SIAM J. OPTIM.
Vol. 30, No. 3, pp. 2355–2378

FACETS OF THE STOCHASTIC NETWORK FLOW PROBLEM*

ALEXANDER S. ESTES† AND MICHAEL O. BALL‡

Abstract. We study a type of network flow problem that we call the minimum-cost F-graph flow problem. This problem generalizes the typical minimum-cost network flow problem by allowing the underlying network to be a directed hypergraph rather than a directed graph. This new problem is pertinent because it can be used to model network flow problems that occur in a dynamic, stochastic environment. We formulate this problem as an integer program, and we study specifically the case where every node has at least one outgoing edge with no capacity constraint. We show that even with this restriction, the problem of finding an integral solution is NP-hard. However, we can show that all of the inequality constraints of our formulation are either facet-defining or redundant.

Key words. network flow, stochastic integer programming, F-graphs, facet-defining inequalities, directed hypergraphs

AMS subject classifications. 90C35, 90C27, 90C57

DOI. 10.1137/19M1286049

1. Background, contributions, and layout.

The minimum-cost network flow problem is a well-studied combinatorial optimization problem with many applications. In this problem, some nodes in a directed graph supply some quantity of resource while others demand quantities. The goal is to route the resource through the graph from the supply nodes to the demand nodes while minimizing costs. This problem has a well-known formulation as a linear program (LP) where the constraint matrix of this LP is totally unimodular (TU), ensuring that any extreme point of the feasible region is integral. Thus, the simplex method produces an integral optimal solution (see, for example, [3]).

Many real problems are dynamic, and decisions happen in multiple stages. When there is no uncertainty in the problem parameters, a dynamic network flow problem can be reduced to a static network flow problem on a *time-extended* graph. Each node of the time-extended network corresponds to a node of the physical network at a certain point in time. See, for example, [10, 22, 18, 1] for surveys on these types of problems.

Practical applications of this problem often have stochastic elements. For example, the time required to ship some resource along a road may be uncertain, as may be the demand for a resource in a future time period. Several approximate solution methods for stochastic, dynamic network flow problems have been studied [23, 8, 14, 6, 21, 20, 11], but there is little existing research on the polyhedral structure of these problems. Unlike the deterministic variant, stochastic network flow problems do not exhibit the same structure as a static network flow problem on a directed graph. Instead, these stochastic problems can be expressed as a deterministic problem on a directed hypergraph, specifically a class of hypergraphs called F-graphs. We discuss this connection in section 2.

*Received by the editors September 9, 2019; accepted for publication (in revised form) June 28, 2020; published electronically September 1, 2020.
https://doi.org/10.1137/19M1286049
†Institute for Mathematics and its Applications, University of Minnesota Minneapolis, MN 55455 (este0100@umn.edu, asestes1@github.io).
‡R. H. Smith School of Business and Institute for Systems Research, University of Maryland, College Park, College Park, MD, 20742 (mball@umd.edu).

Results concerning polyhedral properties of the stochastic network flow problem are known in some special cases. A characterization of the extreme points of a stochastic dynamic network that has unlimited arc capacities and has exactly one supply node and one demand node is provided in [14]. This characterization was used in a decomposition algorithm. A special class of stochastic network flow problems has been shown to have TU constraints [2]; this is described in subsection 5.1. When there are no capacity constraints on the network, the formulation that we provide is an example of a type of problem known as a *Leontief substitution system*. This connection is discussed in subsection 5.2. The extreme points of these systems were first studied in [24]. It has been shown that optimization problems on Leontief substitution systems exhibit total dual integrality when the problem parameters are integral. The most general results of this kind that we are aware of appear in [15], although earlier works provided results for more specific contexts, e.g., [19, 16]. When arc capacities are introduced, the problem is no longer a Leontief substitution system. A specialized version of the simplex algorithm for hypergraph flow problems with arc capacities was studied in [5], but no results related to integral feasible points were provided.

In a companion paper [9], we study a special case of the models analyzed in this paper. Empirical results show that the integer program can be solved by commercial solvers with almost no branching, allowing problems of significant size to be solved efficiently. These results suggest that the models analyzed in this paper represent a practical approach to the underlying class of dynamic, stochastic network flow problems.

1.1. Contributions. We provide a detailed study of the convex hull of integral solutions for the minimum-cost flow problem on a directed hypergraph. As we will demonstrate, these problems have connections to stochastic network flow problems. We show that the problem of finding an integral optimal solution to such a problem is NP-hard. However, we show that our formulation for this problem is strong. In particular, we show that each constraint is either facet-defining or redundant, and we provide necessary and sufficient conditions for each constraint to be facet-defining.

1.2. Layout. The layout of the remainder of this paper is as follows. In section 2 we define a dynamic stochastic network flow problem that motivates our interest in the F-graph flow problem. Section 3 provides a definition of an F-graph and the minimum-cost F-graph flow problem. Section 4 defines some important structures occurring in F-graphs and defines a special class of F-graphs that we refer to as ADF-graphs. This class of hypergraphs is a generalization of directed acyclic graphs (DAGs). In section 5, we discuss integrality and computational complexity properties of the F-graph flow problem. Section 6 shows that each inequality constraint of the F-graph flow problem is either facet-defining or redundant. A full-dimensional formulation whose constraints are equally strong is also provided. Conclusions are provided in section 7.

2. Motivation: Stochastic network flow problem. The problem that motivates this work is a general type of dynamic stochastic network flow problem. The problem takes place in a discretized time horizon $t = 1, \ldots, T$. In each time period, the decision maker is faced with a set of nodes. At each node, there is a quantity of resources and some outgoing edges whose costs and capacities are known. The decision maker must choose how much of each resource to assign to each edge. All of the resources present at each node must be allocated, and the capacities of the edges must be respected. At the time of allocation, it may be uncertain what will happen to the

resources allocated to each edge, as these resources will reach their destination in some later time period. The resources may be "delivered," which is to say that they are removed from the system and no longer have any effect on the problem. Otherwise, the resources allocated along the edge will eventually arrive at some destination node. However, both the destination node and the time at which the resources will arrive can be uncertain. At the time the resources are allocated to an edge, it can even be uncertain whether those resources will remain in the system and arrive at a node in a later time period or whether they will be delivered and removed from the system.

If the resources allocated to an edge do not reach a node and are instead removed from the system, we will refer to such an edge as a *delivery edge*. As we noted, it may be uncertain at the time of allocation whether or not an edge is a delivery edge. A more conventional formulation of a network flows problem would make use of "sink" or "demand" nodes instead of delivery edges. These formulations are equivalent. A problem with demand nodes can be converted into one with delivery edges simply by adding a single delivery edge to each demand node. Conversely, a problem with delivery edges can be converted into one with demand nodes by adding a single demand node to be the destination of all delivery edges. This correspondence is demonstrated more rigorously in section 3. We found that the use of delivery edges led to cleaner notation and simpler handling of feasibility concerns.

After the decision has been made in some time period t, the transition to the next time period $t+1$ can be described as follows. First, for each edge from a previous time period that reaches its destination at time $t+1$, this destination is revealed. If this edge is a delivery edge, then the resources allocated to this edge are removed from the problem. Otherwise, these arriving resources contribute to the quantity of resources available at the destination node. Second, some (possibly zero) new quantity of resource is produced at each node. Finally, the outgoing edges that are available in time $t+1$ are revealed, along with their corresponding costs and capacities. Any of these transitions can be uncertain and would be determined by an underlying stochastic process.

We use the convention in the following discussion that edges in each time period are considered to be distinct from those in other time periods; the edge that departs v in time period t and arrives at w in time period $t+1$ is considered a different edge than one that departs v in time period $t+1$ and arrives at w in time period $t+2$. Since each edge has a corresponding time period, we omit time periods from variables associated with edges.

2.1. Markov decision process formulation. The model just described can be rigorously formulated as a Markov decision process as follows. Let there be a set of nodes V. In the initial time period, for each node v there is a nonnegative quantity q_v^0 and a set of outgoing edges $E_0^+(v)$. Each edge e in $E_0^+(v)$ is either uncapacitated or has a corresponding capacity $a^0(e)$ and cost $c^0(e)$. Let E_t^+ be the set of all outgoing edges present at time t, and let $\mathscr{C}_t$ be the set of capacitated edges in time t. After the initial time period, the state of the problem in the time period t is described by

- the quantity $q_{t,v}$ of resource at each node v in V present at time t,
- the outgoing edges E_t^+ available at time t,
- the cost c_e of each edge e in E_t^+,
- the capacity a_e of each edge e in $\mathscr{C}_t(v)$,
- the set of edges R_t that were assigned flow in some previous time period that have yet to reach their destination,
- the quantity of flow f_e that was assigned to each edge e of R_t.

In each time period, the decision to be made is the amount of flow f_e to be assigned to each outgoing edge of each node, with the requirement that the entire quantity of resources present at each node is assigned to outgoing edges of the node,

$$\sum_{e \in E_t^+(v)} f_e = q_{t,v} \qquad \forall v \in V,$$

and that the flow along each edge does not exceed its capacity,

$$f_e \leq a_e \qquad \forall e \in E_t^+.$$

This incurs a cost of

$$C_t(\boldsymbol{f}) = \sum_{e \in E_t^+} c_e f_e. \tag{2.1}$$

The postdecision state is formed simply by adding the new allocations to the collection of allocated resources that have yet to reach their destination. We will use the notation R_t^f to refer to the postdecision set of edges that have received flow:

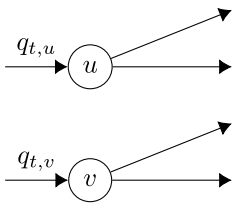
$$R_t^f := R_t \cup E_t^+.$$

Then, the postdecision state consists of the edges e of R_t^f and the associated flows f_e.

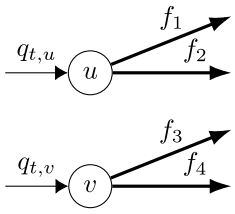
As the system transitions from one time period to the next, some edges reach their destination, some nodes receive new quantities of resources, and the new set of outgoing edges is revealed. Let Ω be the set of possible realizations of the uncertain parameters. We assume that Ω is finite, and we refer to the elements of Ω by the conventional term *scenarios*. Let $E_t^-(v;\omega)$ be the set of edges e that arrive at v in time period t under scenario ω, and let $E_t^-(\omega)$ be the set of all edges that reach their destination in time period t under scenario ω. This may include delivery edges. Let $\kappa_{t,v}(\omega)$ be the quantity of new demand that is produced in time period t at node v under scenario ω. For time period t, let $E_t^+(\omega)$ be the set of available outgoing edges under scenario ω, and let $\mathscr{C}_t(\omega)$ be the set of capacitated edges under scenario ω. There is possible notational confusion between the set of outgoing edges $E_t^+(v)$ from a node v and the set of all outgoing edges $E_t^+(\omega)$ in a scenario ω, but this should be clear from context. For each edge e in E_t, let $c_e(\omega)$ be the corresponding cost, and for each edge e in $\mathscr{C}_t$ let $a_e(\omega)$ be the corresponding capacity in scenario Ω. Then, the transitions are given by

$$\begin{aligned}
q_{t+1,v} &= \kappa_{t+1,v}(\omega) + \sum_{e:E_{t+1}^-(v;\omega)} f_e && \forall v \in V,\\
E_{t+1}^+ &= E_{t+1}^+(\omega), &&\\
c_e &= c_e(\omega) && \forall e \in E_{t+1}^+,\\
a_e &= a_e(\omega) && \forall e \in \mathscr{C}_{t+1},\\
R_{t+1} &= R_t^f \setminus E_{t+1}^-(\omega). &&
\end{aligned}$$

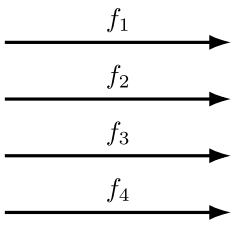
A delivery edge is not an incoming edge for any node and therefore does not contribute to $q_{t+1,v}$ for any node v. Naturally, the quantity of flow assigned in previous time periods does not change over time, so we have omitted this from the description of the transition.



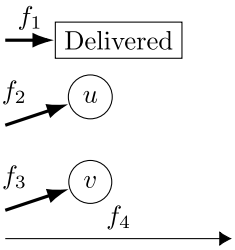
(a) Initial state: quantity of resources at each node, available edges.



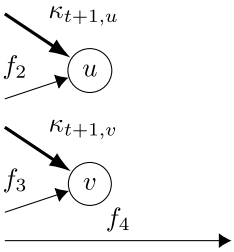
(b) Decision: quantity of resource allocated to each edge.



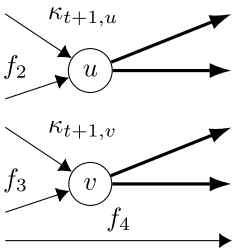
(c) Postdecision state: quantity of resources that have yet to reach their destination.



(d) Transition I: some previously allocated resources reach their destination.



(e) Transition II: new resources are produced at nodes.



(f) Transition III: new edges become available.

FIG. 1. *Stochastic network flow problem.*

The state may also include an *information state* which describes information known about future problem parameters. For example, in time period t some information may be revealed regarding which arcs will be available in time $t+1$. The form of the information state is dependent on the particular problem. The framework provided here is applicable for a broad range of stochastic processes and information states. However, we do require that all of the stochastic elements (the available outgoing edges, the destinations of edges, the quantities of resources produced, the capacities, and the costs) are independent of any decision that is made and are determined entirely by an exogenous random process. The goal is to identify a policy that minimizes the total expected costs. The problem is summarized in Figure 1.

2.2. Example: Production planning. Here, we give a more concrete example of a stochastic network flow problem. Suppose that a company has some factories where a product is manufactured and some stores where a product is sold. Each factory has a maximum quantity of resources that it can produce in each time period.

Any resources produced in a time period must either be shipped to a store or stored at the location of production. It takes a certain amount of time to ship product to a store. In each time period, the store sells some quantity of product and generates some revenue. The quantity sold cannot exceed the demand for this time period, which is not known in advance. Any unsold product must be stored at the store, which incurs a storage cost.

Let us define the parameters of the problems as follows:

T	-	number of time periods.
X	-	set of factory locations.
Y	-	set of store locations.
$m_{t,x}$	-	maximum quantity of resource that can be produced in time period t at factory x.
$c^{\text{prod}}_{t,x}$	-	per-unit cost to produce the resource at factory x in time period t.
$c^{\text{store}}_{t,z}$	-	per-unit cost to store resource at factory or store z.
$c^{\text{ship}}_{t,x,y}$	-	per-unit cost to ship the resource from factory x in time period t to arrive at store y in some later time period.
$\tau(x,y)$	-	time required to ship the resource from factory x to store y.
$r_{t,y}$	-	per-unit revenue from selling the resource at store y in time period t.
$a_{t,d}$	-	the demand at store d in time period t.

For the moment, assume that the demand $a_{t,d}$ is uncertain, while all other parameters are known in advance. This can be formulated as a stochastic network flow problem in the following fashion. The vertices are given by $V = X \cup Y$. That is, each vertex corresponds either to a factory or to a store. In each time period, $m_{t,x}$ resources are produced at each factory x, while no resources are produced at any store.

Each vertex corresponding to a factory x has the following available outgoing edges in time period t:

- a single edge $e^{\text{store}}_{t,x}$ that represents storage at x. The cost of this edge is $c^{\text{store}}_{t,x}$.
- for each store location y, an edge $e^{\text{store}}_{t,y}$ that represents shipping resources from x to y. The cost of this edge is $c^{\text{ship}}_{t,x,y}$.
- a single edge $e^{\text{unused}}_{t,x}$ that represents unused production capacity, with cost $-c^{\text{prod}}_{t,x}$. That is, producing k units less than the maximum production capacity would save $kc^{\text{prod}}_{t,x}$ units of cost.

Each vertex corresponding to a factory y in time period t has the following available outgoing edges:

- a single edge $e^{\text{store}}_{t,y}$ that represents storage at y. The cost of this edge is $c^{\text{store}}_{t,y}$.
- a single edge $e^{\text{sell}}_{t,y}$ that represents selling resources from x to y. The cost of this edge is $c^{\text{sell}}_{t,y}$, and the capacity of this edge under scenario ω is $a_{t,y}(\omega)$.

In each time period t, a single incoming edge will reach each factory x, specifically the storage arc $e^{\text{store}}_{t-1,x}$. Similarly, the storage arc $e^{\text{store}}_{t-1,y}$ will reach each store location y. In addition, for each factory x, the shipping arc $e^{\text{ship}}_{t-\tau(x,y),y}$ will reach the store location y. The edges representing sales of resources are never incoming edges of any node; these are delivery arcs. We will assume that sales are observed in time period t, so that these arcs will reach their destinations in the transition from time period t to $t+1$.

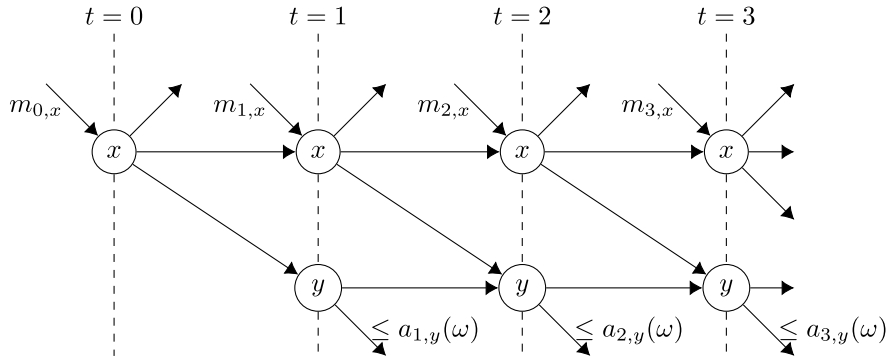


FIG. 2. *Production planning as a stochastic network.*

In this case, the network structure does not depend on the stochastic process, and we can represent the problem by an extended time space network with some uncertain parameters. Figure 2 shows the extended network for a problem instance with a single factory and a single store and where the shipping time is a single time period.

Even when the network structure is not affected by the random process, this problem is generally not equivalent to a standard network flow problem due to the stochastic elements. Decisions in later periods can incorporate more information than those in earlier time periods. For example, suppose that the demands follow two scenarios ω_1 and ω_2 that are the same in time periods 0 and 1 and then become different in time period 2. Then, decisions made in time periods 0 and 1 are made in absence of any information, while those made in time period 2 are made in response to the revealed demand information. While the problem cannot be represented by a standard deterministic network, it can be represented as a deterministic hypergraph. This is shown in Figure 3. The hyperedges are colored and presented with different line styles in order to make the figure clearer.

This framework also allows uncertainty that would affect the structure of the network. For example, we could allow travel times to be uncertain. In this case, let $\tau(t, x, y; \omega)$ be the time it takes for resources shipped from factory x at time period t to arrive at time period y. The incoming edges that arrive at store y in time period t would now be defined as follows: for each factory x and for each time period $s < t$, the edge $e^{\text{ship}}_{s,x,y}$ reaches the store y in time period t if $s + \tau(t, x, y; \omega) = t$.

2.3. Integer programming formulation. An optimal policy to a stochastic network flow problem is a solution to the stochastic program

$$\min_{\boldsymbol{f}} \sum_{e \in E_0^+} c_e f_e + \mathbb{E}\left[Q_1(\mathbf{f}; \omega)\right]$$

subject to

$$\begin{aligned} \sum_{e \in E_0^+(v)} f_e &= q_{0,v} && \forall v \in V, \\ f_e &\leq a_e && \forall e \in \mathscr{C}_0, \\ f_e &\geq 0, \end{aligned}$$

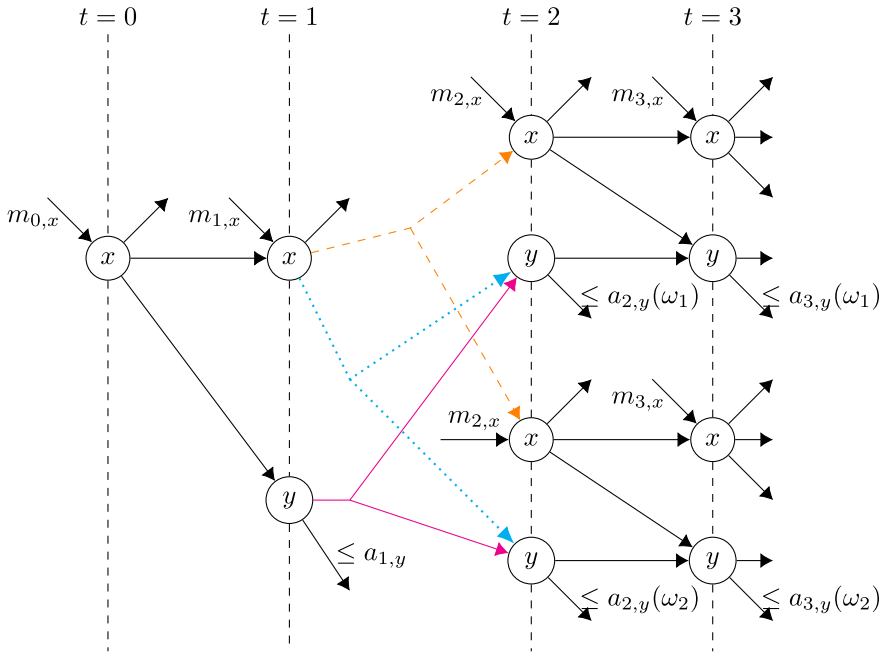


FIG. 3. *Production planning problem as a deterministic hypergraph.*

where

$$Q_t(\mathbf{f};\omega) := \min \sum_{e \in E_t^+(\omega)} c_e(\omega) f_e + \mathbb{E}\left[Q_{t+1}(\mathbf{f};\omega)\right]$$

subject to

$$\begin{aligned} \sum_{e \in E_t^+(v;\omega)} f_e &= \kappa_{t,v}(\omega) + \sum_{e : E_t^-(v;\omega)} f_e && \forall v \in V, \\ f_e &\leq a_e(\omega) && \forall e \in \mathscr{C}_t(\omega), \\ f_e &\geq 0, \end{aligned}$$

and $Q_{T+1} \equiv 0$.

Since the set Ω is assumed to be finite, the stochastic program can be rewritten as a single deterministic problem that selects all of the decisions to be made in all of the stages under all scenarios. This is called the *deterministic equivalent*. The deterministic equivalent of this problem can be formulated as a flow problem on a hypergraph. We make the typical assumption that the set of scenarios Ω is arranged in a *scenario tree*. A scenario tree is a rooted tree in which each node corresponds to a subset of the scenarios Ω, and the tth level of the tree is a partition of Ω that corresponds to the time period t and that refines the $(t-1)$th level. The interpretation is that if ω_1 and ω_2 fall in some node in the tth level of the scenario tree, it is impossible to differentiate between these two scenarios using the information available at time period t. Thus, in order for a solution to be implementable, any decision made at time t must be the same under all scenarios that fall in the same node in the tth level of the tree; we will refer to such nodes as "nodes at time t." In order to distinguish

nodes of the scenario tree Ω from the nodes of the underlying flow problem, we will always refer to the former as *scenario nodes*. We will use the notation $\mathscr{N}_t$ to refer to the set of scenario nodes at time t in the scenario tree, and we will let $p(n)$ be the sum of the probabilities of the scenarios in the scenario node n. For more on scenario trees and deterministic equivalents, the reader may consult any standard reference on stochastic programming, for example, [4].

We make the natural assumption that all quantities that are revealed at time t are the same across any scenarios that share the same scenario node at time t, since these scenarios should be indistinguishable at time t. Then, random quantities that are revealed at time t can be described as a function of scenario nodes rather than scenarios. For example, we can use the notation $\kappa_{t,v}(n)$ to describe the quantity of resource produced at the node v at time t under the scenario node n, where n is a scenario node at time t. Similar notation will apply to other random quantities.

The deterministic equivalent is then given by

$$\min_{\boldsymbol{f}} \sum_{t=0}^{T} \sum_{n \in \mathscr{N}_t} \sum_{e \in E_t^+} p(n) c_e(n) f_e(n) \tag{2.2}$$

subject to

$$\sum_{e \in E_t^+(v;n)} f_e(n) = \kappa_{t,v}(n) + \sum_{e : E_t^-(v;n)} f_e(n) \quad \forall v \in V, t \in \{0, \dots, T\}, n \in \mathscr{N}_t, \tag{2.3}$$

$$f_e(n) \le a_e(n) \qquad \forall t \in \{0, \dots, T\}, e \in \mathscr{C}_t(n), n \in \mathscr{N}_t, \tag{2.4}$$

$$f_e(n) \ge 0 \qquad \forall t \in \{0, \dots, T\}, e \in \mathscr{C}_t(n), n \in \mathscr{N}_t. \tag{2.5}$$

The constraints of this problem certainly resemble a network flow problem. Constraints (2.3) look like conservation of flow constraints, while constraints (2.4) and (2.5) look like capacity and nonnegativity constraints, respectively. As we noted in subsection 2.2, due to the presence of stochastic elements, the structure underlying this flow problem is not a standard graph. Consider an edge e that exits some node v under some scenario node n at time t. For simplicity, suppose that the scenario node n consists of two scenarios ω_1 and ω_2 that become distinguishable at time $t+1$. Suppose that e arrives at some node w in time period $t+1$ under scenario ω_1 but arrives at some node x in time period $t+1$ under scenario ω_2. Then the edge e will be treated as an incoming edge in two separate conservation of flow constraints, one corresponding to the node w under scenario ω_1 in time period $t+1$, and one corresponding to the node x under scenario ω_2 in the time period $t+1$. This is instead a network flow problem on a class of directed hypergraphs, called F-graphs, for which edges are allowed to have multiple destination nodes. In fact, the underlying networks for instances of this problem exhibit additional structure, so we can restrict our attention to a special type of F-graph called an ADF-graph. This is defined in the subsequent sections, and the connection between ADF-graphs and this problem is described in more detail in subsection 4.1.

3. Definition of F-graph and the F-graph flow problem. The minimum-cost network flow problem is defined on a directed graph. The extension that we consider is defined on a type of directed hypergraph, which is called an *F-graph*. See, for example, [13] for a survey of directed hypergraphs and their applications.

An *F-graph* H consists of a set of nodes $V(H)$ and a set of *F-edges* $E(H)$. An *F-edge* is an ordered pair (v, W) where $v \in V(H)$ and where W is a nonempty subset of $V(H)$. When there is no confusion, we will often refer to F-edges simply as edges; we may also use the term arc to refer to an F-edge or edge. The node v is referred to as the *origin node* of e while the nodes W are referred to as the *destination nodes* of e. In this work, it is convenient to allow destination nodes that are not nodes of the hypergraph and to allow edges whose set of destination nodes is empty. Such edges are referred to as *delivery edges*. The interpretation is that these edges represent delivery of resource to some customer, which removes the resource from the network. Thus, an F-edge is more accurately an ordered pair (v, W) where $v \in V(H)$ and W is a (possibly empty) subset of some set $\mathscr{W}$ that contains $V(H)$. An edge (v, W) is a delivery edge if W is disjoint from $V(H)$.

We will say that e is an *outgoing edge* of v if v is an origin node of e; we will say that e is an *incoming edge* of v if v is a destination node of e. We denote the sets of incoming and outgoing edges for a node v in an F-graph H by $E^-(v; H)$ and $E^+(v; H)$, respectively. If there is a single incoming or outgoing edge of a node v, we will sometimes denote this edge by $e^-(v; H)$ or $e^+(v; H)$, respectively.

We define a minimum-cost flow problem on F-graphs that extends the minimum-cost network flow problem on directed graphs. The parameters are similar to the network flow problem. For each edge e, there is a corresponding per-unit cost c_e. Each node v has some corresponding quantity of resource q_v that is produced at the node. Edges in a set $\mathscr{C}$ have a specified capacity, and the flow across each edge e in $\mathscr{C}$ is not allowed to be greater than some value a_e. Edges not in $\mathscr{C}$ are referred to as uncapacitated edges. The *minimum-cost F-graph flow* problem, or *F-graph flow* problem for short, is defined by

$$\min \sum_{e \in E(H)} c_e f_e$$

subject to

$$\sum_{e \in E^+(v;H)} f_e = q_v + \sum_{e \in E^-(v;H)} f_e \qquad \forall v \in V(H), \tag{3.1}$$

$$f_e \le a_e \qquad \forall e \in \mathscr{C}, \tag{3.2}$$

$$f_e \ge 0 \qquad \forall e \in E(H). \tag{3.3}$$

The only difference between the standard network flow problem and this problem is that edges are F-edges instead of standard directed edges. For a given F-graph H, a vector of quantities $\mathbf{q}$, a set of capacitated edges $\mathscr{C}$, and a vector of edge capacities $\mathbf{a}$, we denote the corresponding polytope defined by the constraints of problem by $\mathscr{P}_1(H, \mathbf{q}, \mathscr{C}, \mathbf{a})$. A flow problem has all of these parameters with the addition of a vector of costs $\mathbf{c}$. We will denote such a flow problem by $\mathscr{F}(H, \mathbf{q}, \mathscr{C}, \mathbf{a}, \mathbf{c})$.

In the typical setup of the minimum-cost network flow problem there are no delivery edges, the sum of q_v across the nodes is zero, and the goal is to route resources from nodes with positive q_v to nodes with negative q_v. We will instead require that q_v is nonnegative for each node so that the goal is to route resources from the nodes with positive q_v to delivery arcs, which allow these resources to exit the network. These alterations do not cause any loss of generality. Consider a formulation $\mathscr{P}$ of the problem where q_v sums to zero across the nodes v and the network has no delivery edges. This can be translated into a formulation $\mathscr{P}'$ for the new setting as

follows. For each node v, define a new quantity by $q'_v = q_v$ if q_v is positive and $q'_v = 0$ otherwise. For each node v such that q_v is negative, add a single outgoing delivery edge $d(v)$ from v with capacity $|q_v|$ and with cost $c_{d(v)} = 0$. Note that in any feasible solution, the amount of flow on this delivery edge must be q_v. Then, any feasible solution $\boldsymbol{\lambda}$ for $\mathscr{P}$ has a corresponding feasible solution $\boldsymbol{\lambda}'$ for $\mathscr{P}'$ that is formed by setting $f_{d(v)} = q_v$ on the aforementioned delivery edges and leaving all other variables unchanged. Conversely, any feasible solution $\boldsymbol{\lambda}'$ for $\mathscr{P}'$ has a corresponding feasible solution $\boldsymbol{\lambda}$ for $\mathscr{P}$ that is formed by omitting the variables $f_{d(v)}$ on these delivery edges. Thus, these formulations are equivalent.

For some F-graph H, a *subgraph* S of H is an F-graph such that $V(S) \subseteq V(H)$ and such that $E(S) \subseteq E(H)$. In a manner consistent with the definitions given above, the origin node of every edge in $E(S)$ must be contained in $V(S)$, but the destination nodes need not be. If S is a subgraph of H and $E(S) \neq E(H)$, then we will say that S is a *proper subgraph*. We use the notation $S \subseteq H$ to indicate that S is a subgraph of H, while $S \subset H$ denotes that S is a proper subgraph of H. For a set of edges $\mathcal{E} \in E(H)$, we say the *induced subgraph* $H[\mathcal{E}]$ is the subgraph such that

$$V(H[\mathcal{E}]) = \{v \in V(H) : v \text{ is an origin node for some edge } e \in \mathcal{E}\}$$

and

$$E(H[\mathcal{E}]) = \mathcal{E}.$$

In some cases, we will discuss a flow problem on a subgraph S of some graph H in which all relevant parameters take the same value as a flow problem $\mathscr{F}(H, \mathbf{q}, \mathscr{C}, \mathbf{a}, \mathbf{c})$ on the entire graph. In this case, we denote the flow problem on the subgraph as $\mathscr{F}(S, \mathbf{q}, \mathscr{C}, \mathbf{a}, \mathbf{c})$ with the understanding that only the elements of $\mathbf{q}, \mathscr{C}, \mathbf{a}$, and $\mathbf{c}$ relevant to S are used. Similar notation applies to the polytope $\mathscr{P}_1$.

4. Paths, compaths, branchings, and ADF-graphs. Two fundamental structures in F-graphs are *paths* and *compaths*. These extend the notion of a path in a network. A *path* P in an F-graph H is the induced subgraph for a set of edges $e_0 = (v_0, W_0), e_1 = (v_1, W_1) \ldots, e_k = (v_k, W_k)$ such that each edge e_j is in $E(H)$ and such that $v_j \in W_{j-1}$ for each j from 1 to k. A path *starts at* v if v is the origin node of the first edge of P and *ends at* w if w is a destination node of the last edge in P. For two nodes v and w, we say that the path P is a *path from* v *to* w if the path starts at v and ends at w. This definition is in accordance with existing sources, for example, [13].

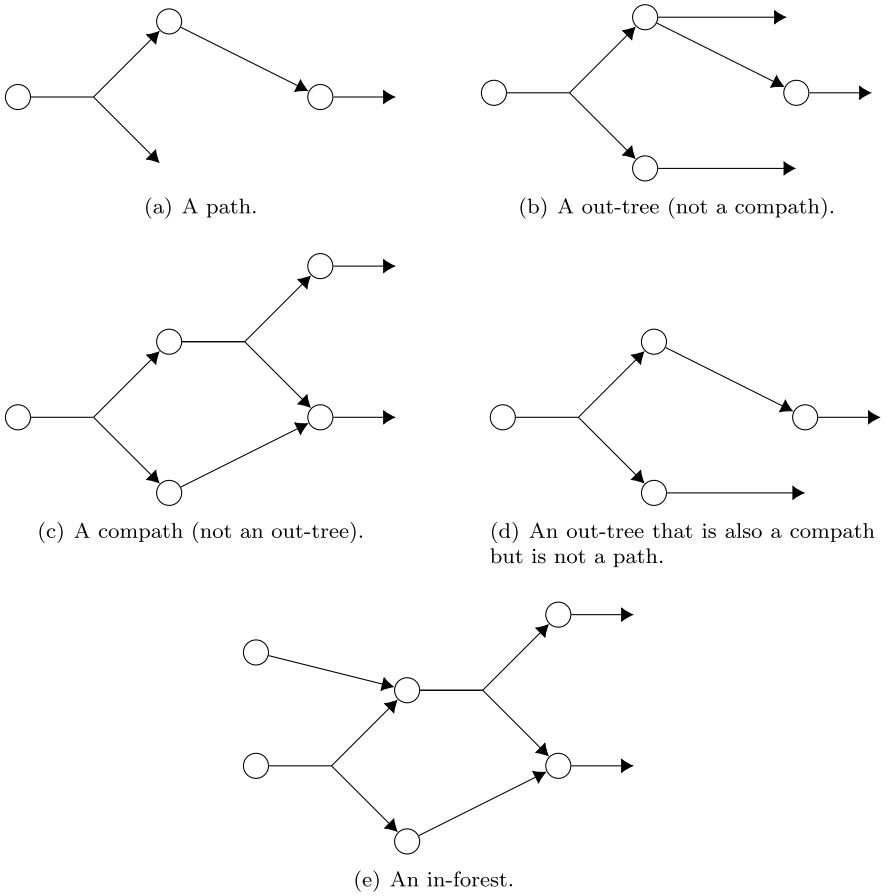
For a node v, a *compath* P *starting at* v is a subgraph of H such that v is the unique node of $V(P)$ with no incoming edge in $E(P)$ and every node of $V(P)$ has exactly one outgoing edge in $E(P)$. Note that every path starting at v is a compath starting at v. A compath (or path) P starting at v is *maximal* if there is no compath (or path) P' starting at v such that $P \subset P'$. The term compath was introduced in [14] for the stochastic network flow setting, where it was shown that the extreme points of a special type of stochastic network are compaths (more specifically, maximal compaths). The definition that we give here generalizes that notion.

For a node v, an *out-tree rooted at* v is a subgraph T such that every node of $V(T)$ except for v has exactly one incoming edge, while v has no incoming edge. This extends the notion of an out-tree (also called an arborescence or branching) in a directed graph. Existing works have studied the polyhedra of out-trees in graphs and the problem of generating an optimal spanning out-tree; see, for example, [7] or

[12]. We similarly extend the notion of an in-forest (also called an anti-arborescence). An *in-forest* is a subgraph T such that every vertex of $V(T)$ has exactly one outgoing edge in $E(T)$. A *spanning in-forest* is an in-forest that contains every node of the hypergraph.

Note that while the root of an out-tree is the unique minimal node of that tree, in-forests may have multiple maximal nodes. This is due to an asymmetry inherent in the definition of an F-graph: edges have multiple destination nodes but only a single origin node. Further note that in the definitions given here, the root of an out-tree has no incoming edge, but the maximal nodes of an in-forest must have an outgoing edge. The outgoing edge of a maximal node of an in-forest T must be a delivery edge T (i.e., has no destination nodes in $V(T)$), and any maximal edge of a spanning in-forest must be a delivery edge of both T and the F-graph of which T is a subgraph. Examples of paths, compaths, in-forests, and out-trees are given in Figure 4.

Every path starting at v is also a compath, an out-tree rooted at v, and an in-forest. Every compath is also an in-forest. However, a compath is not necessarily an out-tree, and vice versa. An F-graph has the *disjoint reachability* property if every compath is an out-tree (this property was discussed in [15, 19]). Venn diagrams summarizing the relationships between paths, compaths, in-trees, and out-trees are shown in Figure 5.



(a) A path.

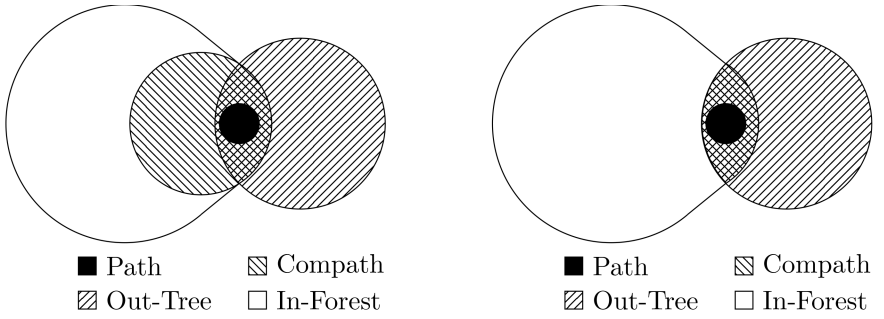
(b) A out-tree (not a compath).

(c) A compath (not an out-tree).

(d) An out-tree that is also a compath but is not a path.

(e) An in-forest.

FIG. 4. *Paths, compaths, out-trees, and in-forests.*



(a) Relationships between structures in F-graphs without disjoint reachability

(b) Relationships between structures in F-graphs with disjoint reachability

FIG. 5. *Relationship between paths, compaths, out-trees, and in-forests.*

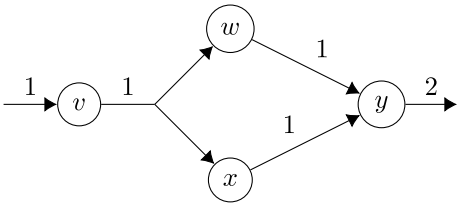


FIG. 6. *F-graphs without disjoint reachability allow resources to be created.*

Consider a network with a single supply node and a single delivery edge, and suppose that there is exactly one unit of resource available at the supply node. A solution to an F-graph flow problem would route this unit of resource from the supply node through the network to the delivery edge. If the F-graph has a compath that is not an out-tree, then there may be solutions in which more than a single unit of resource is delivered. This is demonstrated in Figure 6. Within the context of stochastic network flow problems as described in section 2, this behavior seems unusual. We would not expect that resources could be created simply by moving a single unit of resources through the network. Indeed, the F-graphs that arise from the aforementioned stochastic network flow problems have the disjoint reachability property. In this case, hyperedges represent points at which two scenarios become distinct. These scenarios should not subsequently interact with each other.

An *acyclic* F-graph is an F-graph such that whenever there is a path from node v to node w, there is no path from w to v. This extends the notion of acyclicity in directed graphs. The F-graphs underlying stochastic network flow problems are acylic because every node has an associated time period and edges move forward in time. We will refer to acyclic F-graphs with disjoint reachability as *ADF-graphs*. Note that in a DAG, every compath is a path and every path is an out-tree. Thus, ADF-graphs are a generalization of DAGs.

Similarly to DAGs, acyclic F-graphs have a *natural partial ordering* on their nodes and edges. This ordering is defined as follows:

- For v and w in $V(H)$, $v \prec^H w$ if there is a path from v to w.
- For v in $V(H)$ and $e = (v', W)$ in $E(H)$, $v \prec^H e$ if $v = v'$ or there is a path from v to v'; $e \prec^H v$ if there is a path from v' to v.
- For $e = (v_1, W_1)$ and $\eta = (v_2, W_2)$ in $E(H)$, $e \prec^H \eta$ if there exists some node $w \in W_1$ such that $w \preceq^H v_2$.

There are alternate characterizations of paths and compaths in terms of the natural partial ordering. A subgraph P of H is a path if and only if the nodes and edges of P are totally ordered under the partial ordering $\prec^P$. Any compath P starting at v is the induced subgraph for a set of edges $\mathscr{E}$ such that no two edges share the same origin node and such that v is the unique minimal node under the partial ordering $\prec^P$. A compath (or path) P is maximal if and only if any maximal edge e under the partial ordering $\prec^P$ is also a maximal edge under the partial ordering $\prec^H$. It will be convenient to have compact notation for the subgraph induced by the elements preceding some node v in some F-graph H according to the natural partial ordering. We will let $L(v;H)$ be the subgraph of H given by

$$V(L(v;H)) = \left\{w \in E(H) : w \preceq^H v\right\},$$
$$E(L(v;H)) = \left\{e \in E(H) : e \prec^H v\right\}.$$

In a similar fashion, we define $U(v;H)$ to be the subgraph where

$$V(U(v;H)) = \left\{w \in E(H) : w \succeq^H v\right\},$$
$$E(U(v;H)) = \left\{e \in E(H) : e \succ^H v\right\}.$$

For an edge e of H, the subgraphs $L(e;H)$ and $U(e;H)$ are defined similarly.

When an F-graph is acyclic, there is an alternative characterization of the disjoint reachability property that is useful.

LEMMA 4.1. *Let H be an acyclic F-graph. Then H has disjoint reachability if and only if $\{U(w;H) : w \in W\}$ is a collection of disjoint sets for every edge $e = (v, W)$ in $E(H)$.*

A proof is provided in the supplementary material, section SM1. The supplementary material is available at asestes1.github.io/files/fgraph_siopt_supplement.pdf. An important property of ADF-graphs is that certain subgraphs are DAGs with delivery edges. In particular, any ADF-graph with a unique maximal node is a DAG.

LEMMA 4.2. *Let H be an ADF-graph with a unique maximal node ρ. Then for all edges $e = (v, W)$ in $E(H)$ it is true that $|W \cap V(H)| \leq 1$.*

Proof. Suppose that there is some arc (v, W) in $E(H)$ where $|W \cap V(H)| \geq 2$. Let w and w' be distinct elements of $W \cap V(H)$. By definition, $w \preceq^H \rho$ and $w' \preceq^H \rho$. This implies that $U(w;H)$ and $U(w';H)$ share the node ρ. Lemma 4.1 implies that this is impossible. ∎

As indicated below, this result implies that the constraint matrix of a flow problem on an ADF-graph is TU when that graph has a unique maximal node.

LEMMA 4.3. *Let there be an ADF-graph with a unique maximal node ρ. The constraint matrix of the corresponding flow problem is TU.*

Proof. This follows almost immediately from Lemma 4.2 and the well-known property that the constraint matrix of the minimum-cost network flow problem on a directed graph is TU. There is a small detail that must be taken care of: ADF-graphs may have delivery edges while typical directed graphs do not. However, it is easy to see that the constraint matrix of a flow problem on a DAG H with delivery edges is a submatrix of a flow problem on a standard DAG G in which a node has been placed at the end of each delivery edge of H. Thus, the constraints will be TU. ∎

4.1. ADF-graphs and stochastic network flow. Recall the stochastic network flow problem described in section 2, whose formulation is given by

$$\min \sum_{t=0}^{T} \sum_{n \in \mathscr{N}_t} \sum_{e \in E_t^+} p(n) c_e(n) f_e(n)$$

subject to

$$\begin{aligned}
\sum_{e \in E_t^+(v;n)} f_e(n) = \kappa_{t,v}(n) + \sum_{e : E_t^-(v;n)} f_e(n) \quad & \forall v \in V, t \in \{0, \ldots, T\}, n \in \mathscr{N}_t, \\
f_e(n) \le a_e(n) \quad & \forall t \in \{0, \ldots, T\}, e \in \mathscr{C}_t(n), n \in \mathscr{N}_t, \\
f_e(n) \ge 0 \quad & \forall t \in \{0, \ldots, T\}, e \in \mathscr{C}_t(n), n \in \mathscr{N}_t.
\end{aligned}$$

This is a network flow problem on an ADF-graph that is defined as follows. Let

$$V(H) = \{(t, n, v) : t \in \{1, \ldots, T\}, n \in \mathscr{N}_t, v \in V\}.$$

The interpretation of node (t, n, v) of $V(H)$ is that it represents the node v of V at time t under the scenarios in the scenario node n. For each edge e of $E_t^+(v; \omega)$, there is a corresponding F-edge η in $E(H)$. The origin node of η is $(t, n_t(\omega), v)$ where $n_t(\omega)$ is the scenario node at time t containing ω. The destination nodes of η are the set

$$\{(\tau, n_\tau(\omega), w) : e \in E_\tau^-(w; n_\tau(\omega))\}.$$

The F-edge η represents the uncertainty in the network edge e; the multiple destination nodes represent the possible destinations of the edge e under the various scenarios. The cost c_η of each F-edge η is simply the cost c_e of the corresponding network edge e. Similarly, η is in $\mathscr{C}$ if and only if e is in $\mathscr{C}_t$, and the capacity a_η is set equal to a_e. It is straightforward to verify that this graph is an ADF-graph.

Since graphs resulting from stochastic network flow problems are acyclic and have disjoint reachability, for the most part we will restrict our study to this class of graphs.

5. Integrality and computational complexity. In some special cases, there are existing results that imply integrality properties or properties related to the computational complexity of finding optimal solutions.

5.1. Total unimodularity. In general, the constraint matrix corresponding to an F-graph flow problem may not be TU. Restricting our attention to ADF-graphs still does not guarantee a TU constraint matrix. A simple example is given in Figure 7.

When all edges are unconstrained, the constraint matrix for this flow problem is given by

$$\begin{array}{c} \\ u \\ v \\ w \\ x \end{array} \begin{array}{c} \begin{array}{ccccc} e_1 & e_2 & e_3 & d_1 & d_2 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 1 & 0 \\ -1 & 0 & -1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \end{pmatrix} \end{array}.$$

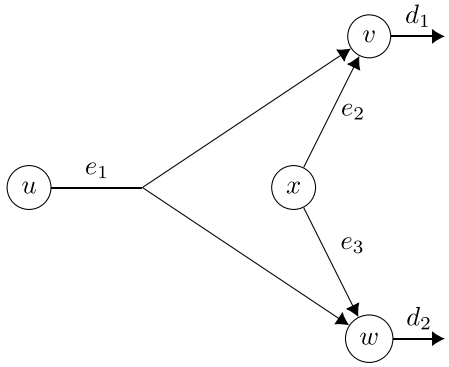


FIG. 7. *An ADF-graph whose corresponding constraint matrix is not TU.*

This has the submatrix

$$\begin{array}{c} \\ v \\ w \\ x \end{array} \begin{array}{c} \begin{array}{ccc} e_1 & e_2 & e_3 \end{array} \\ \begin{pmatrix} -1 & -1 & 0 \\ -1 & 0 & -1 \\ 0 & 1 & 1 \end{pmatrix} \end{array}$$

which has a determinant of -2.

We can provide necessary conditions that are satisfied by any F-graph flow problem with a TU constraint matrix. An *undirected hypergraph* H consists of a set of vertices $V(H)$ and a set of *undirected hyperedges* $E(H)$, where an undirected hyperedge is a subset of $V(H)$. For a given F-graph H, we define the *underlying undirected hypergraph* to be the undirected hypergraph H^* with vertices given by $V(H^*) := V(H)$ and with edges defined by

$$E(H^*) := \{W \cup \{v\} : (v, W) \in E(H)\}.$$

There is a natural correspondence ϕ between the edges of an F-graph H and the edges of its underlying undirected hypergraph, given by

$$\phi(v, W) := W \cup \{v\}.$$

Note that if H is acyclic, then this correspondence is a bijection (this is not necessarily true in general).

We require the concept of a *loose cycle* in an undirected hypergraph. This is a sequence of edges $(e_1, \ldots, e_k)$ such that

- $|e_i \cap e_{i+1}| = 1$ for $i \in 1, \ldots, k-1$ and $|e_1 \cap e_k| = 1$;
- $e_i \cap e_j = \emptyset$ for any $i, j \in 1, \ldots, k$ such that $i \neq j+1$ and $j \neq i+1$.

In other words, a loose cycle is a sequence of edges such that consecutive edges have a single common node while nonconsecutive edges do not share any nodes (in this context, the first and last edges would be considered consecutive). Note that this is a generalization of a cycle in an undirected graph. Loose cycles (as well as other types of hypergraph cycles) have been studied in the context of extremal combinatorics; see, for example, [25]. We will say that a set of edges in an F-graph forms a loose cycle if the corresponding edges in the underlying undirected hypergraph form a loose cycle. For example, the sequence of edges (e_1, e_2, e_3) form a loose cycle in the network displayed

in Figure 7. Given a loose cycle $(e_1, \ldots, e_k)$ in an F-graph, we say that an edge e_i is a *sideways edge* of the loose cycle if $e_i \cap e_{i+1}$ and $e_i \cap e_{i-1}$ are both destination nodes of e_i (here, we use the convention that $e_0 = e_k$ and $e_{k+1} = e_1$ so that this definition can be applied to the first and last edges of the cycle). For example, in the network displayed in Figure 7, edge e_1 is a sideways edge of (e_1, e_2, e_3) while edges e_2 and e_3 are not. We provide the following necessary condition.

PROPOSITION 5.1. *If an F-graph has a loose cycle with an odd number of sideways edges, then the corresponding flow problem does not have a TU constraint matrix.*

A proof is provided in the supplementary material, section SM4. The supplementary material is available at asestes1.github.io/files/fgraph_siopt_supplement.pdf. We conjecture that this condition is also sufficient.

Conjecture 5.2. If an F-graph does not have a loose cycle with an odd number of sideways edges, then the corresponding flow problem has a TU constraint matrix.

5.1.1. A TU special case. We are aware of one special class of F-graph flow problems with a TU constraint matrix. This example was adapted from a two-stage network flow model presented in [2], which was shown to be TU in the same work. The existing model addresses the following problem. There is some predefined quantity of resource that will be produced at some origin in each time period of a time horizon. This resource can either be stored at the origin or shipped to a processing location. The amount of resource that can be processed in each time period is uncertain. If the amount of resources present at the processing location exceeds the processing capacity in some time period, then the excess resources can be stored. There are costs of storing resource at the origin and at the processing location, as well as costs of processing that may change over time. Potential applications of this model include oil production planning and air traffic flow management.

We make some slight modifications to this existing model in order to produce a capacitated F-graph flow problem that is also TU. Let Ω be a finite set of scenarios, where each scenario provides a realization of the capacity of the destination in each time period. Let T be the time horizon. Let C_{ts} be the capacity of the processing location in time period t under scenario s. For each time period t in the time horizon, there is a node o_t that represents the origin of the resource in time period t. In addition, for each time period t in T and scenario s in Ω, there is a node d_{ts} that represents the processing location in time period t and scenario s. We define four sets of edges. One set of edges represent the quantities of resources that will be sent to the destination in some time period. This consists of T edges, each of which is of the form

$$(o_t, \{d_{ts} : s \in \Omega\})$$

for some time period t in T. The second set of edges represent resources that are stored at the origin. This again consists of T edges, each of the form

$$(o_t, \{o_{t+1}\})$$

for some time period t. These edges represent the resources stored at the origin between time periods t and $t+1$. The third set of edges similarly represents resources stored at the processing location and consists of $T|\Omega|$ edges, each of the form

$$(d_{ts}, \{d_{t+1,s}\})$$

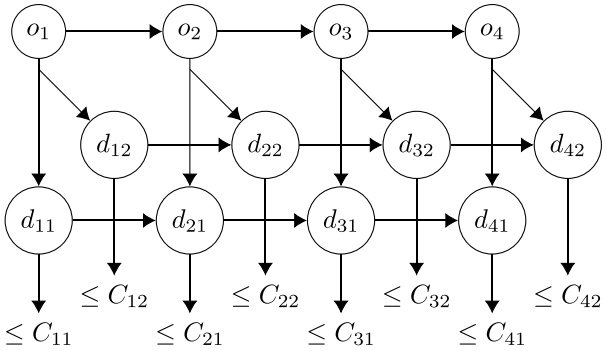


FIG. 8. *An F-graph representation based on the model in* [2].

for some time period t and scenario s. The fourth and final set of edges represents resource processed in each time period t. These edges are delivery edges. There are $T|\Omega|$ of these edges taking the form

$$(d_{ts}, \emptyset)$$

with capacity C_{ts} for some time period t and scenario s. Figure 8 depicts the resulting F-graph when $|\Omega| = 2$ and when $T = 4$. The fact that this class of F-graph flow problems is TU follows nearly immediately from the fact that the model presented in [2] is TU; we omit the details for the sake of brevity. Here, we have specified the capacitated edges in a manner consistent with the existing work. However, the TU property will hold regardless of which edges are uncapacitated and which capacitated.

5.2. Leontief substitution systems. Existing results from [15] imply that F-graph flow problems with no capacity constraints have integer extreme solutions. More specifically, a matrix A is a *Leontief matrix* if every column of A has exactly one positive element and there exists some nonnegative real vector x^* such that $Ax^* > 0$. A *Leontief substitution system* is a system of constraints that takes the form

$$\begin{aligned} Ax &= b, \\ x &\geq 0, \end{aligned}$$

where A is a Leontief matrix and where b is a nonnegative real vector.

We can show that, barring degenerate cases, any instance of the uncapacitated minimum-cost F-graph flow problem is a special case of a Leontief substitution system. If a network had no delivery edges, then any supplied resources would never be able to exit the network. This would imply that the problem is either infeasible or has a trivial unique feasible solution in which every edge receives zero flow. We can therefore assume that there is at least one delivery edge. Under this assumption, every uncapacitated minimum-cost F-graph flow problem is a Leontief substitution system.

PROPOSITION 5.3. *Let $\mathscr{P}_1(H, \boldsymbol{q}, \mathscr{C}, \boldsymbol{a}, \boldsymbol{c})$ be the system of constraints for an F-graph flow problem that satisfies the following:*

1. *H has at least one delivery edge;*
2. *$\mathscr{C}$ is empty.*

This system of constraints is a Leontief substitution system.

A proof is given in the supplementary material, section SM5. The supplementary material is available at asestes1.github.io/files/fgraph_siopt_supplement.pdf.

In [15], it is shown for Leontief substitution systems that if the constraint matrix A is integral, then there is a value iteration algorithm that converges in strongly polynomial time to an optimal solution. If, in addition, the right-hand-side vector b is integral, then any extreme point is integral. Thus, optimal integral solutions to uncapacitated F-graph flow problems can be found in polynomial time.

5.3. NP-completeness of capacitated and semi-uncapacitated problems. When capacity constraints are placed on some arcs of the network, then the F-graph flow problem is no longer a Leontief substitution system. Furthermore, we can show that problem of finding an optimal solution to an F-graph flow problem with capacity constraints is NP-hard. This remains true if we restrict our attention to ADF-graph flow problems.

In fact, we will prove a slightly stronger property: finding an optimal solution to an ADF-graph flow problem is NP-hard even when every node of the ADF-graph flow problem has an uncapacitated outgoing edge. We will refer to such flow problems as *semi-uncapacitated*. Recall that a spanning in-forest X is a set of edges such that every node has exactly one outgoing edge in X. Note that if a flow problem is semi-uncapacitated, we can choose an spanning in-forest that consists entirely of uncapacitated edges. Conversely, if there exists a spanning in-forest X that uses only uncapacitated edges, then the flow problem is necessarily semi-uncapacitated since every node has an outgoing edge in X and every edge of X is uncapacitated. In many results, we make the assumption that the flow problem has an uncapacitated spanning in-forest rather than the equivalent assumption that the problem is semi-uncapacitated.

Semi-uncapacitatedness is a useful property because it ensures that feasible solutions for flow problems on subgraphs can always be extended into feasible solutions for the entire graph. Lemma 5.4 states that given a feasible solution ϕ to a subnetwork and an uncapacitated spanning in-forest X for the entire network, it is possible to form a solution ϕ' that is feasible for the entire network and that takes the same values as ϕ on all edges except those in the spanning in-forest X. Furthermore, if ϕ is integral, it is possible to ensure that ϕ' is also integral. In the statement of the following lemma, we use the convention that, given a point $\boldsymbol{\lambda}$ in an F-graph flow polytope $\mathscr{P}_1(H, \boldsymbol{q}, \mathscr{C}, \boldsymbol{a})$, the notation $f_e(\boldsymbol{\lambda})$ refers to the value of the variable f_e in the point $\boldsymbol{\lambda}$.

LEMMA 5.4. *Let H be an F-graph, let $\boldsymbol{q}$ be a vector of demands, let $\mathscr{C}$ be a subset of $E(H)$, and let $\boldsymbol{a}$ be a vector of edge capacities. Suppose that there exists a spanning in-forest X that consists of edges in $E(H) \setminus \mathscr{C}$. Let $\boldsymbol{\phi}$ be an (integral) point in $\mathscr{P}_1(S, \boldsymbol{q}, \mathscr{C}, \boldsymbol{a})$ for some subnetwork S of H. Then there exists an (integral) point $\boldsymbol{\phi}'$ in $\mathscr{P}_1(H, \boldsymbol{q}, \mathscr{C}, \boldsymbol{a})$ such that $f_e(\boldsymbol{\phi}) = f_e(\boldsymbol{\phi}')$ for all edges $e \in E(S) \setminus X$ and such that $f_e(\boldsymbol{\phi}') = 0$ for all edges $e \in E(H) \setminus (X \cup E(S))$.*

A proof of Lemma 5.4 can be found in the supplementary material, which is available at asestes1.github.io/files/fgraph_siopt_supplement.pdf.

THEOREM 5.5. *The problem of deciding whether there exists an integral solution to the semi-uncapacitated ADF-graph flow problem with objective of at least k is NP-complete.*

Proof. The semi-uncapacitated ADF-graph flow problem is in NP since it is a special case of an integer programming problem, which is known to be in this class.

We prove that this problem is NP-complete by reduction from set packing. An instance of the set packing problem is defined by a finite universe of items $\mathscr{U}$, a collection $\mathscr{S}$ of subsets of $\mathscr{U}$, and a positive integer k. The goal is to determine

whether or not there exist k disjoint elements of $\mathscr{S}$. This problem is well-known to be NP-complete; it is one of Karp's 21 NP-complete problems [17].

Let there be a finite universe $\mathscr{U}$ of items, and let $\mathscr{S}$ be a collection of subsets of $\mathscr{U}$. Define a corresponding F-graph problem $\Gamma(\mathscr{U},\mathscr{S}) = \mathscr{F}(H,\mathbf{q},\mathscr{C},\mathbf{a},\mathbf{c})$ as follows. The graph has one node for each element of $\mathscr{S}$ and one node for each item of $\mathscr{U}$. The edges are defined so that every node has two edges, one of which is an uncapacitated delivery edge with cost 0. For each node corresponding to some element S of $\mathscr{S}$, the other outgoing edge has destination nodes $\{u \in \mathscr{U} : u \in S\}$. In other words, this edge connects the node corresponding to the set S with the nodes corresponding to the elements in S. The cost of this edge is $|S| - 1$, and its capacity is 1. For nodes corresponding to items in $\mathscr{U}$, the second outgoing edge is a delivery edge with capacity 1 and cost -1. Every node corresponding to an element of $\mathscr{S}$ has a demand quantity of 1, while those nodes corresponding to elements in $\mathscr{U}$ have a demand quantity of 0. It can be easily seen that the size of $\Gamma(\mathscr{U},\mathscr{S})$ is bounded by a polynomial of the size of $(\mathscr{U},\mathscr{S})$. Following this construction, we believe the proof to be clear.

An example of this construction is provided in Figure 9. This figure shows the corresponding F-graph flow problem for a set packing problem that is defined as follows. The universe of items $\mathscr{U}$ is the set $\{a,b,c,d\}$. The collection of subsets $\mathscr{S}$

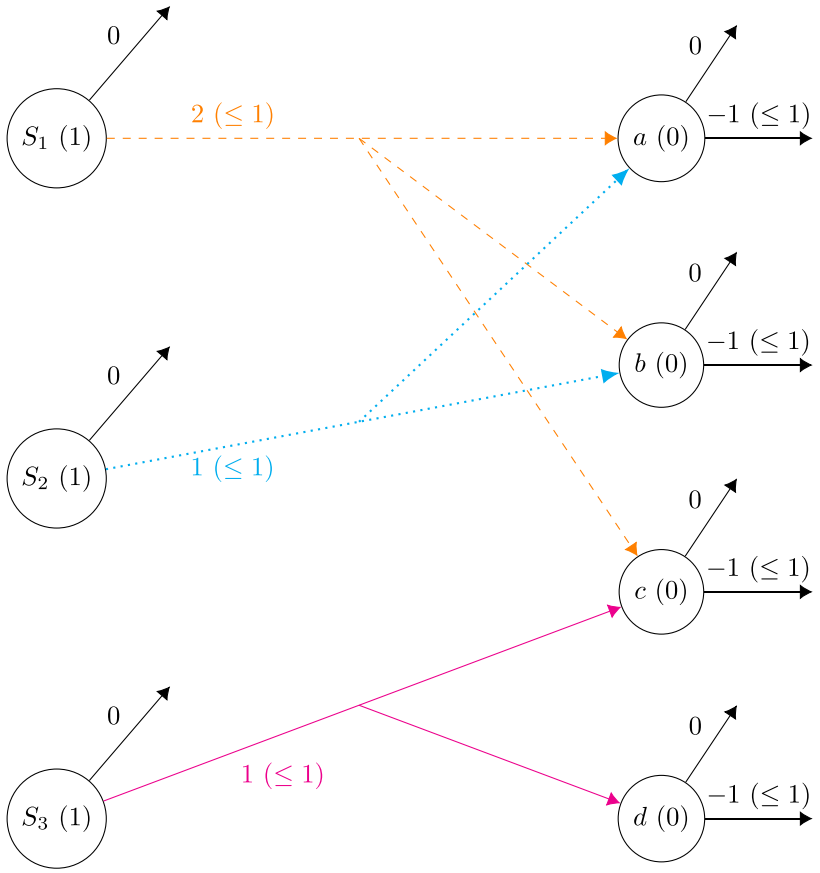


FIG. 9. *Semi-uncapacitated F-graph flow problem constructed from set packing problem,* $\mathscr{U} = \{a,b,c,d\}$, $\mathscr{S} = \{\{a,b,c\},\{a,b\},\{c,d\}\}$.

.

has three elements: $S_1 = \{a, b, c\}$, $S_2 = \{a, b\}$, and $S_3 = \{c, d\}$. The value shown in parenthesis for each node is the corresponding quantity of resource present at that node. The value on each edge outside of the parentheses is the cost of the edge, while the value within the parentheses is the edge capacity (if the edge is capacitated). ∎

Theorem 5.5 shows that it is NP-complete to find an integral solution to the semi-uncapacitated ADF-graph flow problem with objective of at least k, which implies that it is NP-hard to find an integral solution of minimal cost.

6. Full-dimensional formulation with facet-defining inequalities. In this section, we first provide a full-dimensional projection of the polytope for the semi-uncapacitated F-graph flow problem. There is some generality lost by including the restriction that the problem is semi-uncapacitated. However, there is a broad range of application and cases to which our results apply. For example, in many cases, it is always possible to dispose of resources (for some cost), and this disposal would be represented by an uncapacitated arc. This would lead to a semi-uncapacitated problem. In most of the following results, semi-capacitatedness is not explictly mentioned. Instead, the results' stated requirements include the existence of an uncapacitated spanning in-forest. This is equivalent, since an uncapacitated spanning in-forest exists if and only if the problem is semi-uncapacitated.

With this restriction, we show that all of the constraints of the F-graph flow polytope are facet-defining and that each constraint has a corresponding constraint in the full-dimensional projection that is also facet-defining. We do make some assumptions to avoid degenerate cases. Specifically we assume the following:

1. The capacity a_e for each capacitated edge e in $\mathscr{C}$ is strictly positive.
2. For every edge $e = (w, V)$ such that $q_w = 0$, there is a path from some node v to w such that the supply q_v is strictly positive.

Assumption 1 can be made because if $a_e = 0$ for some edge e, then no flow can be assigned to that edge, and there is an equivalent flow problem in which the edge is removed. Assumption 2 can also be made because if there is no path from any node with positive demand to an edge e, then there can be no flow along the incoming edges of e. Thus, if there is no supply available at the origin node, there can be no flow assigned to the edge e, and there is an equivalent flow problem in which the edge is removed. We will refer to these assumptions as the *nondegeneracy* assumptions.

6.1. Full-dimensional formulation. Consider an ADF-graph H, a set of capacitated edges $\mathscr{C}$, and a spanning in-forest X of H such that none of the edges of X is capacitated. Then, given vectors of supplies $\mathbf{q}$ and capacities $\mathbf{a}$, recall that the polytope $\mathscr{P}_1(H, \mathbf{q}, \mathscr{C}, \mathbf{a})$ was defined in section 3 to be

$\mathscr{P}_1(H, \mathbf{q}, \mathscr{C}, \mathbf{a})$:

$$\begin{aligned} \sum_{e \in E^+(v;H)} f_e &= q_v + \sum_{e \in E^-(v;H)} f_e && \forall v \in V(H), \\ f_e &\le a_e && \forall e \in \mathscr{C}, \\ f_e &\ge 0 && \forall e \in E(H). \end{aligned}$$

It is possible to manipulate the equality constraints to isolate the flow variables corresponding to edges in X. We claim that the resulting polyhedron is given by

$\mathscr{P}_2(H, \mathbf{q}, \mathscr{C}, \mathbf{a}, X)$:

$$f_{e^+(v,X)} = \sum_{w:w\preceq^X v} \left(q_w + \sum_{e\in E^-(w;H)\setminus X} f_e - \sum_{e\in E^+(w;H)\setminus X} f_e \right) \quad \forall v \in V(H), \tag{6.1}$$

$$f_e \le a_e \quad \forall e \in \mathscr{C}, \tag{6.2}$$

$$f_e \ge 0 \quad \forall e \in E(H). \tag{6.3}$$

This polytope is exactly equal to $\mathscr{P}_1(H, \mathbf{q}, \mathscr{C}, \mathbf{a})$.

THEOREM 6.1. *Let there be an ADF-graph H, a set of edges $\mathscr{C}$ in $E(H)$, and a spanning in-forest X of H such that X is disjoint from $\mathscr{C}$. For any vector of supplies $\boldsymbol{q}$ and capacities $\boldsymbol{a}$, $\mathscr{P}_1(H, \boldsymbol{q}, \mathscr{C}, \boldsymbol{a}) = \mathscr{P}_2(H, \boldsymbol{q}, \mathscr{C}, \boldsymbol{a}, X)$.*

A proof is provided in the supplementary material, section SM7. The supplementary material is available at asestes1.github.io/files/fgraph_siopt_supplement.pdf. While the polytopes $\mathscr{P}_1$ and $\mathscr{P}_2$ are exactly the same, rearranging the constraints in this way allows us to more easily identify a full-dimensional projection of $\mathscr{P}_1$ and $\mathscr{P}_2$. We define a new polytope on the space $\mathbb{R}^{|E(H)\setminus X|}$ by

$\mathscr{P}_3(H, \mathbf{q}, \mathscr{C}, \mathbf{a}, X)$:

$$0 \le \sum_{w:w\preceq^X v} \left(q_w + \sum_{e\in E^-(w;H)\setminus X} f_e - \sum_{e\in E^+(w;H)\setminus X} f_e \right) \quad \forall v \in V(H), \tag{6.4}$$

$$f_e \le a_e \quad \forall e \in \mathscr{C}, \tag{6.5}$$

$$f_e \ge 0 \quad \forall e \in E(H) \setminus X. \tag{6.6}$$

The variables in this polyhedron are the variables f_e for edges $e \in E(H) \setminus X$.

Remark 6.2. Let there be an ADF-graph H, a set of edges $\mathscr{C}$ in $E(H)$, and a spanning in-forest X of H such that X is disjoint from $\mathscr{C}$. Let $\mathbf{q}$ and $\mathbf{a}$ be integral vectors of supplies and capacities, respectively. Then $\mathscr{P}_3(H, \boldsymbol{q}, \mathscr{C}, \boldsymbol{a}, X)$ is the projection of the $\mathscr{P}_2(H, \boldsymbol{q}, \mathscr{C}, \boldsymbol{a}, X)$ that omits the variables f_e for each edge e in X.

Proof. Given a point λ in $\mathscr{P}_2(H, \boldsymbol{q}, \mathscr{C}, \boldsymbol{a}, X)$ or $\mathscr{P}_3(H, \boldsymbol{q}, \mathscr{C}, \boldsymbol{a}, X)$, let us use the convention that $f_e(\boldsymbol{\lambda})$ refers to the value that the variable f_e takes in the point $\boldsymbol{\lambda}$. We must show that for any point $\boldsymbol{\lambda}$ in $\mathscr{P}_3(H, \boldsymbol{q}, \mathscr{C}, \boldsymbol{a}, X)$, we can construct a feasible point $\boldsymbol{\phi}$ of $\mathscr{P}_2(H, \boldsymbol{q}, \mathscr{C}, \boldsymbol{a}, X)$ such that $f_e(\boldsymbol{\lambda}) = f_e(\boldsymbol{\phi})$ for every edge e in $E(H) \setminus X$. For each edge η in X, let $v(\eta)$ be the origin node of η. Then the lifted point $\boldsymbol{\phi}$ can be produced by letting

$$f_\eta(\boldsymbol{\phi}) = \sum_{w:w\preceq^X v(\eta)} \left(q_w + \sum_{e\in E^-(w;H)\setminus X} f_e(\boldsymbol{\lambda}) - \sum_{e\in E^+(w;H)\setminus X} f_e(\boldsymbol{\lambda}) \right)$$

for each edge η in X. It follows immediately from the definition that $\boldsymbol{\phi}$ satisfies constraint (6.1). Constraint (6.2) is satisfied for any capacitated edge e of $E(H) \setminus X$ because $f_e(\boldsymbol{\phi}) = f_e(\boldsymbol{\lambda})$ for any such edge. The set X is assumed to be uncapacitated, so constraint (6.2) does not apply to any edge of X. Similarly, since λ satisfies constraint (6.3) for any edge e in $E(H) \setminus X$ and $f_e(\boldsymbol{\phi}) = f_e(\boldsymbol{\lambda})$ for each such edge, $\boldsymbol{\phi}$ satisfies constraint (6.3). By definition $\boldsymbol{\lambda}$ must satisfy constraint (6.4), since it is an element of $\mathscr{P}_3$. This implies that $\boldsymbol{\phi}$ satisfies the constraints (6.3) corresponding to the edges of X as well. ∎

Under the nondegeneracy assumptions, we can show that the dimension of the convex hull of integral solutions for $\mathscr{P}_1$, $\mathscr{P}_2$ or $\mathscr{P}_3$ is equal to $|E \setminus X| = |E| \setminus |V|$. Thus, $\mathscr{P}_3$ has full dimension.

THEOREM 6.3. *Let there be an ADF-graph H, a set of edges $\mathscr{C}$ in $E(H)$, and a spanning in-forest X of H such that X is disjoint from $\mathscr{C}$. Let $\boldsymbol{q}$ and $\boldsymbol{a}$ be integral vectors of supplies and capacities, respectively. If the nondegeneracy assumptions (stated at the beginning of section 6) hold, then the convex hulls of integral solutions to $\mathscr{P}_1(H, \boldsymbol{q}, \mathscr{C}, \boldsymbol{a})$, $\mathscr{P}_2(H, \boldsymbol{q}, \mathscr{C}, \boldsymbol{a}, X)$, and $\mathscr{P}_3(H, \boldsymbol{q}, \mathscr{C}, \boldsymbol{a}, X)$ all have dimension $|E \setminus X|$.*

A proof is available in the supplementary materials, section SM8. The supplementary material is available at asestes1.github.io/files/fgraph_siopt_supplement.pdf.

6.2. Facet-defining equalities. As it turns out, both the formulation $\mathscr{P}_1$ and its projection $\mathscr{P}_3$ are quite strong. In fact, every inequality constraint either is a facet of the convex hull of integral solutions or is weakly dominated by other constraints in the formulation.

THEOREM 6.4. *Let there be an ADF-graph H, a vector of supply quantities $\boldsymbol{q}$, a set of capacitated edges $\mathscr{C}$, a vector of demand quantities $\boldsymbol{a}$, and a spanning in-forest X of $E(H)$ that contains no capacitated edges. Each inequality constraint of $\mathscr{P}_1(H, \boldsymbol{q}, \mathscr{C}, \boldsymbol{a})$, $\mathscr{P}_2(H, \boldsymbol{q}, \mathscr{C}, \boldsymbol{a}, X)$, and $\mathscr{P}_3(H, \boldsymbol{q}, \mathscr{C}, \boldsymbol{a}, X)$ is either facet-defining or redundant.*

Theorem SM9.1, Theorem SM9.2, and Theorem SM9.3 (located in the supplemental materials, section SM9) provide the specific conditions in which each constraint is facet-defining and the conditions in which each constraint is dominated. Proofs are also located in the supplemental materials.

7. Conclusion. We examined the polyhedral properties of minimum-cost flow problems on ADF-graphs in which every node has at least one uncapacitated outgoing edge. We discussed some integrality properties of these problems, and we showed that identifying an integral optimal solution is NP-hard. For each constraint, we provided conditions under which the constraint is a facet of the convex hull of integral solutions. When these conditions are not satisfied, we proved that the constraint is redundant and provided a set of dominating constraints. We provided a full-dimensional formulation in which these properties were also demonstrated. This work can be immediately applied to the stochastic, dynamic, network flow problems that arise in a variety of fields, ensuring that the formulations for these problems are strong.

There is still more work to be done to improve our understanding of the convex hull of integral solutions for this problem. This would involve identifying valid inequalities that separate fractional solutions. An ideal result would be to completely characterize the convex hull of integral solutions, although such a characterization seems unlikely, given that the problem is NP-hard. Improving our knowledge of these polyhedra could lead to better decomposition methods or other solution techniques for these types of problems. Similar work could also be done for other multistage stochastic problems, and in general more work could be done towards understanding how the structure of multistage stochastic problems relates to their single-stage counterparts.

REFERENCES

[1] J. E. ARONSON, *A survey of dynamic network flows*, Ann. Oper. Res., 20 (1989), pp. 1–66, https://doi.org/10.1007/BF02216922.

[2] M. O. BALL, R. HOFFMAN, A. R. ODONI, AND R. RIFKIN, *A stochastic integer program with dual network structure and its application to the ground-holding problem*, Oper. Res., 51 (2003), pp. 167–171, https://doi.org/10.1287/opre.51.1.167.12795.

[3] D. Bertsimas and J. N. Tsitsiklis, *Introduction to Linear Optimization*, Athena Scientific, Charlestown, MA, 1997.
[4] J. R. Birge and F. Louveaux, *Introduction to Stochastic Programming*, Springer, 1997, Berlin, https://doi.org/10.1007/978-1-4614-0237-4.
[5] R. Cambini, G. Gallo, and M. G. Scutellà, *Flows on hypergraphs*, Math. Program., 78 (1997), pp. 195–217, https://doi.org/10.1007/BF02614371.
[6] R. K. Cheung and W. B. Powell, *An algorithm for multistage dynamic networks with random arc capacities, with an application to dynamic fleet management*, Oper. Res., 44 (1996), pp. 951–963, https://doi.org/10.1287/opre.44.6.951.
[7] J. Edmonds, *Optimum branchings*, J. Res. Natl. Bureau Standards B Math. Math. Phys., 71B (1967), pp. 233–240.
[8] F. G. Engineer, G. L. Nemhauser, and M. W. P. Savelsbergh, *Dynamic programming-based column generation on time-expanded networks: Application to the dial-a-flight problem*, INFORMS J. Comput., 23 (2011), pp. 105–119, https://doi.org/10.1287/ijoc.1100.0384.
[9] A. S. Estes and M. O. Ball, *Equity and strength in stochastic integer programming models for the dynamic single aiport ground-holding problem*, Transp. Sci., 54 (2020), pp. 855–1152.
[10] M. Fonoberova, *Algorithms for finding optimal flows in dynamic networks*, in Handbook of Power Systems II, S. Rebennack, P. M. Pardalos, M. V. F. Pereira, and N. A. Iliadis, eds., Springer, Berlin, 2010, pp. 31–54, https://doi.org/10.1007/978-3-642-12686-4_2.
[11] L. F. Frantzeskakis and W. B. Powell, *A successive linear approximation procedure for stochastic, dynamic vehicle allocation problems*, Transp. Sci., 24 (1990), pp. 40–57, https://doi.org/10.1287/trsc.24.1.40.
[12] D. R. Fulkerson, *Packing rooted directed cuts in a weighted directed graph*, Math. Program., 6 (1974), pp. 1–13, https://doi.org/10.1007/BF01580218.
[13] G. Gallo, G. Longo, S. Pallottino, and S. Nguyen, *Directed hypergraphs and applications*, Discrete Appl. Math., 42 (1993), pp. 177–201, https://doi.org/10.1016/0166-218X(93)90045-P.
[14] G. D. Glockner and G. L. Nemhauser, *A dynamic network flow problem with uncertain arc capacities: Formulation and problem structure*, Oper. Res., 48 (2000), pp. 233–242, https://doi.org/10.1287/opre.48.2.233.12384.
[15] R. G. Jeroslow, K. Martin, R. L. Rardin, and J. Wang, *Gainfree Leontief substitution flow problems*, Math. Program., 57 (1992), pp. 375–414, https://doi.org/10.1007/BF01581090.
[16] R. G. Jeroslow and J. Wang, *Dynamic programming, integral polyhedra and Horn clause knowledge base*, ORSA J. Comput., 1 (1989), pp. 7–19, https://doi.org/10.1287/ijoc.1.1.7.
[17] R. M. Karp, *Reducibility among combinatorial problems*, in Complexity of Computer Computations, R. E. Miller, J. W. Thatcher, and J. D. Bohlinger, eds., Springer, Boston, MA, 1972, pp. 85–103, https://doi.org/10.1007/978-1-4684-2001-2_9.
[18] B. Kotnyek, *An Annotated Overview of Dynamic Network Flows*, Tech. Report RR-4936, INRIA, 2003, https://hal.inria.fr/inria-00071643.
[19] R. K. Martin, R. L. Rardin, and B. A. Campbell, *Polyhedral characterization of discrete dynamic programming*, Oper. Res., 38 (1990), pp. 127–138, https://doi.org/10.1287/opre.38.1.127.
[20] J. M. Mulvey and H. Vladimirou, *Solving multistage stochastic networks: An application of scenario aggregation*, Networks, 21 (1991), pp. 619–643, https://doi.org/10.1002/net.3230210603.
[21] W. B. Powell and L. F. Frantzeskakis, *Restricted recourse strategies for dynamic networks with random arc capacities*, Transp. Sci., 28 (1994), pp. 3–23, https://doi.org/10.1287/trsc.28.1.3.
[22] M. Skutella, *An introduction to network flows over time*, in Research Trends in Combinatorial Optimization: Bonn 2008, W. Cook, L. Lovász, and J. Vygen, eds., Springer, Berlin, 2009, pp. 451–482, https://doi.org/10.1007/978-3-540-76796-1_21.
[23] H. Song, R. K. Cheung, and H. Wang, *An arc-exchange decomposition method for multistage dynamic networks with random arc capacities*, European J. Oper. Res., 233 (2014), pp. 474–487, https://doi.org/10.1016/j.ejor.2013.09.048.
[24] A. F. Veinott, *Extreme points of leontief substitution systems*, Linear Algebra Appl., 1 (1968), pp. 181–194, https://doi.org/10.1016/0024-3795(68)90002-5.
[25] J. Verstraëte, *Extremal problems for cycles in graphs*, in Recent Trends in Combinatorics, A. Beveridge, J. R. Griggs, L. Hogben, G. Musiker, and P. Tetali, eds., Springer, Cham, 2016, p. 83–116, https://doi.org/10.1007/978-3-319-24298-9_4.