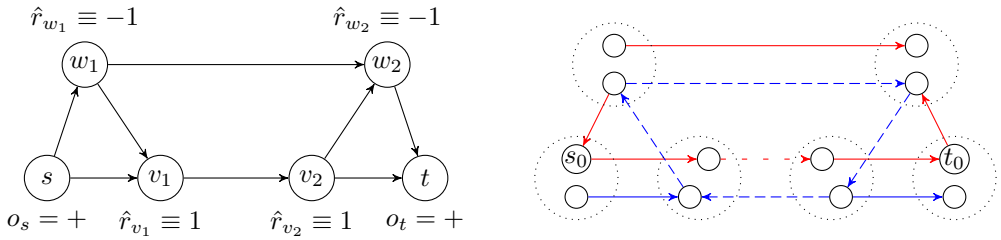

Figure 3 Network with orientations $S = \{+, -\}$ and unit capacities. The right-hand side shows a reduced version of the residual network after augmenting along s-w_1-v_1-v_2-w_2-t. The colors indicate the orientation of the flow on the arcs. The dashed red arc does not belong to the residual network. It needs to be unblocked, to obtain another augmenting path.

▶ **Definition 11.** The binary operation $\oplus$ takes an oriented flow f and a residual flow φ_f of G_f and returns a function $f \oplus \varphi_f : A(G^{\#}) \to \mathbb{R}$ with

$$f \oplus \varphi_f(a) = \begin{cases} f(a) + \varphi_f(a) & \text{if } a' \notin A(G_f) \\ f(a) + \varphi_f(a) - \varphi_f(a') & \text{if } a \in A(G_f) \text{ and } a' \in A(G_f) \\ f(a) - \varphi_f(a') & \text{if } a \notin A(G_f) \text{ and } a' \in A(G_f). \end{cases}$$

▶ **Corollary 12.** *For a flow f and a residual flow φ_f in $\mathcal{N}^{\#}$, the function $f \oplus \varphi_f$ of the above definition is an oriented flow of value $val(f \oplus \varphi_f) = val(f) + val(\varphi_f)$ in $\mathcal{N}^{\#}$.*

▶ **Lemma 13.** *Let P be an augmenting path in $\mathcal{N}^{\#}$, then f is not of maximum value.*

Proof. Let $\varepsilon > 0$ and choose $\varphi(e) = \varepsilon \ \forall e \in A(P)$ and $\varphi(e) = 0 \ \forall e \in A(G_f) \setminus A(P)$. Obviously, it fulfills the flow conservation constraint. We define a function $u_{\text{eff}} : A(P) \to \mathbb{R}_+ \setminus \{0\}$ on the arcs of P. Let $u_{\text{eff}}(e) = u_f(e)$ for $e \in A(P) \cap (A_I^{\#} \cup A_P^{\#\prime})$. If e is a partner arc, we choose $u_{\text{eff}}(e) = \frac{u_f(e)}{|A(P) \cap \text{partner}(e)|}$. Choosing $\varepsilon = \min_{e \in A(P)} u_{\text{eff}}(e)$ assures that φ fulfills the capacity constraint. Hence, it is a residual flow. Moreover, it is of positive value. Therefore, $f \oplus \varphi$ is a flow of higher value than f and, consequently, f is not of maximum value. ◀

The result of Lemma 13 is not very surprising; however, the converse does not hold. Consider the sample instance given in Figure 3. The oriented network is given on the left-hand side. By augmenting the flow along s-w_1-w_2-t and s-v_1-v_2-t by one unit each, one obtains an oriented flow of value 2, which is maximum. However, if one decides to augment along s-w_1-v_1-v_2-w_2-t in the beginning, one obtains the residual network on the right-hand side of Figure 3. Here, a few vertices have been contracted to simplify the drawing. In this case, there is no path from s_0 to t_0. To unblock the red dashed arc, which is not part of the residual network, one has to shift the flow along the blue dashed cycle (which is part of the residual network) first. In general, it is not clear along which cycles the flow has to be shifted to improve towards a maximum flow. Therefore, this strategy does not lead to a combinatorial polynomial-time algorithm and, in particular, the lack of an augmenting path in the residual network does not prove the optimality of the corresponding oriented flow.

4.4 Distance Criterion

The dual problem of the ordinary maximum flow problem is the minimum cut problem. However, since an optimal oriented flow can be fractional, even if all capacities are integral,

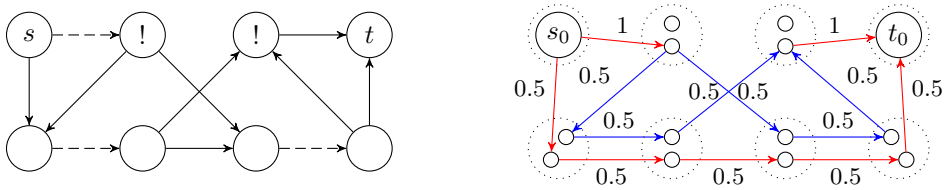

Figure 4 An oriented network with unit capacities and two orientation (+ and −). The source and sink have a positive orientation. The vertices labeled with an exclamation mark negate the orientation, all other vertices preserve it. The right hand figure shows the optimal flow in the simplified expanded network. Assigning length one half to the dashed arcs and zero to all other arcs gives a tight dual bound.

one can deduce that minimum cuts do not provide a tight dual bound. In this subsection, we will inspect the dual problem of the directed maximum oriented flow problem, which is similar to the dual problem of the maximum value multi-commodity flow problem (cf. [12]).

Let $\mathcal{N}^\# = \{G^\#, u, s_0, t_0\}$ denote an expanded network and G its underlying graph. Let $\mathcal{P}$ denote the set of all s_0-t_0 paths in $G^\#$. We construct a matrix $M \in \mathbb{Z}_+^{A(G)\times\mathcal{P}}$ with

$$M_{e,P} = \#\text{occurences of an arc with prototype } e \text{ in } P \quad \forall\, e \in A(G), P \in \mathcal{P}.$$

We can now formulate the directed maximum oriented flow problem as

$$\begin{aligned} \max \quad & \sum_{P\in\mathcal{P}} \lambda_P \\ \text{s.t.} \quad & M\lambda \le u \\ & \lambda \ge 0 \end{aligned} \qquad \text{and its dual as} \qquad \begin{aligned} \min \quad & \sum_{e\in A(G)} u(e) z_e \\ \text{s.t.} \quad & M^T z \ge \mathbb{1} \\ & z \ge 0. \end{aligned}$$

Because any oriented flow can be decomposed into paths and cycles by the flow decomposition theorem and the cycles do not contribute to the value of the flow, this primal formulation yields the same optimal value as Definition 4, although it may restrict the amount of feasible solutions. The dual variables can be interpreted as a length function $z : A(G) \to \mathbb{R}_+$ on the digraph G. Then, the constraints of the dual problem assure that no s_0-t_0 path in $G^\#$ is shorter than 1. Thereby, each partner arc has the length that is assigned to its prototype by z. In other words, the constraints assure that the distance $\text{dist}_z^\#(s_0, t_0)$ from s_0 to t_0 in $G^\#$ is at least one. Note that the dual variables can be scaled by $1/\text{dist}_z^\#(s_0, t_0)$ to ensure this (unless the distance is 0). Therefore, we obtain the bound

$$\text{dist}_z^\#(s_0, t_0)\text{val}(f) \le z^T u \quad \forall\, z \in \mathbb{R}_+^{A(G)}$$

for any oriented flow f in $\mathcal{N}^\#$. This bound is tight by strong duality. An example is given in Figure 4. If the three dashed arcs have a length of 0.5 and all other arcs have length zero, the distance from s_0 to t_0 is 1. One obtains $\text{val}(f) \le 3 \cdot 0.5$, which is a tight bound, since the optimal flow shown on the right hand of the figure has value 1.5. An ordinary minimum s-t or s_0-t_0 cut only gives an upper bound of value 2.

4.5 Relationship to Multi-Commodity Flow Problems

Consider a directed graph $G = (V, A)$ with capacities $u : A \to \mathbb{R}_+$ and two commodities (s_1, t_1) and (s_2, t_2). We claim that we can find the flow of maximum total value in this network using our formulation of the directed maximum oriented flow problem using the

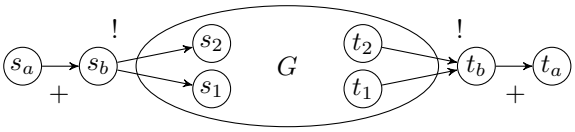

Figure 5 Construction to state the maximum-value 2-commodity flow problem as an maximum oriented flow problem.

construction shown in Figure 5. We add an artificial source vertex s_a and the vertex s_b that we use to split up the orientations. Similarly, we add t_b and t_a. The modified graph G^* contains the additional arcs (s_a, s_b), (s_b, s_1), (s_b, s_2), (t_1, t_b), (t_2, t_b) and (t_b, t_a) with a sufficient capacity. (Say, for example, an estimate of the value of the total maximum flow in the two-commodity network.) We have two orientations, $S = \{+, -\}$. The orientation of the source and the sink is positive, $o_{s_a} = o_{t_a} = +$. We set $\hat{r}_{s_b}((s_a, s_b), (s_b, s_1)) = 1$, $\hat{r}_{s_b}((s_a, s_b), (s_b, s_2)) = -1$, $\hat{r}_{t_b}((t_1, t_b), (t_b, t_a)) = 1$ and $\hat{r}_{t_b}((t_2, t_b), (t_b, t_a)) = -1$. That is, all flow that is emitted by s_1 and absorbed by t_1 has a positive orientation and, likewise, all flow that is emitted by s_2 and absorbed by t_2 has a negative orientation. All other vertices preserve the orientation, i.e., $\hat{r}_v(\cdot, \cdot) = 1 \; \forall \, v \in V(G)$. This completes the description of the oriented network $\mathcal{N}^*$. It is now easy to establish a one-to-one correspondence between an oriented flow in the expanded network $\mathcal{N}^\#$ that we obtain from $\mathcal{N}^*$ and a two-commodity flow (f_1, f_2) in the original network. In fact, since all vertices except for s_b and t_b preserve the orientation, the expanded network decomposes into two independent parts, one that carries positively oriented flow and one that carries negatively oriented flow. Both fulfill the flow conservation constraint independently. Thus, we can directly regard them as f_1 and f_2 and the coupling constraint (1) becomes the capacity constraint of the two-commodity flow.

In Section 1.2 we introduced a distinction between polynomial time algorithms that are combinatorial, e.g., the augmenting paths method for the maximum flow problem, and linear programming based approaches like the interior points method, which rely on numerics. In this sense, there is no known combinatorial polynomial-time algorithm for the directed maximum-value two-commodity flow problem. In fact, Itai [11] showed that one can solve general linear optimization problems using the directed two-commodity flow problem and some combinatorial reductions. Hence, a combinatorial polynomial-time algorithm that solves the directed maximum oriented flow problem would imply such an algorithm for general linear optimization problems. Therefore, it seems unlikely that such an algorithm exists for the directed maximum oriented flow problem.

5 Approximation

In the previous section, we argued why we do not expect that a fast combinatorial algorithm exists for the directed oriented maximum flow problem. Nevertheless, it is still possible to approximate the problem. In this section, we will apply a technique that is used by Garg and Könemann [7] to approximate the multi-commodity flow problem and related problems. Our presentation follows the exposition of their algorithm in [12].

The algorithm basically constructs a feasible solution of the dual problem (cf. Section 4.4) and a primal feasible solution. It is then possible to relate the value of the obtained flow with the optimal dual solution. In the beginning all arcs are assigned length δ. A shortest source-sink path is found in every iteration. This path is used to route flow from the source to the sink and the arc lengths along the path are increased. Once the obtained arc lengths are feasible, i.e., the distance $\text{dist}_z^\#(s_0, t_0)$ from the source to the sink in the expanded graph

Algorithm 1: Modified algorithm of Garg and Könemann

Data: $\mathcal{N}^* = (G, u, s, t, S, o_s, o_t, R)$, $\mathcal{N}^\# = (G^\# = (V^\#, A_P^\# \cup A_I^\#), u, s_0, t_0)$, $0 < \varepsilon \leq \frac{1}{2}$
Result: A flow f that is approximately maximum (within a factor of $1+\varepsilon$)
Initialization: $f \equiv 0$ is an oriented flow of $\mathcal{N}^\#$;
$\delta = (|V^\#|(1+\varepsilon))^{-\lceil \frac{5}{\varepsilon} \rceil}(1+\varepsilon)$;
$z : A(G) \to \mathbb{R}_+$, $e \mapsto \delta$ is a distance function on G; `// end of initialization`
while $dist_z^\#(s_0, t_0) < 1$ **do**
  $P \leftarrow$ shortest s_0-t_0 path in $G^\#$ with respect to z;
  $P' \leftarrow \{e \in A(G) \mid \exists e^\# \in A(P) \cap A_P^\# : \text{proto}_P(e^\#) = e\}$;
  foreach $e \in P'$ **do** $\text{visits}(e) \leftarrow |\{e^\# \in A(P) \mid \text{proto}_P(e^\#) = e\}|$;
  $u \leftarrow \min_{e \in A(P')} \frac{u(e)}{\text{visits(e)}}$;
  `// the following may lead to a violation of the capacity constraints`
  Increase f by u along P;
  foreach $e \in A(P')$ **do** $z(e) \leftarrow z(e)(1 + \varepsilon \cdot \text{visits}(e) \cdot u/u(e))$;
end
Scale f, such that no capacity constraint is violated anymore;

is at least 1, the algorithm stops. It is worth mentioning that the flow f that is obtained during the iterations of the algorithm need not fulfill the capacity constraints. Instead, after the last iteration, it is scaled such that the capacity constraints are fulfilled, that is by the inverse of $\max_{e \in V(G)} \frac{\sum_{e^\#:\text{proto}_P(e^\#)=e} f(e^\#)}{u(e)}$. A detailed listing is given in Algorithm 1.

To avoid a bare repetition, we refer to Garg and Könemann's paper [7] and [12] for the full proof of our our approximation result. Here, we only point out the modifications that have to be made to apply their method to the directed oriented maximum flow problem. In Algorithm 1, the length and capacity function are not defined on the expanded graph $G^\#$, but its underlying graph G. That is, the algorithm basically runs on G and uses $G^\#$ only to determine the paths for routing. This is possible, since the algorithm of Garg and Könemann does not consider commodities as such. It only takes possible source-sink paths, along which the flow can be routed, into account. Thus, we can use the source-sink paths given by the expanded network to determine the routes of the flow. The only problem that arises is that, in the underlying graph, it may look as if a path uses an arc several times because the flow may traverse an arc with different orientations. In this case, the length of the arc in question is increased once and, for the capacity, one has to consider the original capacity divided by the number of visits by the path.

▶ **Theorem 14.** *Algorithm 1 returns an oriented flow whose value approximates the optimal value within a factor of* $1+\varepsilon$ *and it runs in* $O(\frac{1}{\varepsilon^2}|A(G)|\log(|V^\#|)T)$ *time, where* T *is the time required to compute a shortest* s_0-t_0 *path in* $G^\#$*. That is, there is a fully polynomial-time approximation scheme for the directed maximum oriented flow problem.*

6 Conclusion

Incorporating orientations into network flow makes the problem of computing maximum flows slightly harder. The problem is still in $\mathcal{P}$, but since the absence of an augmenting path does not guarantee optimality anymore, there is no obvious combinatorial algorithm. Oriented flows are often a subproblem, e.g., in a container terminal we also have to decide

where to store the containers, in which order they are loaded on the ship, et cetera. Hence, one the one hand, one will most likely use mixed integer programming or other sophisticated techniques anyway. On the other hand, oriented flows are no obvious candidate for a decomposition and a fast subroutine. The structure of oriented flows on undirected networks, where we can also apply linear programming for solving, remains even more nebulous and requires further investigations, since expanded networks are no successful approach in this case. In further research, we will also study dynamic flow versions of this problem, since many practical applications require a time dependent handling of flow units.

References

1. Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Upper Saddle River, New Jersey, 1993.
2. Georg Baier, Thomas Erlebach, Alexander Hall, Ekkehard Köhler, Petr Kolman, Ondřej Pangrác, Heiko Schilling, and Martin Skutella. Length-bounded cuts and flows. *ACM Trans. Algorithms*, 7(1):4:1–4:27, 2010.
3. Ralf Borndörfer, Marika Karbstein, Julika Mehrgahrdt, Markus Reuther, and Thomas Schlechte. The cycle embedding problem. In *Operations Research Proceedings 2014*, pages 465 – 472, 2016.
4. Daniel Dressler and Martin Strehler. Polynomial-time algorithms for special cases of the maximum confluent flow problem. *Discrete Applied Mathematics*, 163:142–154, 2014.
5. Shimon Even, Alon Itai, and Adi Shamir. On the Complexity of Timetable and Multi-Commodity Flow Problems. In *FOCS*, pages 184–193. IEEE Computer Society, 1975.
6. Lester R. Ford and Delbert R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8(3):399–404, 1956.
7. Naveen Garg and Jochen Koenemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. *SIAM Journal on Computing*, 37(2):630–652, 2007.
8. Ewgenij Gawrilow, Ekkehard Köhler, Rolf H. Möhring, and Björn Stenzel. Dynamic routing of automated guided vehicles in real-time. In Willi Jäger and Hans-Joachim Krebs, editors, *Mathematics — Key Technology for the Future*, pages 165–178. Springer, 2008.
9. Andrew V. Goldberg and Robert E. Tarjan. Efficient maximum flow algorithms. *Communications of the ACM*, 57(8):82–89, 2014.
10. Te C. Hu. Multi-commodity network flows. *Operations research*, 11(3):344–360, 1963.
11. Alon Itai. Two-Commodity Flow. *JACM*, 25(4):596–611, 1978.
12. Bernhard Korte and Jens Vygen. *Combinatorial Optimization: Theory and Algorithms*, volume 21 of *Algorithms and Combinatorics*. Springer, 4th edition, 2008.
13. Bruce L. Rothschild and Andrew B. Whinston. Feasibility of two commodity network flows. *Operations Research*, 14(6):1121–1129, 1966.
14. Alexander Schrijver. *Combinatorial Optimization – Polyhedra and Efficiency*, volume 24 of *Algorithms and Combinatorics*. Springer, 2003.
15. Eva Tardos. A strongly polynomial algorithm to solve combinatorial linear programs. *Operations Research*, 34(2):250–256, 1986.