Unsplittable Transshipments

Srinwanti Debgupta
Institute of Mathematics, TU Berlin, Germany

Sarah Morell
Faculty of Mathematics and Computer Science, University of Bremen, Germany

Martin Skutella
Institute of Mathematics, TU Berlin, Germany

Abstract

We introduce the *Unsplittable Transshipment Problem* in directed graphs with multiple sources and sinks. An unsplittable transshipment routes given supplies and demands using at most one path for each source–sink pair. Although they are a natural generalization of single source unsplittable flows, unsplittable transshipments raise interesting new challenges and require novel algorithmic techniques. As our main contribution, we give a nontrivial generalization of a seminal result of Dinitz, Garg, and Goemans (1999) by showing how to efficiently turn a given transshipment x into an unsplittable transshipment y with $y_a < x_a + d_{\max}$ for all arcs a, where $d_{\max}$ is the maximum demand (or supply) value. Further results include bounds on the number of rounds required to satisfy all demands, where each round consists of an unsplittable transshipment that routes a subset of the demands while respecting arc capacity constraints.

2012 ACM Subject Classification Mathematics of computing → Combinatorial optimization; Mathematics of computing → Network flows

Keywords and phrases Network flow, unsplittable flow, flow augmentation

Digital Object Identifier 10.4230/LIPIcs.ICALP.2026.74

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: https://doi.org/10.48550/arXiv.2602.07230 [3]

Funding The first and third author are funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – The Berlin Mathematics Research Center MATH+ (EXC-2046/1, EXC-2046/2, project ID: 390685689).

Acknowledgements We want to thank the anonymous referees for their thoughtful comments that helped to improve the presentation of the paper.

1 Introduction

Network flows constitute one of the most fundamental classes of problems in combinatorial optimization and mathematical programming. We refer to the historical book by Ford and Fulkerson [5], the classical textbook by Ahuja, Magnanti, and Orlin [1], as well as the more recent textbook by Williamson [19] for comprehensive treatments of the subject.

Starting with the pioneering work of Kleinberg [7, 8], the past three decades have witnessed substantial progress in the study of unsplittable flows, in which the demand of each commodity must be routed along a single path from its source to its sink subject to given arc capacity constraints. Unsplittable flows are crucial in applications such as logistics, traffic, and telecommunication networks, where splitting a commodity across multiple paths may degrade the quality of service, increase operational complexity, or be infeasible altogether, as exemplified by optical networks requiring specialized equipment for path splitting and freight logistics where dividing loads across routes is often impractical.


The most general multicommodity version of the problem, in which each commodity has its own source and sink, generalizes the arc-disjoint paths problem and is notoriously difficult to approximate. If the maximum demand is smaller than the minimum arc capacity, the randomized rounding technique of Raghavan and Thompson [14] achieves a logarithmic approximation factor for minimizing congestion, while for densely embedded unit-capacity graphs, Kleinberg [7] obtains a constant-factor approximation with high probability. We refer to the survey by Kolliopoulos [10] for an overview of results on general unsplittable flows. The difficulty of the general problem has motivated extensive study of the special single source case, in which all commodities share a common source. In this setting, the underlying problem of computing a fractional (i.e., not necessarily unsplittable) flow that satisfies all demands reduces to a classical single commodity flow problem and can be solved via a maximum flow computation.

1.1 Unsplittable transshipments

We study unsplittable transshipments as an extension of single source unsplittable flows to networks with multiple sources and sinks and prescribed supplies and demands, where for each source–sink pair flow is routed along at most one path. More precisely, we are given a directed graph $D = (V, A)$ together with a vertex balance function $b\colon V \to \mathbb{R}$ satisfying $\sum_{v \in V} b(v) = 0$. A vertex v is called a *source* if $b(v) > 0$, and the set of all sources is denoted by S^+. Similarly, a vertex v is called a *sink* if $b(v) < 0$, and the set of all sinks is denoted by S^-. A flow $x \in \mathbb{R}^A_{\geq 0}$ is a *b-transshipment* if

$$\sum_{a \in \delta^{\text{out}}(v)} x_a - \sum_{a \in \delta^{\text{in}}(v)} x_a \;=\; b(v) \qquad \text{for all } v \in V,$$

where $\delta^{\text{in}}(v)$ and $\delta^{\text{out}}(v)$ denote the sets of incoming and outgoing arcs of vertex v, respectively. Given arc capacities $c \in \mathbb{R}^A_{\geq 0}$, a b-transshipment x is *feasible* if $x_a \leq c_a$ for all $a \in A$. As for the special single source case, finding a feasible b-transshipment reduces to a maximum flow computation. An *unsplittable b-transshipment* is a b-transshipment together with a path decomposition consisting of source–sink paths only, and with at most one path for each source–sink pair. In order to emphasize the fact that a particular transshipment x is not necessarily unsplittable, we sometimes refer to x as a *fractional* transshipment.

1.2 Single source unsplittable flows

The special case of unsplittable transshipments with a single source, that is, $S^+ = \{s\}$, coincides with the well-studied Single Source Unsplittable Flows (SSUF) problem. The rich literature on SSUF mostly refers to the demand values at sinks as $d_t := |b(t)|$ for $t \in S^-$. Arguably the most influential work on SSUF is due to Dinitz, Garg, and Goemans [4]. Their central result shows that any fractional flow x satisfying all demands can be efficiently transformed into a single source unsplittable flow y such that

$$y_a < x_a + d_{\max} \qquad \text{for all arcs } a \in A, \tag{1}$$

where $d_{\max} := \max_{t \in S^-} d_t$ denotes the maximum demand of any sink. In particular, in a capacitated single source network in which all arc capacities are at least $d_{\max}$ (so that any commodity might be routed along any arc) this result immediately yields a 2-approximation algorithm for finding an unsplittable flow that minimizes congestion. The result has also inspired a substantial body of subsequent research. In the early 2000s, Kolliopoulos and

Stein [11] and Skutella [16] observed that any fractional flow x satisfying all demands can be written as a convex combination of unsplittable flows y satisfying (1) if the demand values are multiples of one another, which has interesting consequences for finding unsplittable flows of bounded cost in digraphs with additional cost coefficients on the arcs. Only recently, Swamy, Traub, Vargas Koch, and Zenklusen [17] came up with interesting new results on unsplittable cost flows in this context. Morell and Skutella [13] show that any fractional flow x can be turned into an unsplittable flow y while maintaining the lower bounds $y_a \geq x_a - d_{\max}$ on all arcs $a \in A$. They also conjecture the existence of an unsplittable flow that satisfies both these lower bounds and the upper bounds (1). Recently, for the special case of acyclic and planar digraphs, this conjecture has been proved by Traub, Vargas Koch, and Zenklusen [18]. Moreover, Majthoub Almoghrabi, Skutella, and Warode [12] prove the conjecture on series-parallel digraphs, even for the more general setting of multiflows where demands must be routed between given source-sink pairs.

Beyond the congestion-minimization problem discussed above, Kleinberg [8] introduces further variants of unsplittable flow problems. The *minimum number of rounds* problem asks for a partition of the set of sinks into a minimum number of subsets (rounds), together with a feasible unsplittable flow for each subset. Building upon their main result, Dinitz, Garg, and Goemans [4] obtain a 5-approximation algorithm for this problem, improving upon earlier approximation results by Kleinberg [8] and Kolliopoulos and Stein [11]. In particular, if a feasible fractional flow satisfying all demands exists, then all commodities can be routed unsplittably in five rounds, and there are instances for which three rounds are necessary [4].

Another variant of unsplittable flows is the *maximum routable demand* problem, which seeks a feasible unsplittable flow for a subset of demands that maximizes the total routed demand. Again improving upon earlier results by Kleinberg [8] and Kolliopoulos and Stein [11], Dinitz, Garg, and Goemans [4] present a 0.226-approximation algorithm for this problem. They also exhibit an instance that admits a feasible fractional solution but for which at most a 0.385 fraction of the total demand can be routed unsplittably.

Beyond the single source case and the results discussed above, unsplittable flows have been studied extensively in more general settings and under a variety of objective functions. For a broader overview of this literature, we refer to the survey by Kolliopoulos [10] and to the work of Grandoni, Mömke, and Wiese [6].

1.3 Contribution and outline

We introduce the *Unsplittable Transshipment Problem*, a natural generalization of SSUF to networks with multiple sources and sinks and prescribed supplies and demands. Our main contribution is a nontrivial extension of the seminal result of Dinitz, Garg, and Goemans [4]. We show how to efficiently transform any feasible fractional transshipment into an unsplittable transshipment while increasing the flow on each arc by at most the maximum demand. It is interesting to note that a result like this cannot be achieved for the multiflow setting with given source-sink pairs; see, e.g., [10].

Compared to SSUF, unsplittable transshipments are more challenging since, in addition to selecting suitable paths for source–sink pairs, one must also determine appropriate flow values on these paths so that both supplies and demands are satisfied exactly. For example, even in networks with unit arc capacities and integral supplies and demands, deciding whether a feasible unsplittable transshipment exists is NP-hard; see [3]. Moreover, even when all supplies, demands, and arc capacities are integral, feasible unsplittable transshipments need not be integral; see [3].

Our main algorithm builds upon the DGG Algorithm and augments it with a carefully designed mechanism for splitting sinks and their demands into sub-sinks that are routed to distinct sources, thereby ensuring feasibility of the resulting unsplittable transshipment. Moreover, the solutions produced by our algorithm exhibit strong structural properties: the induced source–sink bipartite graph, where a source s is connected to a sink t if the unsplittable transshipment routes flow along an s–t path, is acyclic, yielding a tree-like structure. Moreover, the flow entering each sink t is confluent, meaning that any two paths ending at t that meet at some vertex coincide from that point onward until they reach t.

Beyond congestion, we also investigate the *minimum number of rounds* and the *maximum routable demand* objectives. Also these variants turn out to be more intricate for unsplittable transshipments than in the single source setting. In particular, we derive constant bounds on the number of rounds only under additional structural assumptions, and we obtain such results only in special cases.

The paper is organized as follows. In Section 2, we review the DGG Algorithm for single source unsplittable flows and highlight the key ideas underlying its analysis. Section 3 presents our Modified DGG Algorithm for unsplittable transshipments and proves the main structural and approximation results. In Section 4, we investigate routing in rounds and the maximum routable demand problem under additional assumptions. We conclude in Section 5 by highlighting several open questions for future research.

Due to length constraints, some details are omitted here; see [3] for the full version of the paper.

2 A short review of the Dinitz-Garg-Goemans Algorithm for SSUFs

In the following, we summarize the algorithm of Dinitz, Garg, and Goemans [4] (the *DGG Algorithm*) and highlight key ideas underlying its analysis. Subsequently, in Section 3, we present an extended version of the DGG Algorithm for the Unsplittable Transshipment Problem.

2.1 Description of the DGG Algorithm

We consider an instance of the SSUF problem on an acyclic digraph $D = (V, A)$ with a single source s, a set of sinks S^- with associated demands d_t for $t \in S^-$, and a flow x that satisfies these demands. The DGG Algorithm iteratively modifies the initial flow x, and whenever the flow on an arc is reduced to zero during this process, the arc is removed from the network.

In a preliminary phase, starting from the given flow $y := x$, the DGG Algorithm iteratively moves sinks backward toward the source s along flow-carrying arcs whose flow meets or exceeds the sink's demand, decreasing the flow y on the traversed arcs accordingly. The preliminary phase terminates once every sink has either reached the source s or is located at a vertex with at least two incoming arcs and is regular: A sink is called *regular* if the flow on each arc entering its current vertex is strictly smaller than the demand of the sink; otherwise, the sink is *irregular*.

Once the preliminary phase ends, the algorithm iteratively augments the current flow y along a carefully chosen alternating cycle and, in each iteration, tries to move the sinks toward the source s according to certain rules.

A key notion for constructing alternating cycles is that of *singular arcs*. At the beginning of an iteration, an arc (u, v) is labeled *singular* if v and all vertices reachable from v have out-degree at most one. An alternating cycle is constructed as follows. Starting from an arbitrary vertex, follow the outgoing arcs until a *junction vertex* w, that is, a vertex with no

outgoing arcs, is reached (the *forward path*). Then, beginning with an incoming arc at w that is distinct from the arc used to reach w, construct a *backward path* traversing the incoming singular arcs as far as possible. Upon reaching a vertex with at least two outgoing arcs, resume the construction of a forward path. This alternation of forward and backward paths is repeated until a vertex is revisited, thereby forming an *alternating cycle* C consisting of alternating forward and backward paths.

The flow y is then augmented along the alternating cycle C as follows. The flow on each forward arc of C is decreased, and the flow on each backward arc of C is increased, by the same positive amount which is the minimum of two values: The first value is the minimum flow along a forward arc of C. The second is the minimum of $d_t - y_a$ taken over all backward arcs $a = (v, w)$ of C and over all sinks t at w with $d_t > y_a$. If the minimum is achieved by the flow y_a on a forward arc a, then the augmentation reduces the flow on this arc to zero and the arc is thus removed. Finally, if possible, sinks are moved backward towards source s, preferably along singular arcs carrying exactly their demand, and otherwise along non-singular arcs carrying at least their demand.

The algorithm terminates once all sinks have reached the source s, and the resulting unsplittable flow is defined by the collection of paths traced by the sinks during their backward movement.

2.2 Analysis of the DGG Algorithm

The correctness and analysis of the DGG Algorithm relies on the following invariants, maintained throughout its execution:

(i) Flow y satisfies all current demands of sinks not yet moved to s.

Proof. In the augmentation step, flow is increased along backward paths and decreased along forward paths by the same amount, preserving the excess (inflow minus outflow) at each vertex, thus maintaining a flow that meets all current demands. ◀

(ii) At the beginning of each iteration, every vertex contains at most one irregular sink. If a vertex does contain an irregular sink, then its out-degree is zero and it also contains a regular sink. In particular, every vertex that contains sinks has at least two incoming arcs.

Proof. We refer to the proof of Lemma 3.2 in [4]. ◀

Notice that the last part of this invariant is crucial for the construction of alternating cycles since it guarantees that, for each junction vertex, there exists an incoming arc different from the forward arc used to reach it such that a backward path can be constructed.

(iii) A singular arc is removed at the end of an iteration in which any sink moves along it.

Proof. A sink moves along a singular arc in an iteration only when the flow on that arc exactly matches the demand of that sink before moving. ◀

Until an arc a becomes singular, its flow never increases and the total demand of sinks moved along it is bounded by its initial flow value x_a. Once a is singular, at most one sink with demand at most $d_{\max}$ can move along it before a disappears. Thus, in the final unsplittable flow, the flow on any arc is less than $x_a + d_{\max}$. We refer the interested reader to [4] for further details.

3 Modified DGG Algorithm for unsplittable transshipments

In this section, we modify the DGG Algorithm and its analysis in order to compute unsplittable transshipments with bounded congestion. We are given an acyclic digraph $D = (V, A)$ with a set of sources S^+ and a set of sinks S^-, together with supplies and demands specified by a balance function $b\colon V \to \mathbb{R}$ and a fractional b-transshipment x. We first transform this unsplittable transshipment instance into an SSUF instance by introducing a super-source s^* and dummy arcs (s^*, s) to each source $s \in S^+$, each carrying flow $x_{(s^*,s)} := b(s)$, and by setting $d_t := |b(t)|$ for all sinks $t \in S^-$.

However, applying the DGG Algorithm to the resulting SSUF instance does not necessarily yield a valid b-transshipment. During the execution of the algorithm, the flow on dummy arcs (s^*, s) that lie on alternating cycles may be modified, which can result in an unsplittable flow that no longer satisfies the fixed supply constraints at the sources $s \in S^+$.

On the other hand, the DGG Algorithm routes the entire demand of each sink along a single path and does not exploit the possibility of sending flow into a sink along multiple paths originating from different sources in S^+. The central idea of our Modified DGG Algorithm is to leverage this flexibility in order to keep the flow on dummy arcs (s^*, s), and hence the supplies at the sources, fixed.

3.1 Description of the Modified DGG Algorithm

As in the original DGG Algorithm, the Modified DGG Algorithm begins with the flow $y := x$ and, in a preliminary phase, iteratively moves sinks backward toward the super-source s^* along flow-carrying arcs incident to sinks whose flow meets or exceeds their demands. After the preliminary phase, the algorithm iteratively attempts to construct alternating cycles that do not contain the super-source s^*. Augmenting flow along a cycle that includes s^* would modify the flow on arcs (s^*, s) and thus alter the fixed supply values of such sources $s \in S^+$. We therefore proceed as follows.

Nice alternating cycles and singular digraphs

As in the original DGG Algorithm, we attempt to construct an alternating cycle by alternating between *forward* (arbitrary) and *backward* (singular) arcs. The last vertex reached on a forward path during an iteration is a junction vertex w with no outgoing arcs. All incoming arcs of w are singular, and, by flow conservation, w necessarily contains one or more sinks. Furthermore, a vertex is called a *funnel vertex* if it has at most one outgoing arc; otherwise, it is called a *non-funnel vertex*. Note that the head of any singular arc is a funnel vertex, and that backward paths in alternating cycles always terminate upon reaching a non-funnel vertex. We call an alternating cycle *nice* if it does not contain the super-source s^*. Nice alternating cycles allow us to augment the flow without changing the supplies at the sources in S^+ and thus to proceed exactly as in the original DGG Algorithm.

There are, however, situations in which no nice alternating cycle can be found. This occurs precisely when, starting from a junction vertex w, all possible backward paths lead to the super-source s^*. Equivalently, s^* is the only non-funnel vertex reachable from w via backward paths. This can be verified by a simple backward depth-first search (DFS) starting at w while ignoring the arc a' used to reach w. If backward DFS finds a non-funnel vertex $v \neq s^*$, we can continue our search for a nice alternating cycle along a forward path from v. Otherwise, if all backward paths lead to s^*, we refer to the sub-digraph explored by this backward DFS as the *singular digraph rooted at* w.

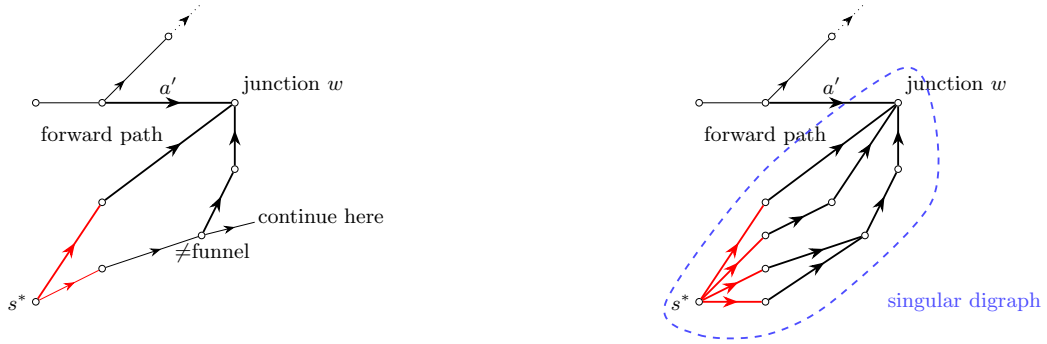

Figure 1 Backward path discovery in the Modified DGG Algorithm starting at junction vertex w with singular arcs depicted in thick. *Left*: A non-funnel vertex $\neq s^*$ is found along some backward path, allowing to continue the construction of a nice alternating cycle. *Right*: The only non-funnel vertex found on backward paths is s^*, yielding a singular digraph rooted at junction vertex w.

If a singular digraph rooted at the junction vertex w is found, we select a (sub-)sink t located at w with $d_t > y_{a'}$. The existence of such a (sub-)sink is guaranteed by Invariant (ii), whose validity is established below. This (sub-)sink t is uniquely chosen whenever a singular digraph is encountered. It will later be split into two sub-sinks t_1 and t_2 with demands $d_{t_1} := y_{a'}$ and $d_{t_2} := d_t - y_{a'}$, respectively. After all other (sub-)sinks in the singular digraph have been moved to the super-source as described below, the sub-sink t_2 is created and routed to the super-source as well. Subsequently, the sub-sink t_1 is moved backward along the arc a', reducing the flow on a' to zero and thereby removing a' from the network.

▶ **Lemma 1.** *With the exception of arc a', used to reach w on the forward path, all incoming arcs of vertices in the singular digraph rooted at w belong to the singular digraph. Moreover, every vertex $v \neq s^*$ in the singular digraph has at most one outgoing arc, and this outgoing arc belongs to the singular digraph.*

Proof. The first statement follows from the fact that the singular digraph is constructed via a backward depth-first search. For the second statement, observe that every vertex $v \neq s^*$ in the singular digraph is a funnel vertex and hence has at most one outgoing arc. Moreover, this outgoing arc lies on the unique path from v to the junction vertex w, which is entirely contained in the singular digraph. ◀

In other words, Lemma 1 states that the sub-digraph of the current network induced by all vertices of the singular digraph except s^* is an in-tree, i.e., a directed tree rooted at w with all arcs oriented toward w. As a consequence of Lemma 1, the flow y on the singular digraph exactly satisfies the demands of all (sub-)sinks located at its vertices, with the sole exception of the uniquely chosen (sub-)sink t, for which only the amount $d_{t_2} := d_t - y_{a'}$ is satisfied within the singular digraph. Since the singular digraph contains at most one path from each source in S^+ to each of its (sub-)sinks, any path decomposition of its flow yields an unsplittable transshipment that satisfies all these demands.

Equivalently, the algorithm decomposes the flow y on the singular digraph into paths, splits each (sub-)sink into as many sub-sinks as there are paths serving it, and then moves each such sub-sink backward along its corresponding path to the super-source s^*. Finally, all arcs of the singular digraph are removed.

At the end of an iteration, further sinks may move backward towards the super-source s^* exactly as in the original DGG Algorithm. A summary is given in Algorithm 1.

Algorithm 1 Modified DGG Algorithm.

Input: Unsplittable transshipment instance (D, S^+, S^-, b, x)
Output: Unsplittable transshipment defined by path set $\{\mathcal{P}_t\}_{t \in S^-}$

construct corresponding SSUF instance (D^*, s^*, S^-, d, x); set $y := x$
while there exists $a = (v, t) \in A$ with $y_a \geq d_t$ **do** ▷ **Preliminary phase**
    decrease y_a by d_t; move sink t to v; delete a if $y_a = 0$
while some (sub-)sink has not reached s^* **do** ▷ **Iterations**
    label singular arcs and funnel vertices
    if a *nice alternating cycle* is found **then**
        augment flow and move (sub-)sinks exactly as in the original DGG Algorithm
    else
        move the (sub-)sinks in the singular digraph[1] backward to the super-source
        move further (sub-)sinks according to DGG moving rules

[1] *(sub-)sinks in the singular digraph* refers to all (sub-)sinks other than t located at vertices of the singular digraph, and the sub-sink t_2 of t.

3.2 Analysis of the Modified DGG Algorithm

The correctness of the Modified DGG Algorithm is based on the preservation of the same Invariants (i), (ii), and (iii) as the original DGG Algorithm.

Proof of Invariant (i). If a nice alternating cycle is found in an iteration, the flow is augmented using the same procedure as in the original DGG Algorithm. This augmentation preserves the property that the flow satisfies the demands of all current (sub-)sinks that have not yet reached s^*. In the remaining iterations, the flow on the singular digraph exactly satisfies the demands of the (sub-)sinks located at its vertices. Consequently, removing the arcs of the singular digraph together with its (sub-)sinks does not violate Invariant (i). ◀

Proof of Invariant (ii). The proof proceeds by induction on the number of iterations. All sinks are regular at the end of the preliminary phase such that the invariant holds at the beginning of the first iteration. If a nice alternating cycle is found in an iteration, Invariant (ii) is preserved exactly as shown in the proof of [4, Lemma 3.2]. Otherwise, the algorithm identifies a singular digraph rooted at the junction vertex w and splits a sink t located at w into two sub-sinks t_1 and t_2 such that the demand d_{t_1} of t_1 equals the flow on the arc a' along which the forward path reaches w. Except for t_1, all (sub-)sinks at the vertices of the singular digraph are deleted together with its arcs, while t_1 is moved to the tail of a' (and possibly further), and the arc a' is removed.

Consequently, t_1 is the only sink that can potentially violate Invariant (ii). Suppose that t_1 is eventually moved to a vertex v. If t_1 is regular at v, the invariant is clearly preserved. Otherwise, t_1 is irregular at v, which implies the existence of an incoming arc a with flow value $y_a \geq d_{t_1}$. Since t_1 does not move further backward along a, it must hold that $y_a > d_{t_1}$ and that the arc a was already labeled singular at the beginning of the iteration.

In particular, the outgoing arc of v along which t_1 was moved backward to v must have been singular and, moreover, the only outgoing arc of v. This arc is therefore deleted after the move, implying that the new out-degree of v is zero. Since there was no irregular sink at v at the beginning of the iteration (as its out-degree was one), t_1 is the only irregular sink at v. Finally, as $y_a > d_{t_1}$, there must exist at least one additional sink at v, which completes the proof. ◀

By Invariant (ii), whenever a forward path reaches a junction vertex w along an arc a', there exists at least one additional incoming (singular) arc at w along which the backward depth-first search can continue and either reach a non-funnel vertex $v \neq s^*$ or identify a singular digraph. Consequently, in every iteration, the algorithm finds either a nice alternating cycle or a singular digraph, as described in the algorithm.

Proof of Invariant (iii). A (sub-)sink can be moved along a singular arc in two distinct ways. If it is moved according to the first DGG moving rule, the flow on the arc is reduced to zero and the arc is removed immediately. Otherwise, the arc belongs to a singular digraph, along which (sub-)sinks are routed without augmenting the flow; all arcs in this singular digraph are removed at the end of the iteration. In both cases, any singular arc used to move a (sub-)sink toward the source is removed from the digraph within the same iteration, thereby preserving the invariant. ◀

Next, we prove that the b-transshipment computed by the Modified DGG Algorithm is unsplittable. We actually prove the even stronger property that the demand of every sink is routed *confluently*. This means that the paths along which the sub-demands of a sink are routed backward toward the sources in S^+ form an in-tree, that is, a directed tree with all arcs oriented toward the sink.

▶ **Lemma 2.** *The Modified DGG Algorithm computes a b-transshipment in which the demand of every sink is routed confluently.*

Proof. As already mentioned, we argue that after deleting the super-source s^*, the paths along which a sink t (or its sub-sinks) are moved backward toward s^* form an in-tree rooted at t, whose leaves are sources in S^+.

During the preliminary phase, as well as in later iterations in which a (sub-)sink t is not located at a vertex of a singular digraph, it is moved backward along a single path toward the super-source s^*. Therefore, it remains to consider iterations in which t is located at a vertex of a singular digraph rooted at a junction vertex w. We distinguish two cases.

First case. Assume that t is the uniquely chosen (sub-)sink located at w that is split by the algorithm into two sub-sinks t_1 and t_2. In this case, t_2 may be further split into sub-sinks that are routed backward toward s^* within the singular digraph. By Lemma 1, the corresponding backward paths, after deleting s^*, form a tree. Since all arcs of the singular digraph are removed at the end of the iteration, all its vertices except for s^* have no outgoing arcs afterwards. Consequently, the paths along which the sub-sink t_1 (or its sub-sinks) is later moved backward toward s^* do not intersect any of these vertices.

Second case. Otherwise, by the same argument as for t_2 in the first case, the backward paths taken by t (or its sub-sinks), after deleting s^*, form a tree.

In summary, the backward paths along which the demand of a sink t (or its sub-sinks) is moved form, after deleting the super-source s^*, an in-tree rooted at t. ◀

We are now ready to prove the main result of this section, generalizing the classical result of Dinitz, Garg, and Goemans [4] to the Unsplittable Transshipment Problem.

▶ **Theorem 3.** *Given a b-transshipment x, the Modified DGG Algorithm efficiently computes an unsplittable transshipment such that the flow on each arc a is strictly less than $x_a + d_{\max}$.*

Proof. Lemma 2 implies that the algorithm computes an unsplittable transshipment. Thus, it remains to prove the bound on the flow values. As in the original DGG Algorithm, the flow on a non-singular arc is never increased. Hence, the total demand routed along an arc

up to the point at which it becomes singular is less than its initial flow. Subsequently, the arc a may appear repeatedly as a backward arc in nice alternating cycles, during which its flow may be increased multiple times. Observe that a remains singular until it is removed from the network, since no arcs are added during the algorithm and vertex out-degrees never increase. In the following, we argue that the total increase of flow on a is bounded by $d_{\max}$.

After the last increase of flow on a, let y denote the resulting flow and let t be the first regular (sub-)sink encountered at a vertex v along the unique directed path starting at the head of a. If v is the head of a, set $a_v := a$; otherwise, let a_v be the incoming arc of v on this path. By Invariant (ii), v is the first vertex on this path that contains a sink. Hence, by flow conservation, $y_a \leq y_{a_v} \leq d_t$. In particular, the total increase of flow on arc a is strictly less than $d_t \leq d_{\max}$. ◀

The *congestion* of a transshipment is defined as the maximum, over all arcs, of the ratio between the flow routed on the arc and its capacity. We assume the standard *balance condition*, that is, the capacity of every arc is at least the maximum demand $d_{\max}$. Under this assumption, Theorem 3 immediately implies the following corollary.

▶ **Corollary 4.** *Under the balance condition, the Modified DGG Algorithm yields a* 2*-approximation algorithm for minimizing congestion in unsplittable transshipment instances.*

Instead of working with a super-source s^* as we did above, one could just as well introduce a super-sink t^* and then reverse all arc directions.

▶ Remark 5. Reversing all arcs yields a maximum capacity violation of at most $\max_{s \in S^+} b(s)$. Thus, the Modified DGG Algorithm computes, in polynomial time, an unsplittable transshipment whose arc capacity violation is bounded by the minimum of the maximum supply $\max_{s \in S^+} b(s)$ and the maximum demand $\max_{t \in S^-} |b(t)|$.

3.3 Further observations

Arc-wise lower bounds for unsplittable transshipments

Rohwedder and Svensson (personal communication, 2021) observed that the DGG Algorithm can be adapted to guarantee lower bounds on arc flows for unsplittable flows. By applying the same idea, one obtains an analogous result for the Unsplittable Transshipment Problem.

▶ **Theorem 6.** *Given a b-transshipment x, the adapted Modified DGG Algorithm efficiently computes an unsplittable transshipment such that the flow on each arc a is greater than $x_a - d_{\max}$.*

As with the original DGG Algorithm, all steps remain the same except the augmentation step when a nice alternating cycle is found. In that case, we augment in the opposite direction: we increase the flow on the forward paths and decrease it on the backward paths.

Tightness of the Modified DGG Algorithm

For the SSUF problem, Dinitz, Garg, and Goemans show that a capacity violation of $d_{\max}$ is unavoidable in the worst case; see [4, Figure 5]. Since SSUF is a special case of the unsplittable transshipment problem, this bound is also tight for unsplittable transshipments when each sink is reachable from a single source. The tightness of the bound is less immediate in highly connected instances, where each sink is reachable from many sources, and its demand could, in principle, be split so as to reduce congestion. We note that this intuition is misleading: the bound of $d_{\max}$ remains tight even when every sink is connected by directed paths to an arbitrarily large number of sources; see [3].

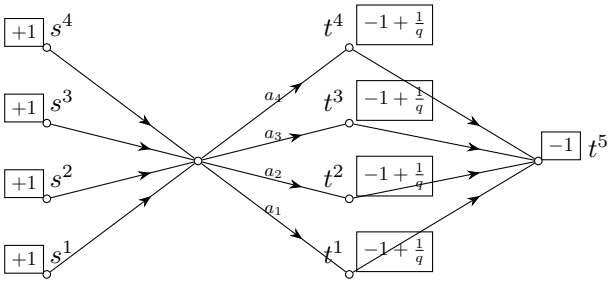

Figure 2 Family of instances (depicted for $q = 4$) with unit arc capacities and $d_{\max} = 1$, admitting a feasible non-confluent unsplittable transshipment, while any confluent unsplittable transshipment has congestion $1 - 1/q$, tending to $d_{\max}$ as $q \to \infty$.

Bounding the total number of paths

By definition, an arbitrary unsplittable transshipment may use up to $|S^+| \cdot |S^-|$ distinct paths. However, the following theorem implies that any unsplittable transshipment produced by the Modified DGG Algorithm uses at most $|S^+| + |S^-| - 1$ paths; see [3] for a proof.

▶ **Theorem 7.** *Given an unsplittable transshipment, define a bipartite graph with vertex set $S^+ \cup S^-$ by adding an edge between a source $s \in S^+$ and a sink $t \in S^-$ if and only if the unsplittable transshipment routes flow along an s–t path. Then, the bipartite graph associated with any unsplittable transshipment computed by the Modified DGG Algorithm is acyclic and thus contains at most $|S^+| + |S^-| - 1$ edges.*

Cost of Confluence

Lemma 2 implies that the Modified DGG Algorithm returns a sink-wise confluent unsplittable transshipment. This leads to the question of whether such confluence has an inherent cost: are there instances with a feasible (non-confluent) unsplittable transshipment for which every confluent unsplittable transshipment has congestion close to $d_{\max}$, i.e., the largest possible gap?

The answer is yes, as demonstrated by a family of instances parameterized by q, illustrated for $q = 4$ in Figure 2. Each sink t^i with $i \in \{1, \dots, q\}$ must route its demand through arc a_i. Thus, any unsplittable routing reserves $1 - 1/q$ units of capacity on a_i for sink t^i. If we additionally require confluence for sink t^{q+1} (denoted t^5 in Figure 2), then some arc a_i must carry a total flow of $1 + (1 - 1/q)$. Hence, in any confluent unsplittable transshipment, the capacity violation on that arc is at least $1 - 1/q$, which tends to $d_{\max} = 1$ as $q \to \infty$.

4 Minimum number of rounds and maximum routable demand

Let the digraph $D = (V, A)$ have arc capacities $c_a \geq 0$, $a \in A$. We impose the *balance condition* that $d_{\max} := \max_{t \in S^-} |b(t)| \leq \min_{a \in A} c_a =: c_{\min}$ and assume that a feasible fractional transshipment exists for the given unsplittable transshipment instance. Let $d_{\text{tot}} := \sum_{t \in S^-} |b(t)|$ denote the total demand of all sinks and let $b^+_{\text{avg}} := \sum_{s \in S^+} b(s)/|S^+| = d_{\text{tot}}/|S^+|$ denote the average supply of a source. Note that we make no assumptions about the supply values; in particular, sources and sinks need not be symmetric with respect to supply and demand values.

In this section, we generalize Kleinberg's notions of *minimum number of rounds* and *maximum routable demand* for SSUF [8]. In the context of unsplittable transshipments, routing in rounds admits several natural generalizations. In one variant, an instance is

routable in k rounds if and only if there exist k feasible unsplittable transshipments whose union (of their path decompositions) is an unsplittable transshipment that satisfies all source supplies and sink demands. We refer to this as *routing with permissible splitting of demands between rounds*.

However, the definition of the *minimum number of rounds* problem that we adopt in this paper, asks for a partition of the set of sinks into a minimum number of subsets (rounds), together with a feasible unsplittable transshipment for each subset, such that for every source the total flow sent to sinks over all rounds equals its prescribed supply. This definition does not allow sinks to split their demands between rounds, so the demand of each sink must be routed in a single round, unlike in routing with permissible splitting of demands between rounds.

Moreover, the *maximum routable demand* problem seeks a feasible unsplittable transshipment that satisfies the demands of a subset of sinks, maximizes the total demand of this subset, and does not exceed the available supplies at the sources.

4.1 Minimizing the number of rounds

SSUF versus unsplittable transshipments

Under the balance condition, it is shown in [4] that all demands in an SSUF instance can be routed in polynomial time using at most five rounds, by partitioning demands into *small* ones [4, Lemma 4.1] and *large* ones [4, Lemma 4.2], and then routing each class separately on suitably capacity-scaled copies of the network.

However, for unsplittable transshipments, routing in rounds poses additional challenges because rounds are *not independent*: the supply of a source consumed in earlier rounds reduces the supply available in later rounds. This cross-round "memory" has no analog in the SSUF setting. Moreover, unlike in SSUF, where all indivisible demands are known in advance, the Modified DGG Algorithm may generate arbitrarily small sub-demands during its execution, so that no meaningful global partition of demands into "small" and "large" classes exists. Furthermore, scaling down capacities does not guarantee that at most one demand uses a given arc. As a consequence, feasibility arguments based on capacity scaling, such as those used in [4, Lemma 4.2], no longer apply. Routing in rounds for unsplittable transshipments therefore requires more refined techniques.

For the case in which demand values are sufficiently smaller than $c_{\min}$, we prove bounds on the number of rounds for several such regimes in [3]. In the following, we present a unified strategy that routes all instances with $d_{\max} < c_{\min}$ in a number of rounds that depends only on the gap between $d_{\max}$ and $c_{\min}$.

Routing demands in rounds when $d_{\max} < c_{\min}$

We may assume without loss of generality that $d_{\max} \leq (1 - 1/n) \cdot c_{\min}$ for some $n \in \mathbb{N}$. In this case, we show the following bound on the number of needed rounds.

▶ **Theorem 8.** *Given a feasible fractional transshipment instance with $d_{\max} < (1 - 1/n) \cdot c_{\min}$, the instance is unsplittably routable in a constant number N_{tot} of rounds, where N_{tot} is a large constant dependent only on n.*

Proof. Following the construction of [4, Figure 6], we construct an auxiliary digraph $D' = (V', A')$ consisting of $n + 1$ identical copies of D. For each arc $a \in A$ and each copy $\alpha \in \{1, \ldots, n + 1\}$, we denote its copy by a_α and set its capacity to $c_{a_\alpha} := c_a/n+1$.

For each sink $t \in S^-$, we add a super-sink T and connect every copy of t to T with an arc of capacity $d_t/(n+1)$. Symmetrically, for each source $s \in S^+$ we introduce a head-source S and connect every copy of s to S with an arc of capacity $b(s)/(n+1)$. We now apply the Modified DGG Algorithm by connecting every head-source S to a super-source S^* by an arc of capacity $b(s)$. The original fractional single source flow corresponds to routing a fraction $1/(n+1)$ of the transshipment through each copy of D. Since

$$c_{a_\alpha} + d_{\max} \leq \frac{c_a}{n+1} + \left(1 - \frac{1}{n}\right) c_{\min} \leq c_a,$$

each copy is individually feasible with respect to the original capacities. However, we cannot directly interpret each copy as a routing round, because the Modified DGG Algorithm may split the demand of a sink across multiple copies of D which is incompatible with the notion of routing in rounds.

We therefore distinguish two types of sinks. A sink is *critically split* if its demand is routed partly along forward path(s) contained in a single copy of D and partly along a *singular digraph* that may span multiple copies. Otherwise, it is *non-critically split* and its entire demand is routed within one copy, which can be interpreted directly as a round where such a sink is routed.

We observe that in any fixed copy of D, each arc a' can be used in the singular digraph of at most one critically split sink. Once used, a' is deleted and cannot be included in further singular digraphs.

Each critically split sink t is associated with

- **i)** a *label* $\mathcal{L} \in \{1, \ldots, n+1\}$ identifying the copy of D containing its forward path(s)
- **ii)** a *demand-share vector* $\theta^t = (\theta^t_1, \ldots, \theta^t_{n+1})$, where $\theta^t_\alpha := d^\alpha_t/d_t$ denotes the fraction of demand routed through copy α of D.

The key idea is to group sinks with *similar demand distributions*, that is, with similar vectors θ^t, so that within each group we can treat all sinks as if they use approximately the same fraction of capacity in every copy. This allows us to upper bound the total flow on each arc by a simple worst-case estimate per copy.

Observe that the demand-share vectors θ^t lie in the n-dimensional simplex. To group sinks of the same label with similar demand distributions, we discretize this simplex using a uniform grid of granularity $\varepsilon \leq 1/M$, where $M := (n^2-1)(n+1)$. For each coordinate, the range $[0, 1)$ is sliced into intervals of size ε. These intervals are of the form $[l_\alpha, u_\alpha)$ where $l_\alpha = k_\alpha \cdot \varepsilon$, and $u_\alpha = l_\alpha + \varepsilon = (k_\alpha + 1) \cdot \varepsilon)$ for $\alpha \in \{1, \ldots, n+1\}$. A *group* is now defined by a label and a choice of one interval per coordinate of the form

$$(\mathcal{L},\ ([l_1, u_1), \ldots, [l_{n+1}, u_{n+1}))^T.$$

Such a group can contain a demand-share vector if and only if

$$\sum_{\alpha=1}^{n+1} l_\alpha \leq 1 \quad \text{and} \quad 1 \leq \sum_{\alpha=1}^{n+1} u_\alpha = \sum_{\alpha=1}^{n+1} (l_\alpha + \varepsilon) = \sum_{\alpha=1}^{n+1} l_\alpha + (n+1) \cdot \varepsilon,$$

thus, $1 - (n+1) \cdot \varepsilon \leq \sum_{\alpha=1}^{n+1} l_\alpha \leq 1$, or equivalently, by multiplying with M and using $l_\alpha = k_\alpha \varepsilon$,

$$M - (n+1) \leq \sum_{\alpha=1}^{n+1} k_\alpha \leq M.$$

Hence, for a fixed label $\mathcal{L}$, admissible groups correspond to non-negative integer $(n+1)$–tuples (k_α) whose total sum lies in $\{M-(n+1),\dots,M\}$. By the stars-and-bars argument, the total number Γ of such groups for a fixed label is

$$\Gamma = \sum_{j=0}^{n+1} \binom{M-j+n}{n} = \sum_{j=0}^{n} \binom{(n^2-1)\cdot(n+1)-j+n}{n} < \infty.$$

We now show that superimposing all routing paths of critically split sinks, including the corresponding singular digraphs, in a fixed group yields a feasible routing round.

Fix an arc $a \in A$. We analyze the total flow on a by summing contributions over its copies a_α. The forward paths lie entirely in the copy corresponding to label $\mathcal{L}$ and contribute at most $c_a/n+1 + d_{\max} \cdot u_{\mathcal{L}}$. For each copy $\alpha \neq \mathcal{L}$, the singular digraph contributes at most $d_{\max} \cdot u_\alpha$. Indeed, as we observed earlier, each arc a_α is used by at most one critically split sink in its singular digraph. Thus, the total flow on a is strictly less than

$$\frac{c_a}{(n+1)} + d_{\max} \cdot \sum_{\alpha=1}^{n+1} u_\alpha$$

with

$$\sum_\alpha u_\alpha = \sum_\alpha l_\alpha + (n+1)\cdot\varepsilon \;\le\; 1 + (n+1)\cdot\varepsilon \quad \text{and} \quad \varepsilon \;\le\; \frac{1}{M} \;=\; \frac{1}{(n^2-1)(n+1)}.$$

We obtain

$$\begin{aligned}\frac{c_a}{(n+1)} + d_{\max} \cdot \sum_{\alpha=1}^{n+1} u_\alpha \;&\le\; \frac{c_a}{n+1} + \left(1-\frac{1}{n}\right)\cdot c_{\min}\cdot(1+(n+1)\cdot\varepsilon) \\ &\le\; c_a \cdot \left(\frac{1}{n+1} + \frac{n-1}{n} + \frac{(n^2-1)\cdot\varepsilon}{n}\right) \;\le\; c_a.\end{aligned}$$

This implies that all capacity constraints are satisfied.

Critically split sinks require at most $(n+1)\cdot\Gamma$ rounds. Adding the $n+1$ rounds needed for non-critically split sinks, the total number of routing rounds is $N_{\text{tot}} = (n+1)\cdot(\Gamma+1)$. ◀

▶ **Observation 9.** *Assuming $d_{\max} < (1-1/n)\cdot c_{\min}$, for routing with permissible splitting of demands between rounds, n rounds are sufficient.*

Indeed, we can use a similar construction of auxiliary digraph D' with n identical copies of D as in the proof of Theorem 8 and apply Modified DGG Algorithm. We therefore obtain a flow value of

$$\frac{c_a}{n} + d_{\max} \;\le\; \frac{c_a}{n} + \left(1-\frac{1}{n}\right)\cdot c_{\min} \;\le\; c_a$$

for each arc $a \in A$.

4.2 Special cases for the maximum routable demand

When $d_{\max} < c_{\min}$, the maximum routable demand admits an approximation factor depending on the gap between $d_{\max}$ and $c_{\min}$. The boundary case $d_{\max} = c_{\min}$ remains open, both for routing in rounds and for maximizing routable demand. We now identify settings where additional constraints on source supplies yield constant-factor approximations for the maximum routable demand.

Assuming $d_{\max} \leq \min_{s \in S^+} b(s)$

In this case, we can generalize [4, Corollary 5.4] and prove the following.

▶ **Lemma 10.** *Assume the balance condition $d_{\max} \leq \min_{s \in S^+} b(s)$ and the existence of a feasible fractional transhipment. Then, the maximum routable demand exceeds 0.226 of the total demand.*

Proof. Since each arc (s^*, s) has capacity $b(s) \geq d_{\max}$, the derived SSUF instance with super-source s^* satisfies the balance condition. By [4, Corollary 5.4], at least $0.226 \cdot d_{\text{tot}}$ units of demand can be routed in this SSUF instance without violating arc capacities, and hence also in the original unsplittable transshipment instance without exceeding source supplies. Therefore, there exists a feasible unsplittable routing that serves at least $0.226 \cdot d_{\text{tot}}$ total demand, without demand being split across sources. ◀

Assuming $d_{\max} \leq 1/2 \cdot b^+_{\text{avg}}$

Here, we can show the following.

▶ **Lemma 11.** *Assume the balance condition $d_{\max} \leq 1/2 \cdot b^+_{\text{avg}}$ and the existence of a feasible fractional transhipment. Then, the maximum routable demand exceeds 0.07143 of the total demand.*

Proof. When $d_{\max} \leq c_{\min}/3$, that is, for *small* demands, there exists a feasible six-round unsplittable routing; see [3]. Hence, in this case, at least $1/6 \cdot d_{\text{tot}}$ can be routed unsplittably in a single round.

We now restrict our attention to the complementary case of *large* demands in $[d_{\max}/3, d_{\max}]$, which in particular contains $[c_{\min}/3, d_{\max}]$.

Assume in the following that the demands lie in the range $[d_{\max}/3, d_{\max}]$ and that we are additionally given $d_{\max} \leq 1/2 \cdot b^+_{\text{avg}} = d_{\text{tot}}/2|S^+|$.

Applying the original DGG algorithm to the derived SSUF instance with a super-source routes all demands unsplittably in four rounds [4, Lemma 4.2]. When applied to unsplittable transshipment instances, individual sources may be overutilized, but by at most one sink [4, Theorem 3.7], so the overutilization at any source is less than $d_{\max}$. Thus, the total overutilization is less than $|S^+| \cdot d_{\max}$, which by assumption is at most $1/2 \cdot d_{\text{tot}}$. Therefore, at least $1/2 \cdot d_{\text{tot}}$ units of demand are routed without violating any source supply across the four rounds, and hence at least one round routes a feasible unsplittable flow of value at least $1/4 \cdot 1/2 \cdot d_{\text{tot}} = 1/8 \cdot d_{\text{tot}}$.

Thus, combining the cases of *small* and *large* demands, the maximum routable demand is lower bounded by $\max\{d/6, 1-d/8\} \cdot d_{\text{tot}}$ over all $d \in [0, 1]$. This expression attains its minimum at $d = 3/7$ and is therefore always at least $0.07143 \cdot d_{\text{tot}}$. ◀

5 Conclusion and open problems

We have introduced unsplittable transshipments as a natural extension of single source unsplittable flows. Our results and techniques indicate that unsplittable transshipment problems pose new and interesting algorithmic challenges and are inherently more difficult than their single source counterparts. We conclude by highlighting several open questions and problems that may stimulate future research.

The Modified DGG Algorithm can be adapted to guarantee lower bounds on arc flows in unsplittable transshipments; see Theorem 6. Whether arc-wise upper and lower bounds can be achieved simultaneously remains an intriguing open question, even in the special SSUF case. Extending a conjecture for SSUF by Morell and Skutella [13], we conjecture that such transshipments exist and can be computed efficiently.

▶ **Conjecture 12.** *Given a b-transshipment x, one can efficiently compute an unsplittable b-transshipment y such that*

$$x_a - d_{\max} \;\le\; y_a \;\le\; x_a + d_{\max} \quad \text{for all } a \in A.$$

Motivated by a famous conjecture of Goemans, many authors have studied cost-based variants of SSUF; see, e.g., [11, 16, 18, 17]. Generalizing some of these results to unsplittable transshipments poses an interesting challenge.

In Section 4, we study the problem of minimizing the number of rounds while imposing the balance condition $d_{\max} \le c_{\min}$. Our techniques exploit the additional restriction to $d_{\max} < c_{\min}$, allowing one to sufficiently separate the demands from $c_{\min}$. It remains open whether explicit bounds can be obtained even if $d_{\max} = c_{\min}$, both for routing in rounds and for maximizing the routable demand. Other variants, such as constant-factor approximations for minimizing the number of rounds, may also provide further insight.

Another related model is the *k-splittable flow* problem, introduced by Baier, Köhler, and Skutella [2], in which given supplies and demands must be routed using at most k paths per source–sink pair. They studied the maximum k-splittable s–t flow problem, establishing tight approximation guarantees for small values of k and polynomial-time algorithms for maximum uniform k-splittable s–t flows. Subsequent work investigated congestion- and cost-based variants in the single source setting, including approximation results for $k = 2$ by Kolliopoulos [9] and for general k by Salazar and Skutella [15].

The model extends naturally to *k-splittable b-transshipments*, where each source–sink pair may use at most k paths. By pre-splitting sink demands and applying the Modified DGG Algorithm, one obtains congestion bounded by $d_{\max}/k$ for the k-splittable transshipment problem. Other optimization variants that arise in the context of k-splittable flows remain open for the k-splittable transshipment setting, suggesting several promising directions for future research.

References

1 Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.

2 Georg Baier, Ekkehard Köhler, and Martin Skutella. The k-splittable flow problem. *Algorithmica*, 42(3):231–248, 2005. doi:10.1007/S00453-005-1167-9.

3 Srinwanti Debgupta, Sarah Morell, and Martin Skutella. Unsplittable transshipments. *CoRR*, abs/2602.07230, 2026. doi:10.48550/arXiv.2602.07230.

4 Yefim Dinitz, Naveen Garg, and Michel X. Goemans. On the single-source unsplittable flow problem. *Combinatorica*, 19(1):17–41, 1999. doi:10.1007/S004930050043.

5 Lester R. Ford and Delbert R. Fulkerson. *Flows in Networks*. Princeton University Press, 1962.

6 Fabrizio Grandoni, Tobias Mömke, and Andreas Wiese. A ptas for unsplittable flow on a path. In *Proceedings of the Annual ACM Symposium on Theory of Computing*, 2022. doi:10.1145/3519935.3519959.

7 Jon M. Kleinberg. *Approximation algorithms for disjoint paths problems*. PhD thesis, Massachusetts Institute of Technology, 1996.

8 Jon M. Kleinberg. Single-source unsplittable flow. In *Proceedings of 37th Conference on Foundations of Computer Science*, pages 68–77. IEEE, 1996. doi:10.1109/SFCS.1996.548465.

9 Stavros G. Kolliopoulos. Minimum-cost single-source 2-splittable flow. *Information Processing Letters*, 94(1):15–18, 2005. doi:10.1016/J.IPL.2004.12.009.

10 Stavros G. Kolliopoulos. Edge-disjoint paths and unsplittable flow. *Handbook of Approximation Algorithms and Metaheuristics*, 2007.

11 Stavros G. Kolliopoulos and Clifford Stein. Approximation algorithms for single-source unsplittable flow. *SIAM Journal on Computing*, 31(3):919–946, 2001. doi:10.1137/S0097539799355314.

12 Mohammed Majthoub Almoghrabi, Martin Skutella, and Philipp Warode. Integer and unsplittable multiflows in series-parallel digraphs. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 427–441. Springer, 2025. doi:10.1007/978-3-031-93112-3_31.

13 Sarah Morell and Martin Skutella. Single source unsplittable flows with arc-wise lower and upper bounds. *Mathematical Programming*, 192(1):477–496, 2022. doi:10.1007/S10107-021-01704-4.

14 Prabhakar Raghavan and Clark D. Tompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7(4):365–374, 1987. doi:10.1007/BF02579324.

15 Fernanda Salazar and Martin Skutella. Single-source k-splittable min-cost flows. *Operations research letters*, 37(2):71–74, 2009. doi:10.1016/J.ORL.2008.12.004.

16 Martin Skutella. Approximating the single source unsplittable min-cost flow problem. *Mathematical Programming*, 91(3):493–514, 2002. doi:10.1007/S101070100260.

17 Chaitanya Swamy, Vera Traub, Laura Vargas Koch, and Rico Zenklusen. Unsplittable cost flows from unweighted error-bounded variants. In *2026 SIAM Symposium on Simplicity in Algorithms (SOSA)*, pages 512–523. SIAM, 2026. doi:10.1137/1.9781611978964.42.

18 Vera Traub, Laura Vargas Koch, and Rico Zenklusen. Single-source unsplittable flows in planar graphs. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 639–668. SIAM, 2024. doi:10.1137/1.9781611977912.24.

19 David P. Williamson. *Network flow algorithms*. Cambridge University Press, 2019.