A Faster Parametric Search for the Integral Quickest Transshipment Problem

Mariia Anapolska
Combinatorial Optimization, RWTH Aachen University, Germany

Dario van den Boom[1]
Combinatorial Optimization, RWTH Aachen University, Germany

Christina Büsing
Combinatorial Optimization, RWTH Aachen University, Germany

Timo Gersing
Combinatorial Optimization, RWTH Aachen University, Germany

Abstract

Algorithms for computing fractional solutions to the quickest transshipment problem have been significantly improved since Hoppe and Tardos first solved the problem in strongly polynomial time. For integral solutions, however, no structural improvements on their algorithm itself have yet been proposed. Runtime improvements are limited to general progress on submodular function minimization (SFM), which is an integral part of Hoppe and Tardos' algorithm. In fact, SFM constitutes the main computational load of the algorithm, as the runtime is blown up by using it within Megiddo's parametric search algorithm. We replace this part of Hoppe and Tardos' algorithm with a more efficient routine that solves only a linear number of SFM and, in contrast to previous techniques, exclusively uses minimum cost flow algorithms within Megiddo's parametric search. Our approach improves the state-of-the-art runtime from $\tilde{\mathcal{O}}(m^4k^{15})$ down to $\tilde{\mathcal{O}}(m^2k^5 + m^4k^2)$[2], where k is the number of terminals and m is the number of arcs.

2012 ACM Subject Classification Theory of computation → Network flows

Keywords and phrases Flow over time, dynamic transshipment, quickest transshipment, parametric submodular functions, efficient algorithms

Digital Object Identifier 10.4230/LIPIcs.ESA.2025.112

Related Version *Full Version*: https://arxiv.org/abs/2505.12975 [1]

Funding This work is co-funded by the German research council (DFG) – project number 442047500 – Collaborative Research Center Sparsity and Singular Structures (SFB 1481) and by the German research council (DFG) Research Training Group 2236 UnRAVeL.

Acknowledgements We would like to express our gratitude to the anonymous reviewers for their valuable feedback and constructive comments, which have greatly contributed to improving the presentation of this paper.

1 Introduction

Network flows over time, also referred to as dynamic flows, extend classical static network flows by a time component. They provide a powerful tool for modeling real-world problems in traffic engineering, building evacuation, and logistics. Over the last decades, a wide range of optimization problems dealing with flows over time have been studied. The *maximum flow over time problem* was studied in the seminal work of Ford and Fulkerson [3], who showed

[1] Corresponding Author
[2] We use $\tilde{\mathcal{O}}$ to suppress polylogarithmic terms.

that the problem can be solved in polynomial time by a reduction to the static minimum cost flow problem. The *quickest flow problem* asks for the minimum time-horizon T^* such that a provided demand of $D \in \mathbb{N}$ units of flow can be sent from a single source s to a single sink t. While a straightforward approach is to combine an algorithm for maximum flows over time with a parametric search algorithm [2], recent results have shown that cost scaling algorithms for minimum cost flows can be modified in order to efficiently compute quickest flows [7, 10].

The *quickest transshipment problem* generalizes the quickest flow problem by allowing for *supply* and *demand* at multiple *sources* and *sinks*. It is one of the most fundamental problems in the field of network flows over time and, as recently stated by Skutella [14], "arguably the most difficult flow over time problem that can still be solved in polynomial time." As in the quickest flow problem, our goal is to send the required flow from sources to sinks while simultaneously minimizing the time horizon.

Similar to the quickest flow problem, the quickest transshipment problem can be solved by determining the minimum time horizon via parametric search. Using this idea, Hoppe and Tardos [4] showed that the quickest transshipment problem can be solved in strongly polynomial time, that is, their algorithm's runtime is polynomially bounded in the number of nodes n, number of arcs m, and the combined number of sources and sinks k.

Recently, faster algorithms have been developed. Notably, Schlöter, Skutella, and Tran [12] proposed an algorithm with a time complexity of $\tilde{\mathcal{O}}(m^2k^5 + m^3k^3 + m^3n)$. Unfortunately, these performance improvements come at the expense of fractional solutions, which may be undesirable for applications that do not allow flow particles to be disassembled. While some of the results speed up the search for the optimal time horizon, no improvements have yet been proposed for finding integral flows over time. Hence, the state-of-the-art complexity for the integral quickest transshipment problem remains unchanged at $\tilde{\mathcal{O}}(m^4k^{15})$ [12].

Our Contribution

We propose improvements to Hoppe and Tardos' algorithm by replacing two computationally expensive subroutines in which submodular function minimization is combined with Megiddo's parametric search. By doing so, we reduce the state-of-the-art runtime for the integral quickest transshipment problem from $\tilde{\mathcal{O}}(m^4k^{15})$ to $\tilde{\mathcal{O}}(m^2k^5 + m^4k^2)$. This narrows the gap to the fractional quickest transshipment problem, which can be solved in $\tilde{O}(m^2k^5 + m^3k^3 + m^3n)$ time using the algorithm by Schlöter, Skutella, and Tran [12].

2 Preliminaries

Given a directed graph $G = (V, A)$ with vertices V and arcs A, we define a *dynamic network* as a triple $\mathcal{N} = (G, u, \tau)$ with *capacity* $u_a \in \mathbb{N}$ and *transit time* $\tau_a \in \mathbb{N}_0$ for each arc $a \in A(\mathcal{N})$. For a given dynamic network $\mathcal{N}$, the set $V(\mathcal{N})$ denotes the network's nodes, while $A(\mathcal{N})$ refers to the network's arcs. Throughout this paper, we denote the number of nodes $|V(\mathcal{N})|$ by n and the number of arcs $|A(\mathcal{N})|$ by m.

A *flow over time* is a family of functions $f_a \colon [0, T) \to \mathbb{R}_{\geq 0}$, representing the in-flow rates for each arc $a \in A(\mathcal{N})$ for every point in time until the end of the *time horizon* $T \in \mathbb{N}$. The $f_a(\theta)$-many flow units entering arc a at time $\theta \in [0, T)$ arrive at time $\theta + \tau_a$ at the end node of a. While we only require f_a to satisfy weak flow conservation, Hoppe and Tardos' algorithm computes an optimal solution satisfying even the strict flow conservation, meaning that all flow units entering a non-sink node are immediately forwarded via an outgoing arc. We refer the reader to [13] for more details on flows over time.

Note that, in contrast to Hoppe and Tardos [4] who defined a flow over time as a static flow in a time-expanded network, we use a continuous-time model. However, all prerequisites for their algorithm directly translate from the discrete- to continuous-time model as shown in [14], meaning that it can also be implemented for the continuous-time model.

For the *dynamic transshipment problem*, we have a triple $(\mathcal{N}, b, T)$ comprising a dynamic network $\mathcal{N}$, a *balance function* $b\colon V(\mathcal{N}) \to \mathbb{Z}$ with $\sum_{v \in V(\mathcal{N})} b(v) = 0$, and a time horizon T. The balances describe how supply and demand are distributed across the network. A node with positive balance $b(v) > 0$ is a *source*, while a node with negative balance is a *sink*. Let S^+ denote the set of sources, S^- the set of sinks, and $S = S^+ \cup S^-$ the set of *terminals*. A dynamic transshipment instance is *feasible* if there exists a flow over time sending the supply from the sources to the sinks such that all demands are satisfied.

▶ **Definition 1.** *Given a subset of terminals $X \subseteq S$, the* maximum out-flow $o(X)$ out of X *is the value of the maximum flow over time from the sources $S^+ \cap X$ to the sinks $S^- \setminus X$.*

The central feasibility criterion states that the *net balance* $b(X) \coloneqq \sum_{v \in X} b(v)$ must not exceed the maximum out-flow $o(X)$ for every $X \subseteq S$.

▶ **Theorem 2** (Feasibility Criterion [4])**.** *The dynamic transshipment instance $(\mathcal{N}, b, T)$ is feasible if and only if $v(X) \coloneqq o(X) - b(X) \geq 0$ for all $X \subseteq S$.*

We call a set $X \subseteq S$ with $v(X) < 0$ a *violated set*. In order to determine the feasibility of a given dynamic transshipment instance, it suffices to show that no violated set exists. However, while the value of $o(X)$, and thus of $v(X)$, can be computed with the Ford-Fulkerson algorithm for maximum flows over time, avoiding the enumeration of all subsets $X \subseteq S$ is not obvious. Fortunately, we can use the *submodularity* of $o\colon 2^S \to \mathbb{N}_0$ and $v\colon 2^S \to \mathbb{Z}$.

▶ **Definition 3** (Submodular Function)**.** *A function $f\colon 2^S \to \mathbb{R}$ over a finite* ground set S *is a* submodular function *if for all $X, Y \subseteq S$ it holds that*

$$f(X \cup Y) + f(X \cap Y) \leq f(X) + f(Y), \tag{1}$$

or, equivalently, if for all $s \in S$ and $X \subseteq Y \subseteq S \setminus \{s\}$ it holds that

$$f(Y \cup \{s\}) - f(Y) \leq f(X \cup \{s\}) - f(X). \tag{2}$$

Given a submodular function f over S, we call a set $X^* \in \operatorname{argmin}_{X \subseteq S} f(X)$ a *minimizer* of f. It is well-known that the set of minimizers of a submodular function is closed under union and intersection. Therefore, there always exists a *minimal minimizer* and a *maximal minimizer*, which are the intersection and union of all minimizers, respectively.

In the context of *submodular function minimization (SFM)*, a submodular function f is typically provided in form of an *evaluation oracle* with complexity $\mathcal{O}(\text{EO})$. The performance of algorithms is measured in the number of oracle calls required for finding a minimizer. The fastest strongly polynomial algorithm for submodular function minimization is due to Lee, Sidford and Wong [6] with a runtime of $\mathcal{O}(k^3 log^2 k \cdot \mathcal{O}(\text{EO}) + k^4 log^{\mathcal{O}(1)} k)$ for $k = |S|$.

For our purpose, the evaluation oracle computes a maximum out-flow $o(X)$ out of terminals $X \subseteq S$ using the Ford-Fulkerson algorithm for maximum flows over time. To this end, we introduce a super-source s^+ and a super-sink s^- and connect them to sources $s \in S^+ \cap X$ and sinks $t \in S^- \setminus X$, respectively, via infinite-capacity, zero-transit arcs. Then the maximum out-flow $o(X)$ is the value of the maximum flow over time from s^+ to s^-. This value can be computed using a static min-cost flow in $\mathcal{O}(m \log n(m + n \log n))$ or $\tilde{\mathcal{O}}(m^2)$ time via Orlin's algorithm [9]. We abbreviate this runtime as $\mathcal{O}(\text{MCF}(n, m))$.

Consequently, determining whether a dynamic transshipment instance $(\mathcal{N}, b, T)$ is feasible takes $\mathcal{O}(k^3 log^2 k \cdot \mathrm{MCF}(n, m) + k^4 log^{\mathcal{O}(1)} k)$ or $\tilde{\mathcal{O}}(m^2k^3)$ time, where $k = |S|$ is the number of terminals. To improve readability, we denote the time it takes to check if an instance with k terminals, n nodes and m arcs is feasible by $\mathcal{O}(\mathrm{SFM}(k, n, m))$.

The Algorithm by Hoppe and Tardos

The algorithm by Hoppe and Tardos [4] was the first strongly polynomial algorithm computing quickest transshipments and remains the most efficient one for integral solutions. In the following, we assume that the minimum time horizon T^* is provided and focus on the algorithm's segment that computes an integral dynamic transshipment. The algorithm relies on the concept of tight orders.

▶ **Definition 4** (Tight Set and Order). *A set of terminals $X \subseteq S$ is called* tight *if $o(X) = b(X)$ holds. Let $\preceq$ be a total order on S. We call $\preceq$* tight *if $\{t' \in S \mid t' \preceq t\}$ is tight for all $t \in S$.*

The general idea by Hoppe and Tardos is to construct an equivalent dynamic transshipment instance for which a tight order exists.

▶ **Theorem 5** (Reduction to Lex-Max Flows [4]). *Given a dynamic transshipment instance $(\mathcal{N}, b, T)$ with a tight order $\preceq$ over the terminals S, an integral dynamic transshipment satisfying b can be computed as a lex-max flow over time in $\mathcal{O}(k\,\mathrm{MCF}(n, m))$ time.*

We refer to [14] for more details on lex-max flows over time. Although computing the integral flow is quite efficient, transforming $(\mathcal{N}, b, T)$ into an equivalent instance $(\mathcal{N}', b', T)$ with a tight order is computationally demanding. We propose improvements to this transformation in Section 5. Before that, we briefly discuss the approach by Hoppe and Tardos.

First, one adds a new terminal $\check{s}$ with $b(\check{s}) = b(s)$ for every terminal $s \in S$ and then sets $b(s) = 0$. Note that the set of terminals S now only contains the nodes $\check{s}$. By adding infinite-capacity, zero-transit arcs $(\check{s}, s)$ for $\check{s} \in S^+$ and $(s, \check{s})$ for $\check{s} \in S^-$, one ensures that the resulting dynamic transshipment instance is equivalent to the original instance (cf. Figure 1).

To construct an instance admitting a tight order, we iteratively shift supply / demand from terminals $\check{s}$ to new terminals $\hat{s}$. In this paper, we call the $\check{s}$ *drained terminals* and the $\hat{s}$ *filled terminals*. We refer the reader to Hoppe and Tardos [4] and the description in the extended paper [1] for a general overview of the algorithm. We focus on the two subroutines, MaximizeAlpha and MinimizeDelta, which compute the amount of supply / demand that is shifted from $\check{s}$ to $\hat{s}$.

When discussing both subroutines, we assume that a feasible dynamic transshipment instance $(\mathcal{N}, b, T)$ is given, with a set of terminals S consisting of the drained terminals $\check{s}$, added in the first step, and all filled terminals $\hat{s}$ that were introduced in previous iterations. Furthermore, we are given two tight sets $Q \subset R \subseteq S$ and a drained terminal $\check{s} \in R \setminus Q$ satisfying $o(Q \cup \{\check{s}\}) > b(Q \cup \{\check{s}\})$.

Note that the instance $(\mathcal{N}, b, T)$, its set of terminals S as well as Q, R, and $\check{s}$ are the input of our subroutines and vary between iterations. In contrast, every subroutine adds a new filled terminal. However, since both subroutines are discussed in isolation, we simplify the notation by referring to both filled terminals as $\hat{s}$. Throughout this paper, we define $\hat{Q} := Q \cup \{\hat{s}\}$ and $\hat{R} := R \cup \{\hat{s}\}$. Similar to Hoppe and Tardos, we only discuss the case in which $\check{s}$ is a source, since the treatment of sinks is symmetrical and is sketched in the extended paper [1].

Figure 1 A dynamic transshipment instance $(\mathcal{N}, b, T)$ after initialization with terminal $\check{s}$ (left) and the corresponding α-parametric instance $(\mathcal{N}^\alpha, b^\alpha, T)$ for MAXIMIZEALPHA (right).

Capacity-Parametric Instances

The first subroutine starts with tight sets Q and R and a drained source $\check{s} \in R \setminus Q$ for which $Q \cup \{\check{s}\}$ is not tight. Its aim is to reassign as much supply as possible to a new filled source $\hat{s}$. In doing so, we capacitate the out-flow of $\hat{s}$ such that $\hat{Q}$ is tight and the transformed instance remains feasible.

▶ **Definition 6** (α-Parametric Dynamic Network). *Given a dynamic network $\mathcal{N}$, a drained source $\check{s} \in S^+$ and a parameter $\alpha \in \mathbb{N}_0$, the* α-parametric network *$\mathcal{N}^\alpha$ is constructed by adding a new filled source $\hat{s}$ and connecting it to s via an α-capacity, zero-transit arc $(\hat{s}, s)$.*

Let $o^\alpha : 2^{S \cup \{\hat{s}\}} \to \mathbb{N}_0$ be the parametric counterpart of the maximum out-flow as in Definition 1 in the parametric network $\mathcal{N}^\alpha$. Using this notation, we define *α-parametric dynamic transshipment instances* analogously to Hoppe and Tardos [4]. The construction of an α-parametric instance is illustrated in Figure 1.

▶ **Definition 7** (α-Parametric Dynamic Transshipment Instance). *Given a feasible dynamic transshipment instance $(\mathcal{N}, b, T)$, two tight sets of terminals $Q \subset R \subseteq S$, a drained source $\check{s} \in R \setminus Q$ and a parameter $\alpha \in \mathbb{N}_0$, the corresponding* α-parametric dynamic transshipment instance *$(\mathcal{N}^\alpha, b^\alpha, T)$ consists of the following components.*

- *An α-parametric dynamic network $\mathcal{N}^\alpha$ as in Definition 6.*
- *An α-parametric balance function b^α with $b^\alpha(t) = b(t)$ for all terminals $t \in S \setminus \{\check{s}\}$, $b^\alpha(\hat{s}) = \Delta^\alpha$, and $b^\alpha(\check{s}) = b(\check{s}) - \Delta^\alpha$, where $\Delta^\alpha := o^\alpha(\hat{Q}) - o^\alpha(Q)$.*

We call a parameter value $\alpha \in \mathbb{N}_0$ *feasible* if the corresponding α-parametric dynamic transshipment instance $(\mathcal{N}^\alpha, b^\alpha, T)$ is feasible. Determining whether a value α is feasible is equivalent to checking if a *violated* set $X \subseteq S \cup \{\hat{s}\}$ exists. Recall that this can be done by minimizing the parametric submodular function $v^\alpha(X) = o^\alpha(X) - b^\alpha(X)$. A subroutine of the algorithm by Hoppe and Tardos finds a maximum feasible parameter value $\alpha \in \mathbb{N}_0$.

▶ **Definition 8** (MAXIMIZEALPHA). *Given an α-parametric dynamic transshipment instance, find the maximum feasible $\alpha \in \mathbb{N}_0$.*

We denote the maximum feasible parameter value by α^*. Hoppe and Tardos concluded that, given a feasibility oracle for $(\mathcal{N}^\alpha, b^\alpha, T)$ taking $\mathcal{O}(\mathrm{SFM}(k, n, m))$ time, the value α^* can be found in $\mathcal{O}(\log(nU_{\max}) \cdot \mathrm{SFM}(k, n, m))$ time, where $U_{\max} := \max_{a \in A(\mathcal{N})} u_a$. In addition, one can combine Megiddo's parametric search [8] with the combinatorial SFM algorithm by Orlin and Iwata [5] to achieve a strongly polynomial runtime of $\tilde{\mathcal{O}}(m^4 k^{14})$. We mainly improve upon the strongly polynomial approach.

Figure 2 A dynamic transshipment instance $(\mathcal{N}, b, T)$ after initialization with terminal $\check{s}$ (left) and the corresponding δ-parametric instance $(\mathcal{N}^\delta, b^\delta, T)$ for MinimizeDelta (right).

Transit-Parametric Instances

The second subroutine is closely related to MaximizeAlpha. Again, we start with tight sets Q and R and a drained source $\check{s} \in R \setminus Q$ for which $Q \cup \{\check{s}\}$ is not tight. The aim is to reassign as much supply as possible to a new filled source $\hat{s}$. In doing so, we set the transit time for the flow out of $\hat{s}$ such that $\hat{Q}$ is tight and the transformed instance remains feasible.

▶ **Definition 9** (δ-Parametric Dynamic Network). *Given a dynamic network $\mathcal{N}$, a drained source $\check{s} \in S^+$ and a parameter $\delta \in \mathbb{N}_0$, the* δ-parametric network $\mathcal{N}^\delta$ *is constructed by adding a terminal $\hat{s}$ and connecting it to s via a unit-capacity, δ-transit arc $(\hat{s}, s)$.*

Analogously to o^α, we denote the maximum out-flow of a subset of terminals $X \subset S \cup \{\hat{s}\}$ by $o^\delta(X)$. Next, we combine this parametric variant of our dynamic network with a parametric balance function to form a *δ-parametric dynamic transshipment instance* (cf. Figure 2).

▶ **Definition 10** (δ-Parametric Dynamic Transshipment Instance). *Given a feasible dynamic transshipment instance $(\mathcal{N}, b, T)$, two tight sets of terminals $Q \subset R \subseteq S$, a drained source $\check{s} \in R \setminus Q$ and a parameter $\delta \in \mathbb{N}_0$, a* δ-parametric dynamic transshipment instance $(\mathcal{N}^\delta, b^\delta, T)$ *consists of the following components.*

- *A δ-parametric dynamic network $\mathcal{N}^\delta$ as given in Definition 9.*
- *A δ-parametric balance function b^δ with $b^\delta(t) = b(t)$ for all terminals $t \in S \setminus \{\check{s}\}$, $b^\delta(\hat{s}) = \Delta^\delta$, and $b^\delta(\check{s}) = b(\check{s}) - \Delta^\delta$, where $\Delta^\delta := o^\delta(\hat{Q}) - o^\delta(Q)$.*

Again, a parameter value $\delta \in \mathbb{N}_0$ is *feasible* if the corresponding parametric dynamic transshipment instance $(\mathcal{N}^\delta, b^\delta, T)$ is feasible. This can be checked by minimizing the parametric submodular function $v^\delta(X) = o^\delta(X) - b^\delta(X)$. Again, we call a set $X \subseteq S$ with $v^\delta(X) < 0$ *violated*. This yields the following parametric search problem.

▶ **Definition 11** (MinimizeDelta). *Given a δ-parametric dynamic transshipment instance, find the minimum feasible $\delta \in \mathbb{N}_0$.*

Again, δ^* denotes the minimum feasible parameter value. Hoppe and Tardos showed that the optimal value δ^* can be found in $\mathcal{O}(\log(T) \cdot \mathrm{SFM}(k, n, m))$ time, or in strongly polynomial time of $\tilde{\mathcal{O}}(m^4 k^{14})$ using the parametric search of Megiddo [8].

The algorithm by Hoppe and Tardos performs a total of k iterations, each of which calls both subroutines MaximizeAlpha and MinimizeDelta once. Therefore, the current version of the algorithm by Hoppe and Tardos takes $\mathcal{O}(k \cdot \log(nTU_{\max}) \cdot \mathrm{SFM}(k, n, m))$ time if both subroutines are implemented with a binary search, while Megiddo's parametric search results in a runtime of $\tilde{\mathcal{O}}(m^4 k^{15})$. In the following sections, we improve the runtime of each iteration by introducing better parametric search algorithms for both subroutines.

3 Restricting the Domains of Violated Sets

Both problems MaximizeAlpha and MinimizeDelta introduced in the previous section rely on minimizing submodular functions to determine whether a parameter value is feasible. Even with state-of-the-art algorithms for SFM, this remains a computationally expensive subroutine that scales poorly with the number of terminals in the ground set.

We show that, if our parametric instance is infeasible, there always exists a violated set $X \subseteq S \cup \{\hat{s}\}$ satisfying $\hat{Q} \subset X \subset \hat{R}$, which allows us to restrict the ground sets of our parametric submodular functions v^α and v^δ. This provides a practical improvement and establishes the foundation for the following sections.

▶ **Lemma 12.** *Let $X \subseteq S \cup \{\hat{s}\}$ be a violated set for an α-parametric dynamic transshipment instance with respect to a drained source $\check{s}$. Then $\hat{s} \in X$ and $\check{s} \notin X$. The same applies to δ-parametric dynamic transshipment instances.*

Proof. We only prove the statement for α-parametric instances, as the reasoning is analogous for δ-parametric instances. Let $X \subseteq S \cup \{\hat{s}, \check{s}\}$ be an arbitrary subset of terminals. Given that we assume the transshipment instance to be feasible, we have $v(X) \geq 0$. Remember that the maximum out-flow $o^\alpha(X)$ is the value of a maximum flow over time from a super-source s^+ to a super-sink s^-, where infinite-capacity, zero-transit arcs are used to connect s^+ to every source in $S^+ \cap X$ and to connect every sink in $S^- \setminus X$ to s^-. Using this definition, we derive the following observations regarding $o(X)$:

O1 If X contains neither $\hat{s}$ nor $\check{s}$, then maximum out-flow $o^\alpha(X)$ coincides with the out-flow $o(X)$ in the original instance.

O2 If X contains $\check{s}$, then all flow traveling from s^+ to s can bypass the α-capacity arc $(\hat{s}, s)$ and move along the infinite-capacity arcs $(s^+, \check{s})$ and $(\check{s}, s)$ instead. As a consequence, we have $o^\alpha(X) = o(X)$.

We refer the reader back to Figure 1 for a visual intuition. Combining these observations with the parametric balances $b^\alpha(\hat{s}) = \Delta^\alpha$ and $b^\alpha(\check{s}) = b(\check{s}) - \Delta^\alpha$, we prove Lemma 12 by considering the following three complementary cases.

(I) If $\hat{s} \in X$ and $\check{s} \in X$, then it follows that

$$b^\alpha(X) = b^\alpha(X \setminus \{\hat{s}, \check{s}\}) + b^\alpha(\hat{s}) + b^\alpha(\check{s}) \overset{\text{Def. of } b^\alpha}{=} b^\alpha(X \setminus \{\hat{s}, \check{s}\}) + b(\check{s}) = b(X \setminus \{\hat{s}\}) = b(X).$$

Together with Observation 2 we obtain $v^\alpha(X) = v(X) \geq 0$, meaning that X cannot be a violated set.

(II) If $\hat{s} \notin X$ and $\check{s} \in X$, then it follows that $b^\alpha(X) = b(X) - \Delta^\alpha < b(X)$, which, together with Observation 2, implies that X cannot be a violated set since $v^\alpha(X) \geq v(X) \geq 0$.

(III) If $\hat{s} \notin X$ and $\check{s} \notin X$, then $b^\alpha(X) = b(X)$ and therefore X cannot be a violated set as it follows from Observation 1 that $v^\alpha(X) = o^\alpha(X) - b^\alpha(X) = o(X) - b(X) = v(X) \geq 0$.

Hence, X can only be a violated set of $(\mathcal{N}^\alpha, b^\alpha, T)$ if $\hat{s} \in X$ and $\check{s} \notin X$. ◀

Hoppe and Tardos [4] proved the property from Lemma 12 for the special case of the infeasible parameter value $\delta^* - 1$ and used it to show that there exists a set X satisfying $\hat{Q} \subset X \subset \hat{R}$ which is violated for $\delta^* - 1$ and tight for δ^*. We employ similar arguments to show that this also holds for all infeasible $\alpha, \delta \in \mathbb{N}_0$.

▶ **Lemma 13.** *Let $\alpha \in \mathbb{N}_0$ be an infeasible parameter value for MaximizeAlpha. Then there is a minimizer X of v^α with $\hat{Q} \subset X \subset \hat{R}$. Analogously, let $\delta \in \mathbb{N}_0$ be an infeasible parameter value for MinimizeDelta. Then there is a minimizer X of v^δ with $\hat{Q} \subset X \subset \hat{R}$.*

Proof. We only prove the statement for MAXIMIZEALPHA, as the proof for MINIMIZEDELTA is analogous. Let X^* be an arbitrary minimizer of v^α with $v^\alpha(X) < 0$. It follows from Lemma 12 that $\hat{s} \in X^*$ and $\check{s} \notin X^*$. Next, we show that the set $X := Q \cup (X^* \cap \hat{R})$ is also a minimizer of v^α. For this, we analyze the tightness of the sets Q, $\hat{Q}$, and $\hat{R}$:

- Q was chosen to be a tight set for $(\mathcal{N}, b, T)$. This also does not change in the parametric instance as $\hat{s}, \check{s} \notin Q$ and hence $v^\alpha(Q) = v(Q) = 0$.
- $\hat{Q}$ is tight, since $b^\alpha(\hat{Q}) = b^\alpha(Q) + \Delta^\alpha \overset{Q \text{ tight}}{=} o^\alpha(Q) + o^\alpha(\hat{Q}) - o^\alpha(Q) = o^\alpha(\hat{Q})$.
- $\hat{R}$ is tight because R is tight and $\hat{s}, \check{s} \in \hat{R}$ directly imply $o^\alpha(\hat{R}) = o(R) = b(R) = b^\alpha(\hat{R})$.

Having shown that Q and $\hat{R}$ are tight sets, we study how tight sets and minimizers of v^α behave under union and intersection. For this purpose, let $Y \in \{Q, \hat{R}\}$. It follows from submodularity of v^α that

$$v^\alpha(X^* \cup Y) + v^\alpha(X^* \cap Y) \leq v^\alpha(X^*) + v^\alpha(Y) = v^\alpha(X^*) < 0. \tag{3}$$

Note that both summands on the left hand side are non-positive, since otherwise one of them would be smaller than the minimum value $v^\alpha(X^*)$. Hence, either both summands are negative, or one is equal to zero while the other equals the minimum value of v^α. In other words, exactly one of the following properties must hold:

(1) Both $X^* \cup Y$ and $X^* \cap Y$ are violated sets of v^α.
(2) Either $X^* \cup Y$ or $X^* \cap Y$ is a minimizer, while the other set is tight.

We apply this case distinction to the cases where $Y = Q$ and $Y = \hat{R}$:

- Let $Y = \hat{R}$, then Lemma 12 states that a violated set cannot contain $\check{s}$, implying that $v^\alpha(X^* \cup \hat{R}) \geq 0$ since $\check{s} \in \hat{R}$. This means that $X^* \cap \hat{R}$ is a minimizer of v^α.
- Consider $Y = Q$ and the minimizer $X^* \cap \hat{R}$. We rule out $Q \cap (X^* \cap \hat{R}) = Q \cap X^*$ as a violated set since $\hat{s} \notin Q \cap X^*$. Therefore, $Q \cup (X^* \cap \hat{R})$ is a minimizer.

Finally, observe that the minimizer $X = Q \cup (X^* \cap \hat{R})$ not only satisfies $\hat{Q} \subseteq X \subseteq \hat{R}$ because $\hat{s} \in X^* \cap \hat{R}$ but also $\hat{Q} \subset X \subset \hat{R}$ since $\hat{Q}$ and $\hat{R}$ are tight. ◀

Lemma 13 allows us to determine feasibility of a parameter value by minimizing the *restricted functions* $\tilde{v}^\alpha\colon 2^{\hat{R} \setminus \hat{Q}} \to \mathbb{Z}$ and $\tilde{v}^\delta\colon 2^{\hat{R} \setminus \hat{Q}} \to \mathbb{Z}$ with $\tilde{v}^\alpha(X) := v^\alpha(\hat{Q} \cup X)$ and $\tilde{v}^\delta(X) := v^\delta(\hat{Q} \cup X)$. For future use, we define the functions $\tilde{o}^\alpha$, $\tilde{o}^\delta$, $\tilde{b}^\alpha$, and $\tilde{b}^\delta$ analogously.

This brings us to the main result in this section.

▶ **Corollary 14.** *A parameter value $\alpha \in \mathbb{N}_0$ is feasible for MAXIMIZEALPHA if and only if $\tilde{v}^\alpha(X) \geq 0$ for every $X \subseteq \hat{R} \setminus \hat{Q}$. Analogously, a parameter value $\delta \in \mathbb{N}_0$ is feasible for MINIMIZEDELTA if and only if $\tilde{v}^\delta(X) \geq 0$ for every $X \subseteq \hat{R} \setminus \hat{Q}$.*

The obvious advantage of this result is the reduction of the domain of the submodular functions of interest to $\hat{R} \setminus \hat{Q}$. Although we have $R = S$ and $Q = \emptyset$ at the beginning, the difference becomes smaller over the course of Hoppe and Tardos' algorithm, until both sets are eventually equal. This yields a significant practical speed-up in later iterations of the algorithm. Moreover, the reduction brings further practical and theoretical advantages, which we discuss in the following section.

4 Strong Map Sequences

We concluded the previous section with a useful restriction of our submodular functions v^α and v^δ to sets X with $\hat{Q} \subset X \subset \hat{R}$, while maintaining the guarantee that our parametric instance is feasible if and only if the minimizer of our restricted function is not violated. Building on this, we show that both restricted functions satisfy the *strong map property* (also known as *decreasing differences property*).

▶ **Definition 15** (Strong Map Property [11]). *Let* $f_1, f_2 \colon 2^E \to \mathbb{R}$ *be two submodular functions defined over the same finite ground set* E*. We write* $f_1 \sqsupset f_2$*, or* $f_2 \sqsubset f_1$*, if* $X \subseteq Y \subseteq E$ *implies*

$$f_1(Y) - f_1(X) \leq f_2(Y) - f_2(X).$$

The relation is called a strong map*. Submodular functions* $f_1, f_2, \dots, f_k$ *form a* strong map sequence *if* $f_1 \sqsupset f_2 \sqsupset \cdots \sqsupset f_k$*.*

Recall that the minimizers of submodular functions are closed under union and intersection, meaning that there exists a unique minimal and maximal minimizer for every submodular function. A result by Topkis [15] relates the minimal and maximal minimizers of functions that form strong map sequences.

▶ **Lemma 16** (Minimizers for Strong Map Sequences [11]). *Let* $f_1, f_2 \colon 2^E \to \mathbb{R}$ *be two submodular functions over the same ground set* E *with* $f_1 \sqsubset f_2$*. Denote by* X_1^{min} *and* X_1^{max} *the minimal and maximal minimizer of* f_1*, respectively, while* X_2^{min} *and* X_2^{max} *are the minimal and maximal minimizer of* f_2*, respectively. Then* $X_1^{min} \subseteq X_2^{min}$ *and* $X_1^{max} \subseteq X_2^{max}$*.*

Lemma 16 already gives an intuition for how the strong map property can be used for the parametric search problems MaximizeAlpha and MinimizeDelta: each choice of α or δ gives us a new submodular function, and, if these functions form strong map sequences, the strong map property implies that we can only encounter at most $|S|$ distinct minimal minimizers for a monotonic sequence of parameter values.

Unfortunately, neither v^α nor v^δ satisfy the strong map property as is[2]. However, this changes when considering the restricted functions $\tilde{v}^\alpha$ and $\tilde{v}^\delta$ defined in Section 3. Recall that these functions are defined over the ground set $\hat{R} \setminus \hat{Q}$, and that their definition ensures that $\hat{s}$ is implicitly added, while $\check{s}$ is excluded.

▶ **Lemma 17.** *Let* $0 \leq \alpha \leq \alpha'$*. Then* $\tilde{v}^\alpha \sqsubset \tilde{v}^{\alpha'}$*.*

Proof. Our argument is structured as follows: We first prove that $\tilde{o}^\alpha \sqsubset \tilde{o}^{\alpha+1}$ holds for all $\alpha \in \mathbb{N}_0$ and then use this result to prove that $\tilde{v}^\alpha$ also forms a strong map sequence with $\tilde{v}^\alpha \sqsubset \tilde{v}^{\alpha+1}$. The general claim $\tilde{v}^\alpha \sqsubset \tilde{v}^{\alpha'}$ then immediately follows by transitivity of $\leq$.

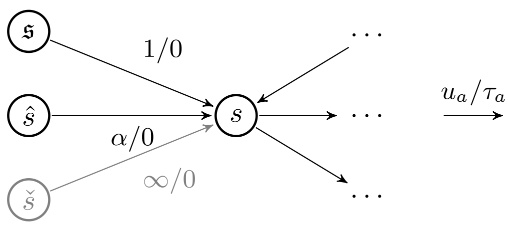

Figure 3 The auxiliary network $\mathfrak{N}$ for MaximizeAlpha.

Let $\alpha \in \mathbb{N}_0$ be arbitrary but fixed. We construct an auxiliary network $\mathfrak{N}$ with nodes $V(\mathfrak{N}) := V(\mathcal{N}^\alpha) \cup \{\mathfrak{s}\}$ and arcs $A(\mathfrak{N}) := A(\mathcal{N}^\alpha) \cup \{\mathfrak{a} := (\mathfrak{s}, s)\}$, where s is the source that was replaced by $\check{s}$ in the first phase of the algorithm. For the new arc, we set $u_{\mathfrak{a}} = 1$ and $\tau_{\mathfrak{a}} = 0$. An example network for a given source $\hat{s}$ is depicted in Figure 3.

[2] In nontrivial cases, the nested sets $\emptyset$, $\{\hat{s}\}$, and $\{\hat{s}, \check{s}\}$ satisfy $o^{\alpha+1}(\{\hat{s}\}) - o^{\alpha+1}(\emptyset) > o^\alpha(\{\hat{s}\}) - o^\alpha(\emptyset)$ and $o^{\alpha+1}(\{\hat{s}, \check{s}\}) - o^{\alpha+1}(\{\hat{s}\}) < o^\alpha(\{\hat{s}, \check{s}\}) - o^\alpha(\{\hat{s}\})$. The same holds for o^δ. Consequently, our proofs for Lemma 17 and Lemma 18 fail for the general functions v^α and v^δ.

Let $\mathfrak{o}(X)$ be the max out-flow function for network $\mathfrak{N}$ restricted to the domain $\hat{R} \setminus \hat{Q} \cup \{\mathfrak{s}\}$. Notice how adding $\mathfrak{s}$ to a set of terminals $X \subseteq \hat{R} \setminus \hat{Q} \cup \{\mathfrak{s}\}$ with $\hat{s} \in X$ and $\check{s} \notin X$ is equivalent to increasing α by one, as flow cannot bypass the arcs $(\mathfrak{s}, s)$ and $(\hat{s}, s)$ through $(\check{s}, s)$. Formally, it holds for every set X with $X \subseteq \hat{R} \setminus \hat{Q}$ that

1. $\mathfrak{o}(X) = \tilde{o}^{\alpha}(X)$, and
2. $\mathfrak{o}(X \cup \{\mathfrak{s}\}) = \tilde{o}^{\alpha+1}(X)$.

Clearly, the function $\mathfrak{o}$ is also submodular. Hence, given two sets $X \subseteq Y \subset \hat{R} \setminus \hat{Q}$, the definition of submodularity in Equation (2) can be restated as

$$\mathfrak{o}(Y \cup \{\mathfrak{s}\}) - \mathfrak{o}(X \cup \{\mathfrak{s}\}) \leq \mathfrak{o}(Y) - \mathfrak{o}(X). \tag{4}$$

Combining Equation (4) with all our previous observations, it follows that the sets X and Y satisfy

$$\begin{aligned} \tilde{o}^{\alpha+1}(Y) - \tilde{o}^{\alpha+1}(X) &\overset{\text{Obs. 2}}{=} \mathfrak{o}(Y \cup \{\mathfrak{s}\}) - \mathfrak{o}(X \cup \{\mathfrak{s}\}) \\ &\overset{\text{Eq. (4)}}{\leq} \mathfrak{o}(Y) - \mathfrak{o}(X) \\ &\overset{\text{Obs. 1}}{=} \tilde{o}^{\alpha}(Y) - \tilde{o}^{\alpha}(X). \end{aligned}$$

Hence, we have shown that $\tilde{o}^{\alpha} \sqsubset \tilde{o}^{\alpha+1}$ holds for every parameter value α. Finally, recall our definition of $\tilde{v}^{\alpha}$ as $\tilde{v}^{\alpha}(X) = \tilde{o}^{\alpha}(X) - \tilde{b}^{\alpha}(X)$ for any $X \subseteq \hat{R} \setminus \hat{Q}$. Due to the strong map property of $\tilde{o}^{\alpha}$, we get for all $X \subseteq Y \subseteq \hat{R} \setminus \hat{Q}$ that

$$\begin{aligned} \tilde{v}^{\alpha+1}(Y) - \tilde{v}^{\alpha}(Y) &= \tilde{o}^{\alpha+1}(Y) - \tilde{o}^{\alpha}(Y) + \tilde{b}^{\alpha}(Y) - \tilde{b}^{\alpha+1}(Y) \\ &= \tilde{o}^{\alpha+1}(Y) - \tilde{o}^{\alpha}(Y) + \Delta^{\alpha} - \Delta^{\alpha+1} \\ &\leq \tilde{o}^{\alpha+1}(X) - \tilde{o}^{\alpha}(X) + \Delta^{\alpha} - \Delta^{\alpha+1} \\ &= \tilde{o}^{\alpha+1}(X) - \tilde{o}^{\alpha}(X) + \tilde{b}^{\alpha}(X) - \tilde{b}^{\alpha+1}(X) \\ &= \tilde{v}^{\alpha+1}(X) - \tilde{v}^{\alpha}(X), \end{aligned}$$

therefore proving that $\tilde{v}^{\alpha} \sqsubset \tilde{v}^{\alpha+1}$. Thus, by transitivity of $\leq$, our claim $\tilde{v}^{\alpha} \sqsubset \tilde{v}^{\alpha'}$ holds. ◀

Next, we utilize a similar argument to prove that $\tilde{v}^{\delta}$ forms a strong map sequence. Here, we rely on our specific definition of the δ-parametric network $\mathcal{N}^{\delta}$.

▶ **Lemma 18.** *Let $0 \leq \delta' \leq \delta$. Then $\tilde{v}^{\delta} \sqsubset \tilde{v}^{\delta'}$.*

Proof. The general approach is identical to the proof of Theorem 17. That is, we show that $\tilde{o}^{\delta}$ forms a strong map sequence with $\tilde{o}^{\delta-1} \sqsupset \tilde{o}^{\delta}$ for $\delta \geq 1$. Afterwards, we transfer this result to $\tilde{v}^{\delta-1}$ and $\tilde{v}^{\delta}$. The general claim then directly follows from the transitivity of $\leq$.

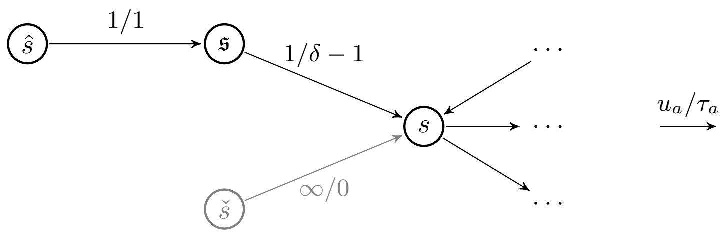

Figure 4 The auxiliary network $\mathfrak{N}$ for MinimizeDelta.

Again, let $\delta \in \mathbb{N}$ be arbitrary but fixed. We construct an auxiliary network $\mathfrak{N}$ with nodes $V(\mathfrak{N}) := V(\mathcal{N}^\delta) \cup \{\mathfrak{s}\}$ and arcs $A(\mathfrak{N}) := (A(\mathcal{N}^\delta) \setminus \{(\hat{s}, s)\}) \cup \{\mathfrak{a} := (\mathfrak{s}, s), \hat{a} := (\hat{s}, \mathfrak{s})\}$, s is the source that was replaced by $\check{s}$ in the first step of the algorithm. Additionally, let $u_{\hat{a}} = u_{\mathfrak{a}} = 1$, $\tau_{\mathfrak{a}} = \delta - 1$ and $\tau_{\hat{a}} = 1$. An example for a source $\hat{s}$ is given in Figure 4.

Again, $\mathfrak{o}(X)$ denotes the maximum out-flow function for this network restricted to the domain $\hat{R} \setminus \hat{Q} \cup \{\mathfrak{s}\}$. In this construction, we have replaced the arc $(\hat{s}, s)$ by a path consisting of arcs $(\hat{s}, \mathfrak{s})$ and $(\mathfrak{s}, s)$ with a combined transit time of δ. As a consequence, the maximum out-flow $\mathfrak{o}(X)$ remains unchanged for sets $X \subseteq \hat{R} \setminus \hat{Q}$. If $\mathfrak{s} \in X$ and $\hat{s} \in X$, on the other hand, both sources compete for access to the arc $(\mathfrak{s}, s)$. Then, the flow out of X is maximized by sending all flow from $\mathfrak{s}$. Formally, it holds for every set $X \subseteq \hat{R} \setminus \hat{Q}$ that

1. $\mathfrak{o}(X) = \tilde{o}^\delta(X)$, and
2. $\mathfrak{o}(X \cup \{\mathfrak{s}\}) = \tilde{o}^{\delta-1}(X)$

These observations plus the second definition of submodularity from Equation (2) imply for all sets $X \subseteq Y \subseteq \hat{R} \setminus \hat{Q}$ that

$$\begin{aligned}
\tilde{o}^{\delta-1}(Y) - \tilde{o}^{\delta-1}(X) &\overset{\text{Obs. 2}}{=} \mathfrak{o}(Y \cup \{\mathfrak{s}\}) - \mathfrak{o}(X \cup \{\mathfrak{s}\}) \\
&\overset{\text{Eq. (2)}}{\leq} \mathfrak{o}(Y) - \mathfrak{o}(X) \\
&\overset{\text{Obs. 1}}{=} \tilde{o}^\delta(Y) - \tilde{o}^\delta(X).
\end{aligned}$$

Therefore, the function $\tilde{o}^\delta$ forms a strong map sequence with $\tilde{o}^{\delta-1} \sqsupset \tilde{o}^\delta$. Finally, by the same arguments as in the proof of Lemma 17, the parametric function $\tilde{v}^\delta$ forms a strong map sequence with $\tilde{v}^\delta \sqsupset \tilde{v}^{\delta'}$ as claimed. ◀

In the following section, we use the strong map property and the resulting nested sequences of minimizers to construct new parametric search algorithms for v^α and v^δ .

5 An Improved Parametric Search

In this section, we adapt Schlöter's [11] parametric search algorithm for the single-source or single-sink quickest transshipment problem to MaximizeAlpha and MinimizeDelta. We start by showing that $\tilde{v}^\alpha$ and $\tilde{v}^\delta$ are monotonic in their respective parameters.

▶ **Lemma 19.** *Given a set of terminals $X \subseteq \hat{R} \setminus \hat{Q}$, both maps $\alpha \mapsto \tilde{v}^\alpha(X)$ and $\delta \mapsto \tilde{v}^\delta(X)$ are monotonic in the parameters α and δ, respectively. That is, we have $\tilde{v}^\alpha(X) \geq \tilde{v}^{\alpha+1}(X)$ and $\tilde{v}^\delta(X) \leq \tilde{v}^{\delta+1}(X)$ for all $\alpha, \delta \in \mathbb{N}_0$.*

Proof. Let $\alpha, \delta \in \mathbb{N}_0$ be arbitrary but fixed parameter values. We set $Z := X \cup \hat{Q}$ and decompose $\tilde{v}^\alpha(X)$ (and, analogously, $\tilde{v}^\delta(X)$) as

$$\begin{aligned}
\tilde{v}^\alpha(X) &= \tilde{o}^\alpha(X) - \tilde{b}^\alpha(X) \qquad (5)\\
&= \tilde{o}^\alpha(X) - \big(b(Z \setminus \{\hat{s}\}) + \Delta^\alpha\big) \\
&\overset{\text{Def. 7}}{=} \tilde{o}^\alpha(X) - \big(b(Z \setminus \{\hat{s}\}) + o^\alpha(\hat{Q}) - o^\alpha(Q)\big) \\
&= \tilde{o}^\alpha(X) - \big(b(Z \setminus \{\hat{s}\}) + \tilde{o}^\alpha(\emptyset) - o^\alpha(Q)\big) \\
&= \tilde{o}^\alpha(X) - \tilde{o}^\alpha(\emptyset) + \big(o^\alpha(Q) - b(Z \setminus \{\hat{s}\})\big).
\end{aligned}$$

Observe that we can treat the term $o^\alpha(Q) - b(Z \setminus \{\hat{s}\})$ as constant: since $\hat{s}, \check{s} \notin Q$, the out-flow $o^\alpha(Q)$ is identical for all $\alpha \in \mathbb{N}_0$ (or $\delta \in \mathbb{N}_0$). Similarly, the value of $b(Z \setminus \{\hat{s}\})$ does not depend on α or δ.

We now prove monotonicity using this simplification. For MaximizeAlpha, the monotonicity $\tilde{v}^{\alpha}(X) \geq \tilde{v}^{\alpha+1}(X)$ holds if and only if $\tilde{o}^{\alpha}(X) - \tilde{o}^{\alpha}(\emptyset) \geq \tilde{o}^{\alpha+1}(X) - \tilde{o}^{\alpha+1}(\emptyset)$ is true, which follows directly from the strong map property $\tilde{o}^{\alpha} \sqsubset \tilde{o}^{\alpha+1}$ shown in the proof of Lemma 17. Similarly, replacing α with δ in Equation (5), it follows from $\tilde{o}^{\delta} \sqsupset \tilde{o}^{\delta+1}$ that $\tilde{o}^{\delta}(X) - \tilde{o}^{\delta}(\emptyset) \geq \tilde{o}^{\delta+1}(X) - \tilde{o}^{\delta+1}(\emptyset)$ and thus $\tilde{v}^{\delta}(X) \leq \tilde{v}^{\delta+1}(X)$ holds. ◀

The monotonicity and strong map property of $\tilde{v}^{\alpha}$ and $\tilde{v}^{\delta}$ allow us to introduce new parametric search algorithms for MaximizeAlpha (cf. Algorithm 1) and MinimizeDelta (cf. Algorithm 2). In their core, they follow a rather simple approach similar to algorithms by Schlöter, Skutella, and Tran [12, 11]: we start with an infeasible parameter value α_1 or δ_1 and a corresponding minimizer X_1 of $\tilde{v}^{\alpha_1}$ or $\tilde{v}^{\delta_1}$. Possible initial values are $\alpha_1 := \alpha_{\max} = nU_{\max}$ and $\delta_1 := 0$ [4]. Next, we alternate between two steps *jump* and *check* until the current minimizer X_i is no longer violated. In the *jump* step, we compute the largest parameter value α_{i+1} or the smallest parameter value δ_{i+1} such that the previous minimizer X_i is no longer violated. Note that this step is, in essence, an integral parametric min-cost flow problem with one parametrized arc. Afterwards, the *check* step finds a minimizer X_{i+1} for the new value α_{i+1} and δ_{i+1}.

In the check step, we use the results of the previous sections in order to restrict the search for minimizers to $X_{i+1} \subset X_i$. We will show in the following that this restriction is actually feasible. Note that the restriction not only reduces the search space in each iteration to $|X_i|$ terminals, but also limits MaximizeAlpha and MinimizeDelta to at most $|S|$ iterations, since the length of the chain $X_1 \supset X_2 \supset \dots$ is limited by $|\hat{R} \setminus \hat{Q}| \leq |S|$.

▶ **Theorem 20.** *Algorithm 1 computes the maximum feasible $\alpha^* \in \mathbb{N}_0$ for MaximizeAlpha; Algorithm 2 computes the minimum feasible $\delta^* \in \mathbb{N}_0$ for MinimizeDelta. Both algorithms terminate after at most $|S|$ iterations of the while loop.*

Proof. We first show by induction that if α_i is infeasible, then there exists a minimizer X_i of $\tilde{v}^{\alpha_i}$ with $X_i \subset X_{i-1}$, where $X_0 = \hat{R} \setminus \hat{Q}$.

By definition, X_1 is a minimizer of $\tilde{v}^{\alpha_1}$. Let α_i and α_{i+1} be infeasible, and assume that X_i is a minimizer of $\tilde{v}^{\alpha_i}$. We have $\alpha_{i+1} < \alpha_i$, since otherwise $\tilde{v}^{\alpha_{i+1}}(X_i) \overset{\text{Lem. 19}}{\leq} \tilde{v}^{\alpha_i}(X_i) < 0$ contradicts the choice of α_{i+1} as feasible for X_i. Together with Lemma 17, this implies the relation $\tilde{v}^{\alpha_i} \sqsupset \tilde{v}^{\alpha_{i+1}}$. Therefore, we have $X_{i+1}^{\min} \subseteq X_i^{\min} \subseteq X_i$ due to Lemma 16, with $X_j^{\min}$ being the minimal minimizer of $\tilde{v}^{\alpha_j}$ for $j \in \{i, i+1\}$. Since α_{i+1} is infeasible with $\tilde{v}^{\alpha_{i+1}}(X_{i+1}^{\min}) < 0$, and, by definition, $\tilde{v}^{\alpha_{i+1}}(X_i) \geq 0$, we even have $X_{i+1}^{\min} \subset X_i$. Hence, $X_{i+1}^{\min}$ is a feasible choice for X_{i+1} in Algorithm 1.

Algorithm 1 Parametric Search for MaximizeAlpha.

Data: Tight sets $\hat{Q} \subset \hat{R} \subseteq S \cup \{\hat{s}\}$, submodular function $\tilde{v}^{\alpha} \colon 2^{\hat{R} \setminus \hat{Q}} \to \mathbb{Z}$, infeasible upper bound $\alpha_{\max}$
Result: Maximum feasible $\alpha \in \mathbb{N}_0$

$\alpha_1 \leftarrow \alpha_{\max}$, $X_1 \leftarrow$ Minimizer of $\tilde{v}^{\alpha_1}$
$i \leftarrow 1$
while $\tilde{v}^{\alpha_i}(X_i) < 0$ **do**
  $\alpha_{i+1} \leftarrow$ Maximum $\alpha \geq 0$ with $\tilde{v}^{\alpha}(X_i) \geq 0$ ▷ Jump step
  $X_{i+1} \leftarrow$ Minimizer $X \subset X_i$ of $\tilde{v}^{\alpha_{i+1}}$ ▷ Check step
  $i \leftarrow i + 1$
end
return α_i

Algorithm 2 Parametric Search for MinimizeDelta.

Data: Tight sets $\hat{Q} \subset \hat{R} \subseteq S \cup \{\hat{s}\}$, submodular function $\tilde{v}^{\delta}\colon 2^{\hat{R}\setminus\hat{Q}} \to \mathbb{Z}$
Result: Minimum feasible $\delta \in \mathbb{N}$
$\delta_1 \leftarrow 0$, $X_1 \leftarrow$ Minimizer of $\tilde{v}^{\delta_1}$
$i \leftarrow 1$
while $\tilde{v}^{\delta_i}(X_i) < 0$ **do**
  $\delta_{i+1} \leftarrow$ Minimum $\delta \geq 0$ with $\tilde{v}^{\delta}(X_i) \geq 0$ ▷ Jump step
  $X_{i+1} \leftarrow$ Minimizer $X \subset X_i$ of $\tilde{v}^{\delta_{i+1}}$ ▷ Check step
  $i \leftarrow i+1$
end
return δ_i

Recall that Algorithm 1 terminates after at most $|S|$ iterations. Let α_{i^*} be the value returned by the algorithm, and let X_{i^*} be the minimizer generated in the final iteration i^*. If α_{i^*} were not feasible, then X_{i^*} would be a minimizer with $v^{\alpha_{i^*}}(X_{i^*}) < 0$. However, since the algorithm terminates, it follows that α_{i^*} is feasible.

To see that α_{i^*} is maximum, recall that $\alpha_1 = \alpha_{\max}$ is infeasible, and thus the jump step is executed at least once. By construction in the jump, the value α_{i^*} is maximum such that the previous minimizer X_{i^*-1} is no longer violated. Hence, for any $\alpha > \alpha_{i^*}$, we have $v^{\alpha}(X_{i^*-1}) < 0$, so α is infeasible. Therefore, α_{i^*} is the optimal solution to MaximizeAlpha.

The proof for Algorithm 2 is analogous. ◀

The remainder of the section is devoted to the runtime analysis. Our main improvement is due to the upper bound $k = |S|$ on the number of iterations the algorithms execute.

▶ **Theorem 21.** *Algorithm 1 can be implemented to solve MaximizeAlpha in strongly polynomial time of* $\mathcal{O}(k[\mathrm{SFM}(k,n,m) + \mathrm{MCF}(n,m)^2])$ *and in weakly polynomial time of* $\mathcal{O}(k[\mathrm{SFM}(k,n,m) + \log(nU_{max}) \cdot \mathrm{MCF}(n,m)])$.

Proof. The runtime is determined by the two main steps performed in each iteration:

- Jump can be implemented using Megiddo's parametric search [8] or binary search over the range $[0, nU_{\max}]$ in conjunction with a minimum cost flow algorithm. The former results in a runtime of $\mathcal{O}(\mathrm{MCF}(n,m)^2)$, the latter in $\mathcal{O}(\log(nU_{\max}) \cdot \mathrm{MCF}(n,m))$.
- Check minimizes the submodular function v^{α} on the restricted domain. In the worst case, this takes $\mathcal{O}(\mathrm{SFM}(k,n,m))$ time.

Together with the upper bound of k on the iterations of the while loop, the final runtime is $\mathcal{O}(k[\mathrm{SFM}(k,n,m) + \mathrm{MCF}(n,m)^2])$ or $\mathcal{O}(k[\mathrm{SFM}(k,n,m) + \log(nU_{\max}) \cdot \mathrm{MCF}(n,m)])$. ◀

We obtain an analogous runtime for Algorithm 2.

▶ **Theorem 22.** *Algorithm 2 can be implemented to solve MinimizeDelta in strongly polynomial time of* $\mathcal{O}(k[\mathrm{SFM}(k,n,m) + \mathrm{MCF}(n,m)^2])$ *and in weakly polynomial time of* $\mathcal{O}(k[\mathrm{SFM}(k,n,m) + \log(T) \cdot \mathrm{MCF}(n,m)])$.

Proof. The proof is analogous to that of Theorem 21. The binary search for the minimum feasible value of δ is done on the range $[0, T]$, which yields the different logarithmic term. ◀

We conclude this section with an improved runtime for computing integral dynamic transshipments compared to the state-of-the-art runtime of $\tilde{\mathcal{O}}(m^4 k^{15})$.

▶ **Theorem 23.** *Given a dynamic transshipment instance* $(\mathcal{N}, b, T)$*, an integral quickest transshipment can be computed in* $\tilde{\mathcal{O}}(m^2k^5 + m^4k^2)$ *time.*

Proof. Hoppe and Tardos [4] already proved that their algorithm for dynamic transshipment terminates after $\mathcal{O}(k)$ iterations, each of which consists of one call of Algorithms 1 and 2. Given the strongly polynomial runtime of $\mathcal{O}\big(k(\text{SFM}(k, n, m) + \text{MCF}(n, m)^2)\big)$ for both subroutines, we obtain a worst-case complexity of $\mathcal{O}(k^2(\text{SFM}(k, n, m) + \text{MCF}(n, m)^2))$.

Suppressing the polylogarithmic terms, we have a time complexity of $\tilde{\mathcal{O}}(m^2k^3)$ for $\mathcal{O}(\text{SFM}(k, n, m))$ and of $\tilde{\mathcal{O}}(m^2)$ for $\mathcal{O}(\text{MCF}(n, m))$. Overall, we obtain an improved runtime of $\tilde{\mathcal{O}}(m^2k^5 + m^4k^3)$ time for the integral dynamic transshipment problem.

In order to compute a quickest transshipment, we have to determine the minimum time horizon first. This can be done by the method by Schlöter, Skutella, and Tran [12] in $\tilde{\mathcal{O}}(m^2k^5 + m^3k^3 + m^3n)$ time. This runtime is dominated by that of our algorithm for computing the corresponding transshipment: since we assume that the network is connected, we have $m \geq n \geq k$, and thus $\tilde{\mathcal{O}}(m^3n) \subset \tilde{\mathcal{O}}(m^4)$ and $\tilde{\mathcal{O}}(m^3k^3) \subset \tilde{\mathcal{O}}(m^4k^2)$. Overall, the time required to compute a quickest integral transshipment is

$$\tilde{\mathcal{O}}\big((m^2k^5 + m^3k^3 + m^3n) + (m^2k^5 + m^4k^2)\big) = \tilde{\mathcal{O}}(m^2k^5 + m^4k^2 + m^4) = \tilde{\mathcal{O}}(m^2k^5 + m^4k^2),$$

which shows Theorem 23. ◀

6 Conclusion and Outlook

In this paper, we propose an improved version of the algorithm by Hoppe and Tardos for the integral quickest transshipment problem. Our approach is based on more efficient parametric search algorithms using the strong map property and yields a substantial reduction of the runtime from $\tilde{\mathcal{O}}(m^4k^{15})$ to $\tilde{\mathcal{O}}(m^2k^5 + m^4k^2)$.

Our findings open room for ensuing research. In particular, the restrictions of submodular functions to suitable domains introduced in this paper may provide even better bounds on the runtime for our algorithms. Furthermore, we see potential improvements to the jump steps in Algorithms 1 and 2 that are currently based on Megiddo's parametric search and contribute a factor of $\tilde{\mathcal{O}}(m^4)$ to the runtime. This factor constitutes the remaining gap in runtime between the integral and the fractional quickest transshipment problem. In order to close this gap, future studies may focus on adapting parametric minimum cost flow algorithms akin to the algorithms by Lin and Jaillet [7] and Saho and Shigeno [10].

References

1. Mariia Anapolska, Dario van den Boom, Christina Büsing, and Timo Gersing. A faster parametric search for the integral quickest transshipment problem. *arXiv preprint arXiv:2505.12975*, 2025. doi:10.48550/arXiv.2505.12975.
2. Rainer E. Burkard, Karin Dlaska, and Bettina Klinz. The quickest flow problem. *Zeitschrift für Operations Research*, 37(1):31–58, 1993. doi:10.1007/BF01415527.
3. Lester R. Ford and Delbert R. Fulkerson. *Flows in Networks*. Princeton University Press, 1962.
4. Bruce Hoppe and Éva Tardos. The quickest transshipment problem. *Mathematics of Operations Research*, 25(1):36–62, 2000. doi:10.1287/moor.25.1.36.15211.
5. Satoru Iwata and James B Orlin. A simple combinatorial algorithm for submodular function minimization. In *Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms*, pages 1230–1237. SIAM, 2009. doi:10.1137/1.9781611973068.133.

6 Yin Tat Lee, Aaron Sidford, and Sam Chiu-Wai Wong. A faster cutting plane method and its implications for combinatorial and convex optimization. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1049–1065. IEEE, 2015. doi:10.1109/FOCS.2015.68.

7 Maokai Lin and Patrick Jaillet. On the quickest flow problem in dynamic networks–a parametric min-cost flow approach. In Piotr Indyk, editor, *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 1343–1356. SIAM, SIAM, 2015. doi:10.1137/1.9781611973730.89.

8 Nimrod Megiddo. Combinatorial optimization with rational objective functions. In *Tenth annual ACM symposium on Theory of computing (STOC)*, pages 1–12, 1978. doi:10.1145/800133.804326.

9 James Orlin. A faster strongly polynomial minimum cost flow algorithm. In *Twentieth annual ACM symposium on Theory of Computing (STOC)*, pages 377–387, 1988. doi:10.1145/62212.62249.

10 Masahide Saho and Maiko Shigeno. Cancel-and-tighten algorithm for quickest flow problems. *Networks*, 69(2):179–188, 2017. doi:10.1002/net.21726.

11 Miriam Schlöter. *Flows over time and submodular function minimization.* PhD thesis, Dissertation, Berlin, Technische Universität Berlin, 2018, 2018.

12 Miriam Schlöter, Martin Skutella, and Khai Van Tran. A faster algorithm for quickest transshipments via an extended discrete newton method. In *2022 Symposium on Discrete Algorithms (SODA)*, pages 90–102. SIAM, 2022. doi:10.1137/1.9781611977073.5.

13 Martin Skutella. An introduction to network flows over time. *Research Trends in Combinatorial Optimization: Bonn 2008*, pages 451–482, 2009. doi:10.1007/978-3-540-76796-1_21.

14 Martin Skutella. An introduction to transshipments over time. *arXiv preprint arXiv:2312.04991*, 2023. doi:10.48550/arXiv.2312.04991.

15 Donald M. Topkis. Minimizing a Submodular Function on a Lattice. *Operations Research*, 26(2):305–321, 1978. doi:10.1287/opre.26.2.305.