ORIGINAL ARTICLE

Multi-Stage Network Interdiction with Decision-Dependent Success: Scenario Clustering and Reformulation Techniques

Baris Tezcan[1*] | **Kayse Lee Maass**[2*]

[1]Mechanical and Industrial Engineering, Northeastern University, Boston, MA, 02115, USA

Correspondence
Baris Tezcan, Mechanical and Industrial Engineering, Northeastern University, Boston, MA, 02115, USA
Email: tezcan.b@northeastern.edu

Funding information
National Science Foundation: CMMI - 1841893

This article studies a network interdiction problem where the success of the first player's attempt to remove arcs is based on decision- dependent success probabilities, and the second player (the follower) responds by pushing maximum flow. In this multi-stage model, previous interdictions affect future interdictions' success probabilities. The decision-dependent structure requires an objective function that includes a conditional probability that is nonlinear, which we reformulate into a mixed integer program using probability flow networks over all possible scenarios. However, this reformulation involves a large number of scenarios. To reduce the scenario space, we present a scenario clustering approach that exploits the problem structure while preserving optimality. We also provide a scenario clustering heuristic for further reductions and faster solution times where optimality is not guaranteed. We illustrate our method by solving instances on various physical grid and supply chain networks. We also investigate how changing decision-dependent success probabilities over time can influence interdiction policies over these networks.

KEYWORDS
Network Interdiction, Stochastic Optimization, Decision-dependent Uncertainty, Scenario Clustering

[*]Equally contributing authors.

1 | INTRODUCTION

Network interdiction models are two-player games where the first decision-maker, the attacker (or the leader), makes the initial decision to change some network properties, whereas the other decision-maker, the defender (or the follower), pushes some network flow after the attacker's decision. They are useful for modeling decisions in a wide variety of application areas, such as allocating detectors for maximizing the detection of nuclear smuggling, border patrol, and dismantling drug trafficking networks [12, 11, 18]. In recent years, researchers have continued to adapt network interdiction models to better reflect the nuances of decision-making in these varying environments. However, with model adaptions often comes added complexity. Multi-stage and decision dependencies in network interdiction models are two ways to help models better reflect reality. Throughout this chapter, we will use the term 'stage' to refer to the time period where the interdictor attempts an interdiction and the defender responds to it by sending a flow through the network. Thus, multistage models capture the reality that decisions have impacts beyond their decision epochs and are relevant to planning interdiction policies that span multiple coordinated interdiction decisions. Decision-dependent models capture the reality that decisions a player makes influence the scenario probabilities of the scenario distribution. This is different than just considering an external distribution in decision-making; in decision-dependent models, decisions also shape the distribution.

To the best of our knowledge, [16] is the only paper that incorporates both multi-stage and decision-dependent success probabilities into network interdiction models. In [16], decision-dependency lies in interdiction success over multiple stages such that previous interdiction attempts affect the success probability of success of future interdictions, causing non-linearities. As [16] shows, network interdiction models with multi-stage and decision-dependent success probabilities are exceedingly difficult to solve. Hence, the authors used a genetic algorithm with repair operations to get solutions for two-stage, 3-by-3, and 5-by-5 grid networks. Yet genetic algorithms are sensitive to input data and their own parameters and can't provide a solution performance guarantee. As such, in this paper, we present a reformulation technique to linearize the model and employ exact and heuristic scenario clustering techniques that exploit the structure of the graph and flow to solve multi-stage decision-dependent network interdiction models more efficiently.

Specifically, we focus on network interdiction models where the defender's problem is a maximum-flow problem, the interdictor attempts to remove arcs over multiple stages, and arc interdiction success is stochastic. The interdiction success probability depends on the decisions (interdiction attempts) made during the previous stages. Consider the case of an attacker who is planning to disrupt an ongoing flow over a network, such as a power grid or an illicit flow, which are common application areas of network interdiction models [4, 9, 11, 18]. We first assume that the attacker plans interdictions for all of the stages (i.e., time periods) at the beginning. This is to consider interdiction attempts that are at the strategic level and require planning and preparation ahead and can not be changed easily on the fly.

The second important assumption is that the interdictor knows the interdiction success probabilities. For each arc, the interdictor knows the following information: if they attempt to interdict arc α for the first time, the interdiction success probability will be p_α^1; the second time they attempt to interdict the same arc, the interdiction probability will be p_α^2 and so on. [16] provides more details on modeling these probabilities to capture how different interdiction success beliefs can suggest varying interdiction policies in the case of human trafficking. In this paper, we refrain from that discussion and focus on the solution methodology, assuming the probabilities are given. While both assumptions have their limitations in capturing all interdiction types and information/belief updates, the model we are analyzing has its own use in strategic interdiction planning and can be used as a sub-problem for versions with more relaxed assumptions, such as problems with sequential decision-making.

In stochastic optimization literature broadly, scenario clustering is an approach to generate clusters of scenarios,

which are representative of the entire scenario space that an optimization problem needs to consider [7]. Our exact approach exploits the graph and flow nature of our problem. In our problem, a scenario represents a realization of the interdiction attempted graph, and the realization of a scenario depends on i) interdiction decisions and ii) interdiction success probabilities. We use a binary scenario tree to generate possible scenarios by considering if an arc is removed or not at each child. We then cluster children scenarios in a scenario tree whenever a parent scenario gets fully disconnected and use the parent as the representative scenario for the cluster. Our heuristic approach is more general and uses decision success probabilities to evaluate the maximum occurrence probability of a scenario in the scenario tree and uses user-defined cutoffs to cluster children scenarios once the maximum occurrence probability goes beyond the cutoff value. This approach does not necessarily guarantee optimality but gives the user the option to decide on the run time vs optimality gap trade-off. We use both the reformulation and the scenario reduction algorithms to solve larger decision-dependent network interdiction problems (either in terms of network size and/or the number of stages) than previous research has. Although our focus is on the situation in which the follower/inner problem seeks to maximize flow, our approach is expected to be useful for other decision-dependent problems in the network interdiction settings where the follower/inner problem is the shortest path, critical path, or some other simple network flow problem that can be solved easily once the interdiction decision is set.

The rest of this paper is as follows; in Section 2, we provide a literature review about relevant network interdiction models and common scenario clustering ideas. Section 3 involves two mixed integer program formulations for the multi-stage decision-dependent network interdiction problem, and in Section 4, we introduce our exact and heuristic scenario clustering techniques. In Section 5, we present the results of our computational experiments on grid and multi-echelon supply chain networks. Finally, in Section 6, we summarize our key results, and limitations and provide future research directions.

2 | LITERATURE REVIEW

In this section, we provide a short literature review on network interdiction models with a focus on multi-stage models and models with decision-dependent uncertainty. For a more comprehensive review, please refer to [15]. The base model we consider in this paper is a min-max flow network interdiction problem in which interdictors make decisions in an attempt to minimize the maximum flow the defender can put through the network. [17] introduces the single-stage (in the sense that there is a single interdiction period) deterministic max-flow network interdiction problem to analyze disrupting an adversary's infrastructure and provides a dual-and-combine framework to have a tractable version of the bi-level problem. The stochastic and single-stage version of the model was introduced by [5], who considered the case where arc interdictions can be successful (i.e., reduce arc capacity to zero) with a given probability. [8] proposes a sample average approximation for the stochastic max-flow network interdiction problem to handle a large number of scenarios. Even though both these works are about the single-stage problem, they show the complexity of the increasing number of scenarios and the need for designing methods to handle them.

***Decision-Dependent Uncertainty*:** In traditional stochastic network interdiction problems, the model stochasticity does not change the underlying network structure, which allows each resulting scenario to have an associated deterministic scenario probability, which is used as input into the problem; for example, stochastic arc capacities do not affect the network structure and enable the associated stochastic network interdiction model to have a set of preidentified scenarios with known scenario probabilities. However, when stochastic success probabilities are considered, calculating the associated scenario probabilities becomes challenging because they depend on the decisions. This is why the term decision-dependent uncertainty is used to describe this type of model feature.

To understand this complexity better, consider a network with a set A of interdictable arcs. Let x_α denote whether an arc interdiction attempt is made on an arc $\alpha \in A$ and p_α denote the success probability of the interdiction. That is, if $x_\alpha = 1$, the arc α will be successfully interdicted and removed with probability p_α; if $x_\alpha = 0$, the arc will not be removed (as no attempt has been made). We define a scenario ω as the realization of which arcs exist in the network after an interdiction decision has been made (i.e., the success or non-success of interdictions has been realized) and Ω be the set of all scenarios. Specifically, let $\mathbf{I}^\omega = \{I_1^\omega, I_2^\omega, \ldots, I_{|A|}^\omega\}$ represent which arcs exist ($I_\alpha^\omega = 0$) and which arcs are successfully removed ($I_\alpha^\omega = 1$) in scenario ω after an interdiction attempt.

Assuming that the arc interdiction probabilities are independent, we can calculate the decision-dependent probability that scenario ω occurs given the interdiction plan attempt $\mathbf{x} : \{x_1, x_2, \ldots, x_{|A|}\}$ as follows:

$$\mathbf{P}_\mathbf{x}\{\mathbf{I}^\omega\} = \prod_{\alpha \in A} P_{x_\alpha}\{I_\alpha^\omega\}$$

where $P_{x_\alpha}\{I_\alpha^\omega\}$ is arc α's realization probability:

$$P_{x_\alpha}\{I_\alpha^\omega\} = \begin{cases} x_\alpha p_\alpha & I_\alpha^\omega = 1 \\ 1 - x_\alpha p_\alpha & I_\alpha^\omega = 0 \end{cases}$$

For example, consider a small network with three arcs. Suppose arcs 1 and 3 are attempted to be interdicted (i.e., $\mathbf{x} : \{x_1 = 1, x_2 = 0, x_3 = 1\}$). One potential resulting scenario, ω' is that arc 1 is successfully interdicted, arc 2 is unchanged, and arc 3 is unsuccessfully interdicted (and therefore remains present in the resulting network). This situation corresponds to the scenario realization $\mathbf{I}^{\omega'} = \{I_1^{\omega'} = 1, I_2^{\omega'} = 0, I_3^{\omega'} = 0\}$ which occurs with probability:

$$\begin{aligned} \mathbf{P}_\mathbf{x}\{\mathbf{I}^{\omega'}\} &= P_{x_1}\{I_1^{\omega'} = 1\} P_{x_2}\{I_2^{\omega'} = 0\} P_{x_3}\{I_3^{\omega'} = 0\} \\ &= (x_1 p_1)(1 - x_2 p_2)(1 - x_3 p_3) \\ &= (1 \cdot p_1)(1 - 0 \cdot p_2)(1 - 1 \cdot p_3) \\ &= p_1(1 - p_3) \end{aligned}$$

Note that the scenario probability function $\mathbf{P}_\mathbf{x}\{\mathbf{I}^\omega\}$ is nonlinear because it consists of conditional probabilities as a function of interdictions. [14] reformulates the nonlinear term by using [10]'s distribution shaping and then uses Benders Decomposition. [13] uses probability flows, the concept of considering conditional probability as network flow, to model and optimize over decision-dependent scenario probabilities in problems such as minimizing the maximum reliability path introduced by [12]. Both models are for a single interdiction stage and, therefore, do not consider how interdiction decisions can change interdiction success probabilities over time in a multi-stage setting.

Multi-stage Network Interdiction Models: To our knowledge, there are no similar multi-stage network interdiction models to this paper, yet the concept of multiple stages is studied in network interdiction settings with different dynamics. For example, [11] analyzes a hierarchical drug trafficking network to decide which arcs should be interdicted when some types of interdictions can only occur after other types of interdictions. [6] studies sequential interdictions where the network topology is known probabilistically. These models illustrate the importance of considering multiple stages in network interdiction models, yet their considerations are drastically different than those of this paper.

Multi-stage Network Interdiction Models with Decision-Dependent Uncertainty: [16] introduces the multi-stage decision-dependent network interdiction model where each stage's interdiction success probability depends on the previous stage's interdiction attempts. The multiple stages magnify the complexities associated with the decision-dependent (and nonlinear) scenario probability calculations. In this paper, we introduce exact and heuristic scenario clustering techniques to solve larger instances of this model more efficiently.

Scenario Clustering: Scenario clustering is a way to reduce the exhaustive search space for a stochastic optimization program by clustering scenarios together. [7] proposes a scheme to cluster scenarios based on minimizing opportunity cost distances between scenarios (or a cluster's diameter). They define the opportunity cost between two scenarios as the objective function difference between optimally responding to a scenario while another scenario happens and propose clustering algorithms to group scenarios where the impact of optimal decisions over different scenarios is small. Our approach is more akin to the scenario bundling idea proposed in [10]. Their problem is concerned with whether to invest in a network's arcs to strengthen them against a disaster so that the paths have high reliability. This problem is similar to ours as the scenario space and probabilities are determined by decisions rather than an external scenario distribution. They dynamically enumerate scenarios in a binary scenario tree and stop branching at a node if they determine all the children's scenarios of the node have the same optimal solution once the scenarios are realized. This can be done as scenarios are neighbors of each other in the sense that one can go between scenario realizations by adding or removing an arc from the original network. Another benefit of this approach is to generate the scenario space dynamically, rather than creating the whole scenario space first and clustering it.

Our contribution in this paper is firstly reformulating the multi-stage decision-dependent network interdiction problem as a mixed integer program on a scenario tree by using probability flows. We also propose exact and heuristic scenario clustering algorithms to exploit the network structure of the problem to reduce the size of the scenario space that the proposed mixed-integer program needs to evaluate. Our clustering algorithms are unique in the sense that it exploits the relationship between the scenarios (subgraphs) dynamically, rather than clustering a set of scenarios generated by an external distribution.

3 | ARC AND PATH-BASED MIXED INTEGER LINEAR PROGRAMS

Single-stage bi-level models can be linearized by enumerating all possible inner optimization problems and storing them in memory if the outer problem has integer variables. Then, these pre-solved optimal values can be used as objective coefficients in zero-sum objectives. This enumeration quickly becomes intractable as the size of the inner-level problem increases. In response, a large body of literature focuses on general and problem-specific solution methodologies to bi-level problems [3]. For example, the single-stage deterministic network interdiction introduced in [17] uses a dualize and combine approach to linearize the nonlinearity (linear variable multiplied by binary variable) in their objective. However, complexities emanating from decision-dependent (and) nonlinear scenario probability calculations over multiple stages require new solution methods not yet present in the literature.

In this section, we introduce two mixed-integer problems that use probability flows over a binary scenario tree to find interdiction attempts that give the minimum-maximum flow. We first introduce the traditional single-stage versions of the models to illustrate how probability flows can be used to calculate decision-dependent scenario probabilities. Then, we expand these formulations to multiple stages, which is our main focus.

3.1 | Single Stage Decision-Dependent Network Interdiction Problem

Our model considers an interdictor attempting to remove arcs from a network while staying within a certain budget to minimize the defender's/follower's maximum flow. We assume that each arc has the same interdiction attempt cost. Throughout the manuscript, we use the terms "interdicted" and "interdiction decision" to refer to the attacker's decision to *attempt* to interdict an arc. However, attempted interdictions have stochastic realizations; they may or may not be successful. Hence, we refer to the resulting effect on the network as the interdictions' "realization" or the "realized network". In the single-stage version, an interdiction attempt does not change future interdiction success probabilities as there is only one decision epoch for the attacker. The only decision-dependency for these models is having the scenario realization probabilities as variables. Multi-stage versions of these models are introduced in Section 3.2 as extensions of the models in this section.

3.1.1 | Arc-Based Formulation

Let $G = (N, A)$ be a directed network where N is the set of nodes and A is the set of arcs. Suppose that each arc in A can be interdicted and that a successful interdiction results in complete removal of the arc. In that case, a decision-dependent network interdiction problem will have $2^{|A|}$ scenarios for a stage (although some of these scenarios can be ignored if they exceed the budget). Also, each scenario corresponds to a subgraph of G, as arc interdictions remove arcs from the graph once they are realized. Therefore, enumerating maximum-flows of every single subgraph's maximum-flow provides a (brute force) way to linearize the bi-level nature of the problem so we can focus on the nonlinear scenario probability calculation. The decision-dependent network interdiction problem can be thought of as controlling the scenario probability distribution, as our reformulation allows us to model the effects of interdiction attempts on scenario probabilities, given the scenario maximum flows.

Given the corresponding scenario tree, we aim to model the interdiction effect on the scenario probabilities. Our scenario probabilities consist of conditional probabilities as a function of interdictions. Therefore, they are nonlinear. [13] introduces using probability chains/flows to linearize such calculations by using network flows.

To do this calculation, let's also define the binary scenario tree G' of graph G where each node represents a subgraph of G. To begin, let the root node of the scenario tree represent the full graph G. Then, we can order each arc in A arbitrarily and let each layer of the scenario tree correspond to an arc. Each node of the scenario tree (except the leaf nodes) will have two children. Each child either contains the arc that their layer corresponds to or not. For ease of discussion and without loss of generality, we assume that the left children will be subgraphs of their parent node's graph with the corresponding layer's arc removed. The right children are exactly the same as their parent node. Continuing in this fashion, we will have $2^{|A|}$ leaf nodes, which each represent a subgraph of the original graph G. For each node, we can calculate the maximum flow (for example, with an augmenting path algorithm or others; see [1] for the maximum-flow problem and efficient algorithms).

We illustrate the above logic with an example in Figure:1. On the left, we see the original graph G with each arc's flow capacity u and interdiction success probability p. The leaf nodes in the corresponding scenario tree G' (right image) highlight the 8 subgraphs of G, which each represent a scenario.

Now, on this scenario tree, we can use the aforementioned probability flows. We want the model to have the following logic given an interdiction. The interdiction variable, x, for an arc α can be defined as follows.

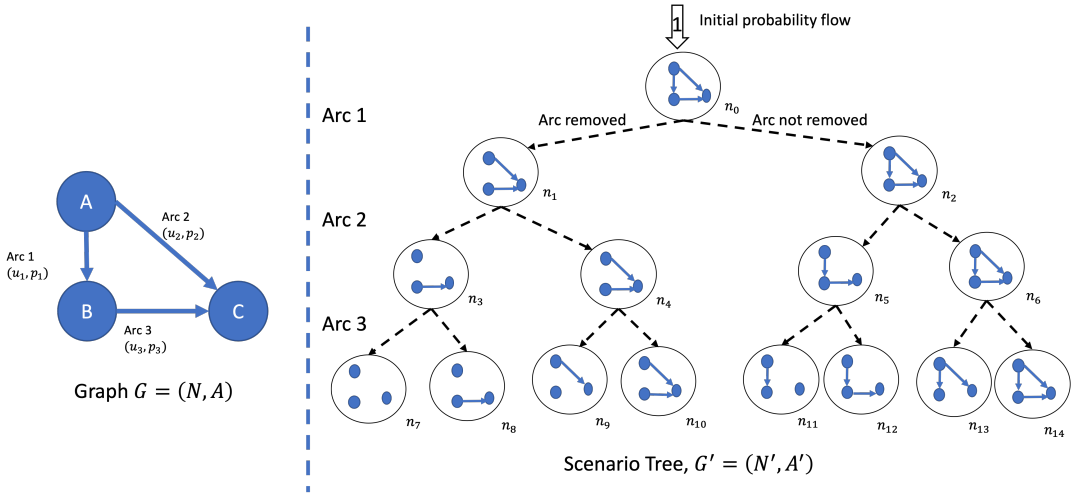


FIGURE 1 Graph G (left) and Scenario Tree G' (right)

$$x_\alpha = \begin{cases} 1 & \text{if arc } \alpha \text{ is interdicted,} \\ 0 & \text{otherwise.} \end{cases}$$

As a reminder, each layer of the binary scenario tree G' corresponds to an arc in the original graph G. If an arc α on the original graph G is interdicted ($x_\alpha = 1$), we split the flow coming into each node n in the α layer of G'. To do this, let i_n be the flow coming into node n. Then, we split the incoming flow i_n into two outgoing flows by multiplying i_n by the arc interdiction success probability (p_α) for the outgoing flow on the left side and with the failure probability $(1 - p_\alpha)$ for the outgoing flow on the right side.

Continuing with the same example graph given in Figure 1, we present an example flow on the scenario tree G' given an interdiction to arcs 1 and 3 of G in Figure 2. The root node, n_0 has the the incoming (probability) flow of 1 and nothing has been realized at the root node yet. As we go down the tree, we observe realizations at each layer. In layer 1 (nodes n_1 and n_2), we want to have the probability flow split as the corresponding arc, arc 1, has been interdicted. Therefore, p_1 will flow to the left and $(1 - p_1)$ will flow to the right. Since there was no interdiction on arc 2, layer 2 does not see any split in flow; all of the flows go to the right children. We observe the last set of probability flow splits at layer 3 to represent the effect of arc 3's interdiction. The probability flow reaching each of the leaf nodes represents the probability that the interdiction attempts will result in the subgraph of G associated with the leaf node. For example, the incoming flow for leaf node n_9 is $p_1 * p_3$ as this node denotes the scenario where both arcs 1 and 3 are successfully interdicted. Leaf node n_{10}'s probability is $p_1 * (1 - p_3)$ and represents the scenario where arc 1 is successfully interdicted but the interdiction on arc 3 is unsuccessful. Probability splits can be done similarly for n_{13} and n_{14}.

We use a mixed- integer program to ensure that the probability flow within the scenario tree behaves as aforementioned. Before introducing the model, we formalize the following notation. Let the variable i_n denote the incoming flow for a node $n \in G'$ and parameter c_n denote the maximum flow value for a node $n \in G'$. i_{n_L} and i_{n_R} are probability flow variables that record the left-hand and right-hand side outgoing flows for node n, respectively as n_L and n_R are

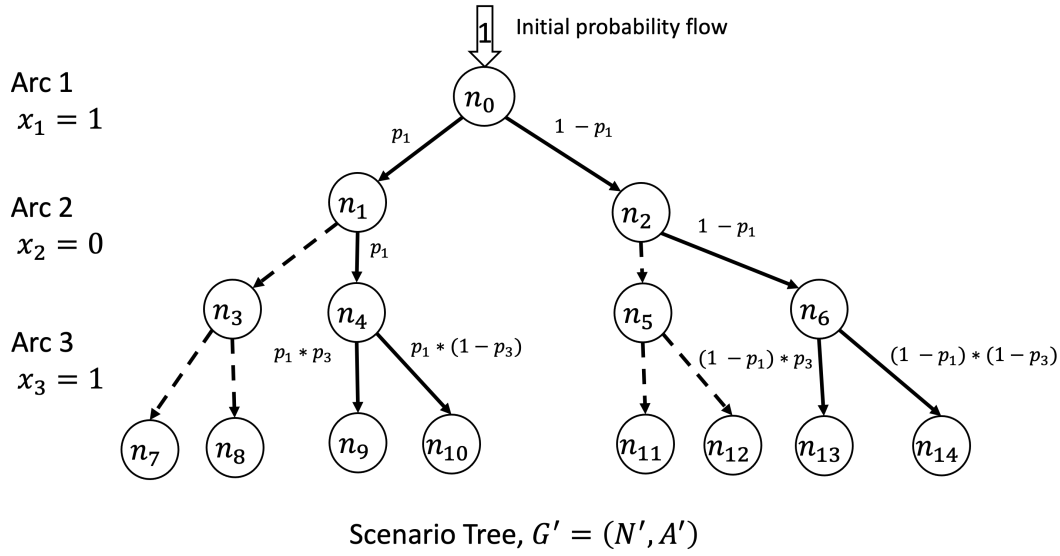


FIGURE 2 Scenario Tree G' and probability flows given an interdiction attempt $\mathbf{x} = (1, 0, 1)$. Solid lines represent arcs in the scenario tree that have probability flow; dashed lines have no flow. Solid arcs also have their associated amount of probability flow listed.

left and right children. We also use a depth function $D : n \mapsto \alpha$, to map scenario tree G' nodes n to their corresponding graph G arc α. For example, in Figure 2, the root node will map to the depth zero, and the first two children, n_1 and n_2, will have $D(n_1) = D(n_2) = 1$ and so on. As previously mentioned, x_α is the binary arc interdiction variable for arc α in graph G. Additionally, since each leaf node in G' represents a realization scenario corresponding to a subgraph of G, each leaf node of G' corresponds to a scenario ω representing a subgraph of G. Thus, the set of all scenarios Ω also represents the set of all leaf nodes of G' and, equivalently, the set of all subgraphs of G. Let's also denote the set of leaf nodes in G' with N'_{leaf}. Finally, R is the number of arcs that can be interdicted.

$$\underset{\mathbf{i,l,r,x}}{\text{minimize}} \sum_{n \in N'_{leaf}} i_n * c_n = \sum_{\omega \in \Omega} i_\omega * c_\omega \tag{1}$$

subject to

$$i_{n_0} = 1 \tag{2}$$

$$i_n = i_{n_L} + i_{n_R} \qquad \forall n \in N' \setminus N'_{leaf} \tag{3}$$

$$i_{n_L} \leq x_{D(n)+1} \qquad \forall n \in N' \setminus N'_{leaf} \tag{4}$$

$$i_{n_L} \leq p_{D(n)+1} * i_n \qquad \forall n \in N' \setminus N'_{leaf} \tag{5}$$

$$\sum_{\alpha \in A} x_\alpha \leq R \tag{6}$$

$$0 \leq i_n, l_n, r_n, \leq 1 \qquad \forall n \in N' \setminus N'_{leaf} \tag{7}$$

$$x_\alpha \in \{0, 1\} \qquad \forall \alpha \in A \tag{8}$$

The objective (1) is a weighted average of the flows incoming to leaf nodes, where the weights represent the maximum flows of the leaf node's graph (which are the original graph's corresponding subgraphs). For each node in the tree, we have flow balance constraints (2) and (3) and a set of upper bound constraints (4) and (5) for the left children nodes. The first set of upper bound constraints, (4), ensures that if there is no interdiction, no flow goes through to the left child. The second set of upper bound constraints (5) split the flow into $i_n * p_\alpha$ and $i_n * (1 - p_\alpha)$ between the two children nodes if there is an interdiction. Note that at the bottom layer, the subgraphs of G corresponding to the right children will always have a maximum-flow ($c_n = c_\omega$) that is at least as much as the subgraphs G corresponding to the left children, because the right child is a supergraph of the left child (i.e. right child has at least all the arcs that the left child has). As we have a minimization problem, the left children's flows will always be at their upper bound because of the flow preservation throughout the tree and the max flow of the left child always being less or equal to the right child's. Finally, (6) is the budget constraint, and (7) and (8) are domain constraints.

3.1.2 | Path-based Formulation

In this section, we introduce an equivalent formulation to the arc-based formulation given in 3.1.1. While this formulation requires more pre-processing on the binary scenario tree, it contains fewer variables because it denotes each root-leaf pair's path flow with a variable.

Let us denote the indices of the set of paths from the root node to each leaf node in scenario tree G' by the set $\mathcal{P}$ and denote the paths in $\mathcal{P}$ that contain node n by set $\mathcal{P}_n$. Then, we can consider the probability flow, f_p, on each path $p \in \mathcal{P}$ as a decision variable. We define the function $parent(\cdot)$ to take a node n as input and provide the parent node for node n as the output. Additionally, let, N'_{left} denote the left children nodes.

With this notation, we introduce the path-based formulation for the single-stage decision-dependent network interdiction problem.

$$\underset{\mathbf{f},\mathbf{x}}{\text{minimize}} \sum_{p \in \mathcal{P}} c_p * f_p \tag{9}$$

subject to

$$\sum_{\alpha \in A} x_\alpha \leq R \tag{10}$$

$$\sum_{p \in \mathcal{P}} f_p = 1 \tag{11}$$

$$\sum_{p \in \mathcal{P}_n} f_p \leq x_{D(n)} \qquad \forall n \in N'_{left} \tag{12}$$

$$\sum_{p \in \mathcal{P}_n} f_p \leq p_{D(n)} * \sum_{p \in \mathcal{P}_{parent(n)}} f_p \qquad \forall n \in N'_{left} \tag{13}$$

$$x_\alpha \in \{0, 1\} \qquad \forall \alpha \in A \tag{14}$$

$$0 \leq f_p \leq 1 \qquad \forall p \in \mathcal{P} \tag{15}$$

Objective (9) is analogous to (1) and calculates the expected maximum flow after interdiction(s) by considering the probability flow that goes into each scenario (leaf node). (10) is the budget constraint, and (14) and (15) are domain constraints. (11) requires that the sum of all path flows is equal to one, ensuring we have a probability space. At each left children node ($n \in N_{left}$), we look at the summation of all passing flows ($\mathcal{P}_n$) and, if there is an interdiction, split

this value using constraints (12) and (13). At the leaf level, there will only be the leaf node's (scenario's) probability flow, ensuring that the correct scenario probability is calculated for each leaf node. We continue with the example in Figure 1 to demonstrate the logic of bounding the summation of scenario probabilities at each of the left children nodes in Figure 3.

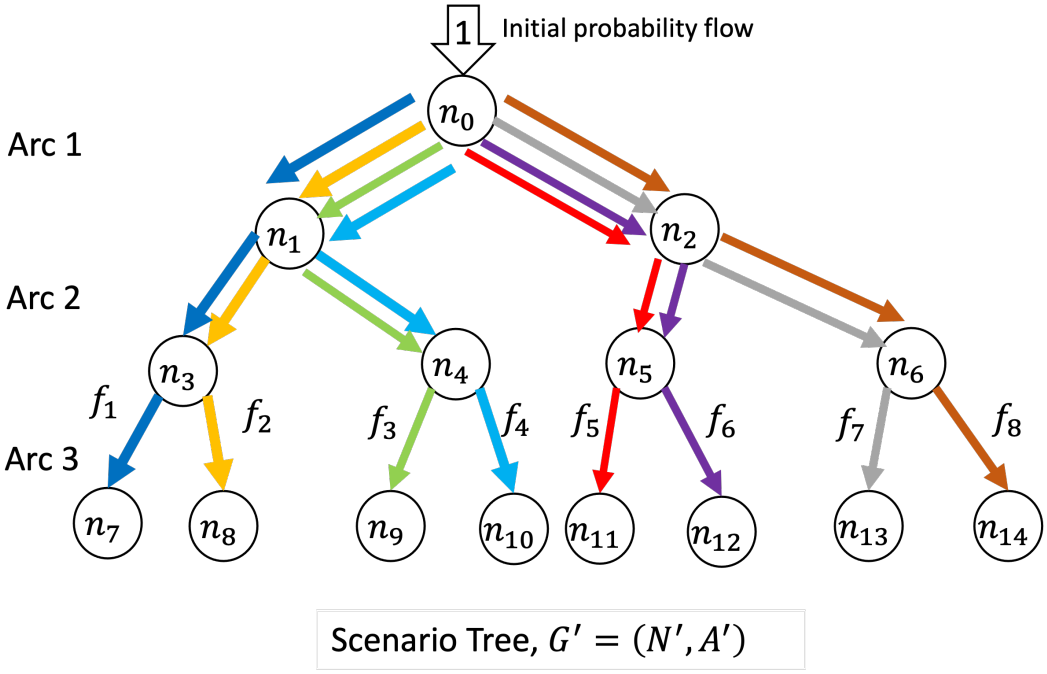


FIGURE 3 Path Variables on Scenario Tree G'

Paths that go through the first left child node, n_1 are f_1, f_2, f_3 and f_4, so $\mathcal{P}_{n_1} = \{1, 2, 3, 4\}$. Therefore, if arc 1 is not attempted to be interdicted ($x_{D(n_1)} = 0$), no flow should go through any of the root-leaf paths that go through n_1. This case activates the corresponding constraint in (12) as the summation of flows going through node n_1, $(\sum_{p \in \mathcal{P}_{n_1}} f_p)$, will be set to zero as $x_{D(n_1)}$, is zero in this case. If there is an interdiction (e.g., $x_{D(n_1)} = 1$), a split should consider the incoming amount of flow into the node (e.g., into n_0)and split it accordingly. In this case, the corresponding constraint in (13) gets activated as the right-hand side of it will be at most one (as it is a multiplication of two real numbers from zero and one). This creates a tighter upper bound for the summation of the flows and ensures the split is done. We continue with this probability split by activating the correct corresponding constraint logic for the multi-stage problems in Section 3.2.

3.2 | Linearization of the Multi-Stage Decision-Dependent Network Interdiction Problems

This section introduces the multi-stage extensions of the arc-based and path-based formulations in sections 3.1.1 and 3.1.2. A key assumption of the multi-stage decision-dependent version of the problem is that deciding to interdict

an arc in one time period impacts the probability of successful interdiction in future periods.[1] Thus, as previously described, this results in the model decisions affecting the scenario probabilities, resulting in nonlinearities in the expected min-max flow objective. This will be the case where we believe repeated attempts would have different success probabilities depending on the number of previous attempts. Let's consider a case where we need to plan interdictions on an arc α where the interdiction success probability decreases in future stages with each interdiction. For example, suppose the success probabilities for an arc over 3 stages is $\boldsymbol{p_\alpha} = \{0.7, 0.6, 0.2\}$. This means the first interdiction on this arc, whether at $t = 0$, $t = 1$, or $t = 2$, will have a success probability of 0.7 (i.e., the first element of the vector $\boldsymbol{p_\alpha}$). If a second interdiction is attempted at on arc α some time in the future it will have a success probability of 0.6. Note that interdictions are not required to happen in successive time periods; the first interdiction could happen at time $t = 0$ and be successful with probability 0.7, and the second interdiction could happen at time $t = 2$ and be successful with probability 0.6.

We use the index k to count the interdictions done on an arc until stage t and use the convention that the initial time period is $t = 0$. We update the binary interdiction variable with the time index, so x_α^t denotes whether an arc α is attempted to be interdicted at time t. To get the success probability for an arc at time t, (for stages after the initial time period, i.e., $t > 0$), we count the number of interdiction attempts on arc α until time t using $k = \sum_{t'=0}^{t-1} x_\alpha^{t'}$ and then use the corresponding success probability p_α^k to calculate the scenario probabilities and the expected maximum-flow values.

We define an auxiliary interdiction counter variable q and a counting network C_α to count how many interdictions are attempted on an arc at a given time. Each arc α has its own directed counting network $C_\alpha : (N_\alpha, A_\alpha)$. This network has nodes to denote each time-interdiction count pair, (t, k). We also connect each of these nodes with their neighboring pairs; that is, (t, k) is connected with $(t + 1, k)$ and $(t + 1, k + 1)$. In this network, $q_{\alpha,(t,k),(t',k')}$, denotes the counting flow between nodes, (t, k) and (t', k'). For a given arc α, we start with node (0,0) to denote that at time $t = 0$, $k = 0$ interdictions have been taken on arc α. If there is an interdiction attempt at $t = 0$ ($x_\alpha^{t=0} = 1$), the counter should increase, and k will be $k = 1$, so we go to node $(1, 1)$. In the other case, we go to $(1, 0)$ as k remains $k = 0$. In other words, the timeline of the updates is like this: if at time period t there is an interdiction attempt, the counter for the time period $t+1$ increases by one to denote which level of interdiction success probability to be used.

To allow the model to capture this nature of decision-dependent counting, we use interdiction decisions $\boldsymbol{x_\alpha}$ as flow capacities on A_α; the upper bound is x_α^t for arcs connecting a node (t, k) to node $(t + 1, k + 1)$ (thereby forcing a capacity of 0 if no intervention is taken on α in time period t) and $1 - x_\alpha^t$ for arcs connecting node (t, k) to node $(t + 1, k)$ (thereby forcing a capacity of 0 if an intervention is taken on α in time period t). The auxiliary interdiction counter variable, q, takes values of 0 and 1, yet because of the unimodularity of the constraints it is involved in, we define it as a linear variable to reduce the number of binary variables in the formulation. We give an example counting network in Figure 4.

The counting logic can be incorporated into both the arc- and path-based formulations using the constraints (16) - (20) below. These constraints use the following new pieces of notation. Let T be the set of stages (i.e., time periods). The lists $O(t, k)$ and $I(t, k)$ respectively store the outgoing and incoming arcs for each node (t,k) in the counting network.

$$\sum_{(t',k') \in O((0,0))} q_{\alpha,(0,0),(t',k')} = 1 \qquad \forall \alpha \in A \tag{16}$$

[1] Note that this is slightly different from the assumption made in [16], which assumes that an interdiction must be successful to impact the probability of success in future periods. Here we assume that any interdiction attempt will impact the probability of success in future periods.

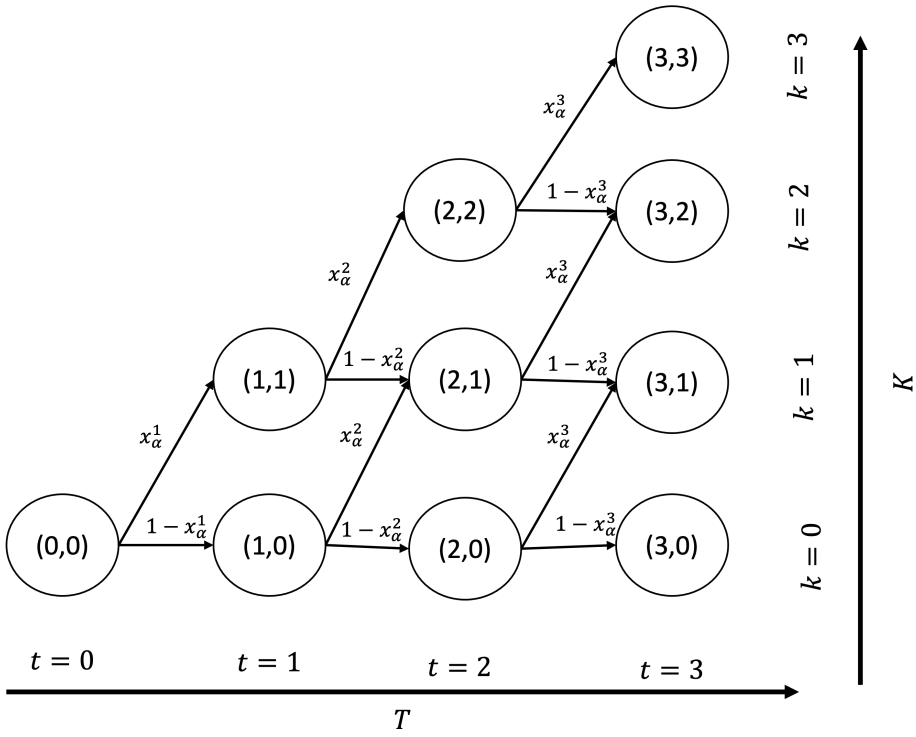


FIGURE 4 Counting network, C_α, for an arc α from graph G over time periods $T = \{0, 1, 2, 3\}$. The values on each arc in the counting network represent the counting flow capacity; this ensures the counting network properly counts the number of interdictions that have happened on network G

$$\sum_{(t',k')\in O((t,k))} q_{\alpha,(t,k),(t',k')} - \sum_{(t'',k'')\in I((t,k))} q_{\alpha,(t'',k''),(t,k)} = 0 \qquad \forall \alpha \in A, \forall (t,k) \in N_\alpha \setminus \{(0,0)\} \tag{17}$$

$$q_{\alpha,(t,k),(t+1,k)} \leq 1 - x_\alpha^t \qquad \forall \alpha \in A, \forall t \in T \setminus \{|T|-1\} \tag{18}$$

$$q_{\alpha,(t,k),(t+1,k+1)} \leq x_\alpha^t \qquad \forall \alpha \in A, \forall t \in T \setminus \{|T|-1\} \tag{19}$$

$$0 \leq q_{\alpha,i,j} \leq 1 \qquad \forall \alpha \in A, \forall (i,j) \in A_\alpha \tag{20}$$

Constraints (16) ensure an initial counting flow is pushed from the node $(0,0)$. (17) are flow balance constraints. (18) and (19) set the correct capacities on the counting flow based on previous interdictions. (20) are the domain constraints.

3.2.1 | Arc-Based Formulation

In this section, we present the multi-stage arc-based formulation. Variables denoting the flow on arcs are now indexed over time. i_n^t denotes the incoming probability flow for node n in G' at time $t \in T$.

$$\underset{\mathbf{i,l,r,x}}{\text{minimize}} \sum_{\omega \in \Omega} \sum_{t \in T} c_\omega * i_\omega^t \tag{21}$$

subject to

(16) – (20)

$$i_1^t = 1 \qquad \forall t \in T \tag{22}$$

$$i_n^t = i_{n_L}^t + i_{n_R}^t \qquad \forall n \in N' \setminus N'_{leaf}, \forall t \in T \tag{23}$$

$$i_{n_L}^t \leq x_{D(n)}^t \qquad \forall n \in N' \setminus N'_{leaf}, \forall t \in T \tag{24}$$

$$i_{n_L}^t \leq p_{D(n)}^k * i_n^t + 1 - \sum_{\substack{((t',k'),(t,k)) \\ \in I((t,k))}} q_{D(n),(t',k'),(t,k)} \qquad \forall n \in N' \setminus N'_{leaf}, \forall t \in T, \forall k \in \{1, \ldots, t\} \tag{25}$$

$$\sum_{\alpha \in A} x_\alpha^t \leq R \qquad \forall t \in T \tag{26}$$

$$0 \leq i_n^t, l_n^t, r_n^t, \leq 1 \qquad \forall n \in N' \setminus N'_{leaf}, \forall t \in T \tag{27}$$

$$x_\alpha^t \in \{0, 1\} \qquad \forall \alpha \in A, \forall t \in T \tag{28}$$

(21) is the objective function denoting the expected min-max flow over all stages. (16) - (20) are the interdiction counting constraints. (22) and (23) are pushing the initial probability flow of 1 and flow balance constraints for each stage in T. Constraints (24) and (25) are constraints describing how the probability flow splits should be handled. If there is no interdiction attempt at a stage $t \in T$, (24) ensures no split is happening. If there is an interdiction attempt, the probability flow should be split according to the correct interdiction attempt count k at stage t. Recall that the summation of incoming counting flow for a node (t, k) in graph C_α (the rightmost term in constraint 25) records which interdiction success probability should be used. Constraint 25 only becomes active when the summation is equal to one. (26) is the budget constraint and (27) and (28) are the domain constraints.

3.2.2 | Path-Based Formulation

In this section, we present the multi-stage path-based formulation. Note that the path flow variable is now indexed over time. f_p^t denotes the probability flow on path $p \in \mathcal{P}$ at time $t \in T$.

$$\underset{\mathbf{f,x,q}}{\text{minimize}} \sum_{t \in T} \sum_{p \in \mathcal{P}} c_p * f_p^t \tag{29}$$

subject to

(16) – (20)

$$\sum_{\alpha \in A} x_\alpha^t \leq R \qquad \forall t \in T \tag{30}$$

$$\sum_{p \in \mathcal{P}} f_p^t = 1 \qquad \forall t \in T \tag{31}$$

$$\sum_{p \in M(n)} f_p^t \leq x_{D(n)}^t \qquad \forall n \in N'_{left}, \forall t \in T \tag{32}$$

$$\sum_{p\in\mathcal{P}_n} f_p^t \le p_{D(n)}^k * \sum_{p\in\mathcal{P}_{parent(n)}} f_p^t + 1 - \sum_{((t',k'),(t'',k''))\in I((t,k))} q_{\alpha,(t',k'),(t'',k'')} \qquad \forall n \in N'_{left},$$

$$\forall t \in T, \forall k \in \{1,\dots,t\} \tag{33}$$

$$x_\alpha^t \in \{0,1\} \qquad \alpha \in A, t \in T \tag{34}$$

$$0 \le f_p^t \le 1 \qquad \forall p \in P, \forall t \in T \tag{35}$$

The objective (29) is the expected minimum maximum flow. (16) - (20) are constraints for counting the number of past interdictions on arcs. (30) is the interdiction budget constraint. (31) ensures that the probability flow in G' sums to one. (32) and (33) make sure that the appropriate decision-dependent upper bound limits each path's flow and ensures the correct flow amount by multiplying the incoming flows from the parent node with the arc interdiction probability p_α^k. Note that in (33), the variable q allows the model to activate the constraints for the appropriate p_α^k. If an arc is being interdicted at a time t for the kth time, the sum of incoming flow to node (t,k) in the counting network becomes one, so the constraints that involve that value can become active as it takes the form,

$$\sum_{p\in\mathcal{P}_n} f_p^t \le p_{D(n)}^k * \sum_{p\in\mathcal{P}_{parent(n)}} f_p^t$$

At each t, only one constraint from the set (33) can be active for each arc, and the active constraint corresponds to the correct k value at time t.

4 | SCENARIO-CLUSTERING FOR DECISION-DEPENDENT PROBLEMS

The models provided in Section 3 require considering a large number of scenarios that require an exponentially increasing number of variables for the purpose of linearization. This exponential increase is prohibitive and not practical. We propose and demonstrate the effectiveness of two scenario-clustering approaches to reduce the solution complexity by reducing the number of scenarios. As previously mentioned, each node in the scenario tree G' is a subgraph of the graph G on which interdiction decisions are being considered; therefore, we use the terms scenario and subgraph interchangeably. In section 4.1, we introduce a scenario clustering approach that preserves optimality by clustering the children scenarios once a parent scenario gets disconnected. Then, to provide greater flexibility to the user to balance the optimization and scenario tree generation run time vs optimality gap considerations, we propose a heuristic in 4.2 that yields a feasible (but not necessarily optimal) solution. We do this by determining each decision-dependent scenario's maximum occurrence probability (MOP) and then cluster the children if the MOP is below a user-provided threshold. Both methods prune and potentially reduce the size of the scenario tree, G'. Once the scenario tree is pruned and scenarios are clustered, they are used as inputs of the mixed-integer programs introduced in Section 3. Our scenario clustering methods aim to reduce the input size so larger networks can be analyzed without memory issues, and models are tractable for both larger networks and more stages.

4.1 | Scenario Clustering that Preserves Optimality

The naive way to generate the scenario tree discussed in Section 3 is to enumerate all potential subgraphs of a given graph and then construct the scenario tree where each subgraph is a leaf node. A graph G, with $|A|$ arcs has $2^{|A|}$ subgraphs. In this section, we show that generating all of these subgraphs is not always required. First, we show that

nodes in the scenario tree corresponding to disconnected subgraphs in G can be clustered. Secondly, because the interdiction budget precludes interdicting all of the arcs simultaneously, some of these subgraphs cannot be realized and can therefore be clustered. Since we assume that each interdiction has unit cost, there would be at most $2^{\sum_{i=0}^{R}\binom{|A|}{i}}$ subgraphs required to calculate the expected min-max flow under the interdiction decisions. We demonstrate these clustering approaches with an example and then with a more formal argument.

Consider the example problem in Figure 1. In this figure, node A is the source, and node C is the sink. We start the process with the full graph G as our root node (n_0) for the scenario tree. Then, for the root node's children, the left node (n_1) corresponds to the subgraph where arc 1 is removed, and the right child (n_2) is where the arc is not removed. Figure 5 illustrates the process of generating the binary scenario tree.

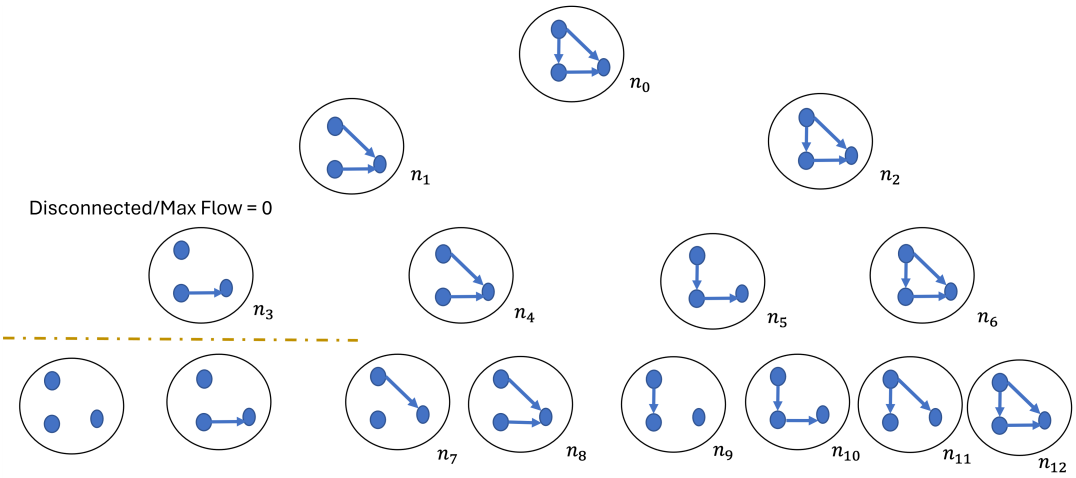


FIGURE 5 Scenario Tree G' pruned. Numbered circles are the unique subgraph identifier of the subgraph. Each node contains the corresponding (not necessarily unique) subgraph. There is no need to branch after subgraph 3, as it's disconnected, and all of it's children can be clustered into this scenario.

Note that the maximum-flow values of the subgraphs depicted in the leaf nodes are the objective function coefficients used to evaluate the expected maximum flow in the original graph G, and the corresponding probability flow variables from the scenario tree G' are the decision-dependent scenario probabilities. Therefore, we do not need to further branch children from scenarios where the maximum flow value is zero (i.e. graph is disconnected). This can be seen in node n_3. Since the maximum flow of the subgraph depicted in n_3 is already zero, and we can cluster n_3 with its children nodes (which correspond to the subgraph with no arcs and subgraph with only arc 3) as a single scenario that has an objective coefficient of zero. For this example, this observation reduces the number of leaf nodes from eight to seven, thereby reducing the problem size. We illustrate scenario reduction levels for different types of graphs and the impact of input size reduction on optimization runtimes in Section 5.

Lemma 1 *A child node in the binary scenario tree has a smaller or equal maximum flow to its parent.*

Proof Because each non-root node in the binary scenario tree is a subgraph of its parent node, each node will have no more augmenting paths than its parent node. Therefore, no more flow can be pushed from the source to the sink in the node as compared to its parent node.

Theorem 2 *Clustering scenarios at nodes where the corresponding subgraph is disconnected preserves the optimal solution.*

Proof We will use the path-based formulation in this proof. The same argument holds for the arc-based formulation without loss of generality. Let $z = \sum_{\omega \in \Omega} c_\omega * f_\omega$ denote the objective function of the optimization problem on the scenario tree G'. Let us also define the clustered scenario tree's objective function by $z_{pruned} = \sum_{\omega \in \Omega_{pruned}} c_\omega f_\omega$ where Ω_{pruned} are the scenarios that remain in the pruned scenario tree. Scenarios that are in Ω but not in Ω_{pruned} have a parent node that has a maximum flow of zero. By Lemma 1, the scenarios that are in Ω but not in Ω_{pruned} must therefore also have a maximum flow of zero (since the lower bound for a maximum flow is also zero by definition). Thus, $c_\omega = 0, \forall \omega \in \Omega \setminus \Omega_{pruned}$.

$$
\begin{aligned}
z &= \sum_{\omega \in \Omega} c_\omega * f_\omega \\
&= \sum_{\omega \in \Omega_{pruned}} c_\omega * f_\omega + \sum_{\omega \in \Omega \setminus \Omega_{pruned}} c_\omega * f_\omega \\
&= z_{pruned} + \sum_{\omega \in \Omega \setminus \Omega_{pruned}} c_\omega * f_\omega \\
&= z_{pruned} + 0 \\
&= z_{pruned}
\end{aligned}
$$

Note that the above proof is given for a single-stage version of the problem. Yet, it generalizes to the multi-stage version of the problem because it only uses the network topology to detect network disconnectedness.

Scenarios can also be clustered by considering the stage's interdiction budget. Specifically, we can stop branching within the scenario tree once the number of times we branch to the left in a path reaches the budget (assuming each node has the same interdiction cost of one unit budget). For example, if the problem instance for the example in Figure 1 had a budget of 1 for each stage, the pruned scenario tree is provided in Figure 6. This results in the scenario tree being reduced from 8 scenarios (i.e., no clustering) or 7 scenarios (i.e., clustering related to disconnectedness) down to 4 scenarios.

While implementing the scenario tree generation, we implement both clustering ideas simultaneously because they both preserve optimality.

4.2 | Scenario Clustering Heuristic Using Maximum Occurence Probability

In this section, we present a more flexible framework that does not necessarily preserve optimality. The previous scenario clustering approach was concerned with catching zero objective coefficients (maximum flows). With this heuristic, we focus on flow variables that can maximally take values below a user-provided threshold.

A decision-dependent scenario's maximum probability is the same as the maximum incoming flow for the corresponding node in the scenario tree. We first illustrate the idea on our ongoing example and then provide a more formal definition for the maximum occurrence probability for a decision-dependent scenario. Let's say we have interdiction success probabilities p_1, p_2 and p_3 for the example in Figure 1. Then the left child of node n_0 (i.e., n_1) in Figure 5 would have the maximum flow of p_1 if arc 1 in G is interdicted ($x_1 = 1$) because otherwise there will be no flow on this arc due to constraint (32). This is analogous to the fact that if no interdiction is taken on an arc, there will be zero probability for that arc to be removed. On the other hand, the right child of n_0 (i.e., n_2) either has an incoming flow of

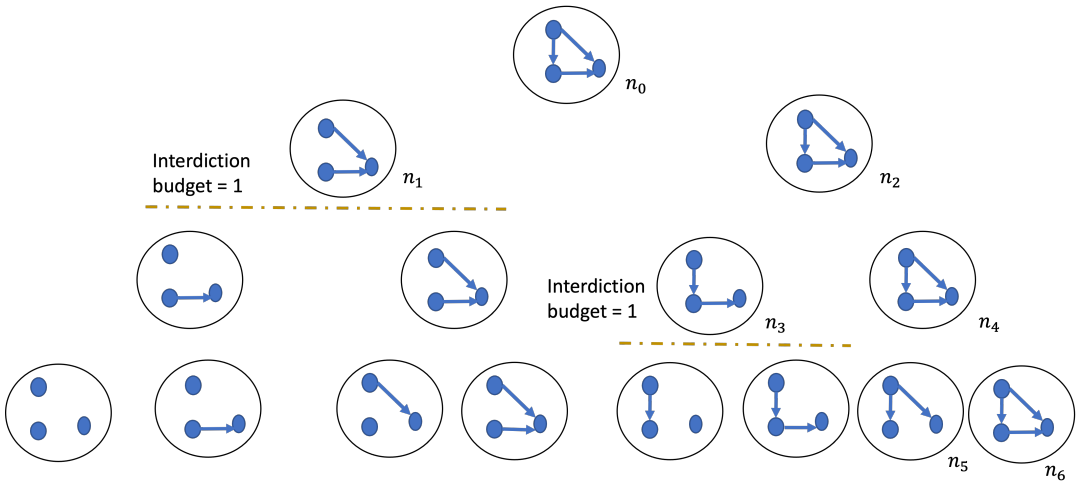


FIGURE 6 Pruned scenario tree for 1 where the interdiction budget, $R = 1$. Dashed lines denote the levels where scenario clustering should happen.

one (which occurs when there is no interdiction for arc 1, because no probability flow split happens) or $1 - p_1$ (which occurs when there is an interdiction, causing a split to happen)..Thus, n_2 receives its maximum flow value of 1 when there is no interdiction on arc 1.

When we extend this observation to the lower levels, we see that flows at right children either stay the same as the incoming flow to their parent node (no interdiction), or get multiplied by the failure probability. For the left children, the incoming flow is either zero (no interdiction) or the incoming flow to their parent node multiplied by the success probability. To formalize this observation, we define a value, *maximum occurrence probability (MOP)* for each node in n in scenario tree G'. This is an upper bound for the incoming probability flow for a node and denotes the maximum probability a scenario can have at that node's level. We can see that the left children get their maximum occurrence probability if the corresponding layer's arc is interdicted. Otherwise, zero flow will pass through them, i.e., their occurrence probability will be zero. For the right children, their maximum occurrence probability is when no interdiction happens, as the resulting incoming flow is fully directed to them.

Let's define the root node's maximum occurrence probability as $MOP_{n_0} = 1$. Any scenario/node n's MOP can be calculated from this initialization and the below definition.

$$MOP_n = \begin{cases} MOP_{parent(n)} * p_{D(n)} & \text{if } n \text{ is a left children,} \\ MOP_{parent(n)} & \text{if } n \text{ is a right children.} \end{cases}$$

We give the maximum occurrence probabilities for each scenario for the example graph and scenario tree in Figure 7. For example, the MOP of n_3 is the MOP of its parent node (i.e., $MOP_{n_1} = 0.1$) multiplied by the probability that arc 2 is successfully interdicted (i.e., $p_2 = 0.3$). Therefore, $MOP_{n_3} = 0.1 * 0.3 = 0.03$

Calculating a scenario's maximum possible value allows us to cluster scenarios with low likelihood under a parent scenario where the impact of interdictions is partially considered (up to the parent scenario's depth in the clustered branch). This grouping reduces the scenario space in a way that ignores the distribution of probabilities after a point in the scenario tree, hoping that not searching for the optimal probability flow for many small likelihood scenarios

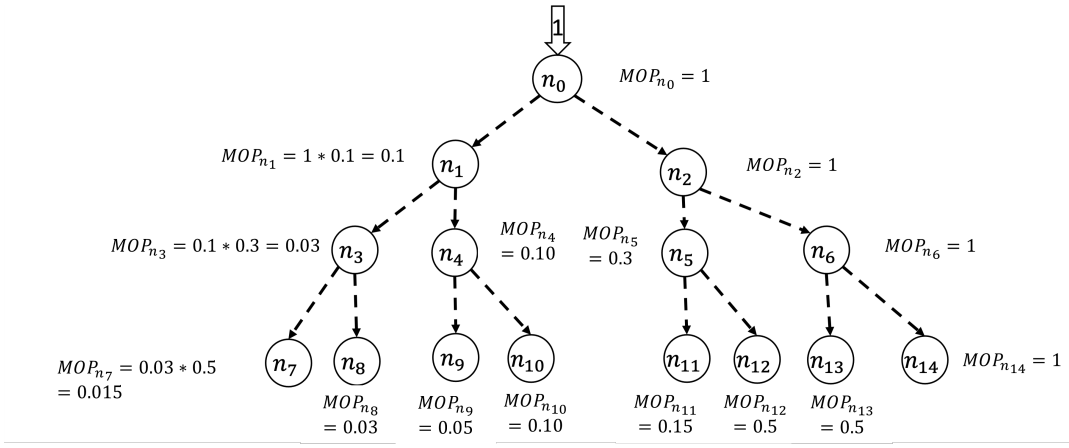


FIGURE 7 Scenario tree with maximum occurrence probabilities for each scenario for the example graph in 1, where $p_1 = 0.1, p_2 = 0.3, p_3 = 0.5$

would not affect the objective value drastically. For example, if we want to cluster scenarios that have a less than 10% maximum occurrence probability into a single parent scenario, we can cluster scenarios by stopping branching at the node n_1 in Figure 7. This would reduce the number of scenarios for the model to evaluate to 5 from 8 (since we then evaluate nodes 1 and 11-14 instead of nodes 7-14) . We analyze the trade-off between clustering with larger cutoffs and optimality gaps in Section 5.

5 | COMPUTATIONAL EXPERIMENTS

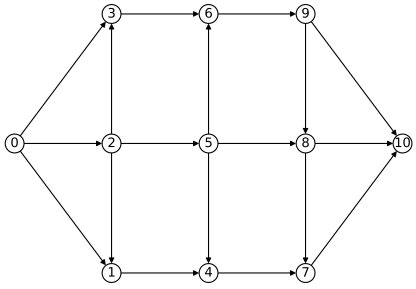
This section provides computational experiments to evaluate and compare the computational performances of the introduced models and scenario clustering techniques. First, we introduce the test network instances regarding their network topology and parameters in 5.1. In 5.2, we compare the performance differences of arc-based and path-based formulations introduced in Section 3. Finally, we observe the runtime-optimality gap trade-off from the scenario clustering techniques introduced in Section 4 in terms of runtime and optimality gaps in 5.3.

5.1 | Test Networks

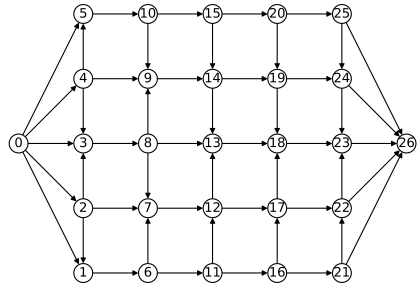
We work with two classes of networks as our test networks: grid networks (GN) and multi-echelon supply chain networks (SCN). We specifically selected these network types as they are relatively more connected and harder to disconnect with small interdiction budgets than networks with low-edge connectivity. Test networks, Python implementation of the MILPs with Gurobi API, and the scenario clustering algorithms implemented in Python can be accessed at: [Blinded for peer review].

We create $M \times N$ grid networks in the same way as [2, 5], where M denotes the number of layers and N denotes the number of nodes in a layer. For the supply chain networks, we use the notation $a \times b \times \ldots$, to denote a network where the first echelon contains a nodes, the second one contains b nodes, and so on. We evaluate the case where each node is connected to all nodes of the next echelon (see Figure 8c) and the case where a node is connected to only some in the next echelon (see Figure 8d). Network instances and the corresponding data can be seen in Figure

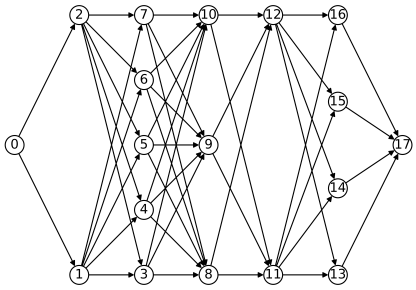
8 and at [Blinded for peer review]



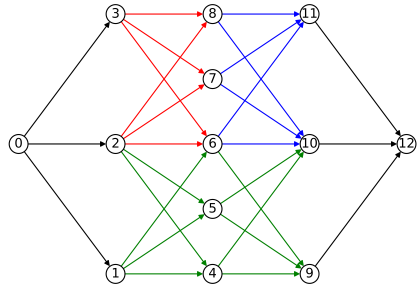
(a) 3 by 3 Grid



(b) 5 by 5 Grid



(c) Multi-echelon 1



(d) Multi-echelon 2, arcs with the same colors have the same interdiction probabilities.

FIGURE 8 Networks for test instances.Note that left-most nodes are the source and the right-most nodes are the sink, and arcs connected to these nodes can not be interdicted and have sufficiently large flow capacities.

For both classes, we assume that the cost of interdiction is one at all stages for each arc. We assume a flow capacity of 1000 and success probabilities of zero for arcs connected to the source and sink nodes. For the remaining arcs, the flow capacities are drawn independently from a uniform distribution with parameters 1 and 100. We test various success probabilities over stages; for each network instance, these can be found in the GitHub Repository. Some instances' probabilities are drawn randomly to create variation in instances, and some success probabilities are more structured because they correspond to more realistic applications (such as repeated interdictions losing their success probability over time or vice versa). We discuss these in Section 6, where we focus more on the interdiction decisions over time for different networks and different instances rather than the algorithmic performances.

5.2 | Impact of Arc-Based Formulation vs Path-Based Formulations with Optimality Preserving Scenario Clustering

We first compare the arc-based and path-based formulations with both of the the optimality preserving scenario clustering approaches (i.e. disconnectedness and budget prunes). Our analysis includes the optimization run times and the time it takes to create the input for the mixed integer linear programs from the raw input data (network

topology, arc capacities, and interdiction success probabilities). Note that the path-based formulation also requires tree search operations to identify the paths that go through the scenario nodes. Our implementation shows that the time for this operation is negligible compared to the scenario tree generation time and optimization runtimes. We acknowledge that there could be more efficient algorithms to identify the paths that go through each scenario tree node. However, our approach was sufficient for the given network sizes.

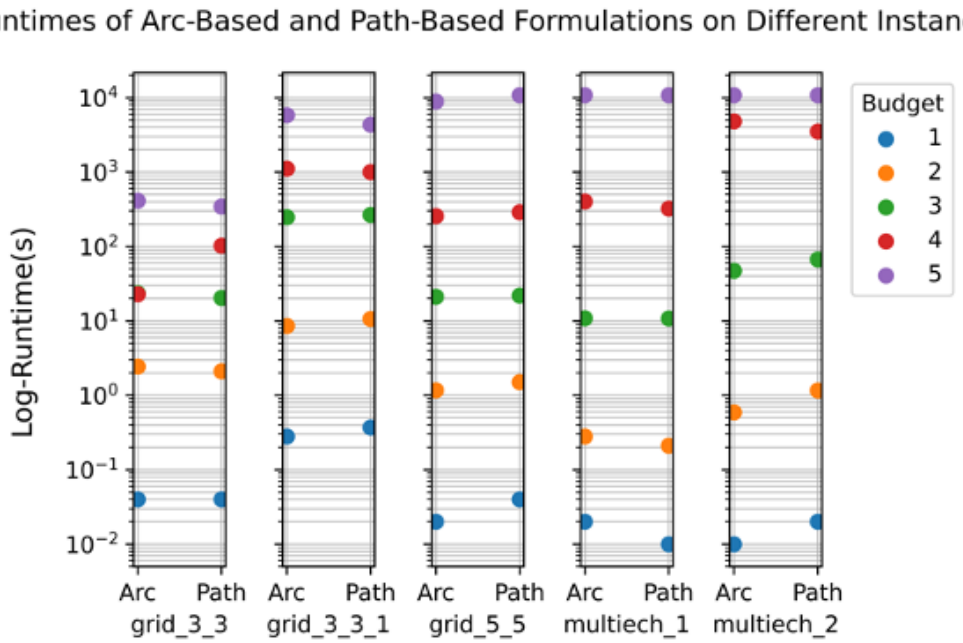


FIGURE 9 Differences in run times for test instances under various budgets.

In Figure 9, we see that both formulations have similar runtimes over various network instances and budgets. We observe that the runtime increases quickly as the interdiction budget increases.

From Table 1, we see that the runtime for generating the input is negligible compared to the optimization runtime. We acknowledge that for larger instances, the scenario tree generation may take significant time and memory, but from our experiments, the current bottleneck seems to be the optimization runtime. In Table 1, we highlight the scenario reductions under the clustering approaches that preserve optimality introduced in 4.1. We both present the reductions within the feasible (according to budget) interdiction space and the whole interdiction space to highlight the benefit of creating the scenario space dynamically rather than creating everything and then filtering. Reduction column is the ratio of the cardinality of the scenario space over the total number of feasible scenarios under the given budget. We also provide the ratio of the cardinality of the scenario space over the cardinality of the whole (feasible and infeasible) interdiction space for reference. In our largest instance, the 5 by 5 Grid with budget 5, we observe a 50% reduction in the scenario space by using the disconnectedness with-in the feasible interdiction space. Also, only working with interdictions that our budget allows results in large reductions as expected, ($\gtrsim$99%) for the input size, and our approach allows implementing this simple idea without first generating all scenarios and then filtering.

TABLE 1 Experiment results under arc-based and path-based formulations (Runtime columns with * indicate runtime limit reached.)

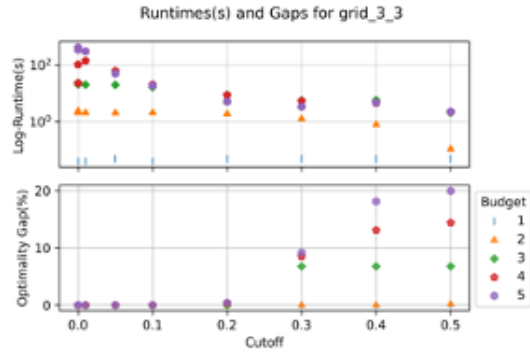
network_id	Budget	Cutoff	# of scenarios	Reduction	Reduction Total	Found obj.	Interdictions	Formulation	Runtime	Tree time(s)
grid_3_3	4	0	1592	0.607	0.99392	20.10166	{0: [3, 5, 7, 8], 1: [3, 5, 7, 8], 2: [3, 5, 7, 8], 3: [3, 5, 7, 8]}	Arc	119.75	0.11
								Path	102.42	0.13
grid_3_3	5	0	3458	0.726	0.98681	18.39501	{0: [3, 5, 7, 8, 9], 1: [3, 5, 7, 8, 9], 2: [3, 5, 7, 8, 9], 3: [3, 5, 7, 8, 9]}	Arc	412.92	0.37
								Path	343.38	0.27
grid_3_3_1	4	0	1592	0.607	0.99393	29.28766	{0: [3, 5, 7, 9], 1: [3, 5, 7, 9], 2: [7, 8, 12, 15], 3: [3, 8, 10, 12]}	Arc	1108.74	0.12
								Path	999.59	0.12
grid_3_3_1	5	0	3458	0.726	0.98681	28.89326	{0: [3, 5, 6, 7, 9], 1: [4, 8, 9, 12, 15], 2: [3, 5, 7, 8, 15], 3: [3, 7, 8, 10, 12]}	Arc	5816.21	0.23
								Path	4340.74	0.26
grid_5_5	4	0	148874	0.407	0.99999	3.35	{0: [18, 20, 25, 32], 1: [29, 32, 36, 40]}	Arc	256.10	21.34
								Path	288.26	21.01
grid_5_5	5	0	1183438	0.501	0.99999	2.54	{0: [18, 20, 22, 25, 32], 1: [29, 32, 34, 36, 40]}	Arc	8871.54	174.67
								Path	10811.45	174.49
multiech_1	4	0	131853	0.197	0.99999	287.02	{0: [28, 29, 30, 31], 1: [28, 29, 30, 31]}	Arc	402.20	18.52
								Path	322.67	19.92
multiech_1	5	0	1036614	0.252	0.99999	264.07	{0: [27, 28, 29, 30, 31], 1: [27, 28, 29, 30, 31]}	Arc	10802.71*	164.55
								Path	10808.37*	162.98
multiech_2	4	0	2*14494	0.4	0.99995	529.68	2*{0: [9, 12, 13, 16], 1: [9, 12, 13, 16]}	Arc	4808.00	1.17
								Path	3506.87	1.69
multiech_2	5	0	62773	0.487	0.99977	491.68	{0: [9, 12, 13, 16, 18], 1: [9, 12, 13, 16, 18]}	Arc	10800.18*	5.33
								Path	10800.58*	5.78

5.3 | Impact of Scenario Clustering

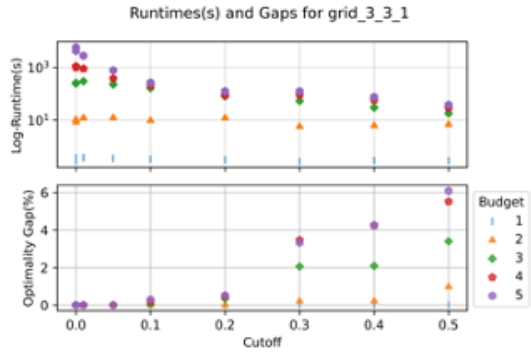
While generating results for the scenario clustering methods that preserve optimality, we observed that the optimization runtime hits our three-hour (10800 seconds) runtime limit for large instances, motivating us to test our heuristic clustering algorithm.

As we have seen in Section 5.2, both the arc-based and path-based formulations can get intractable for large networks and/or with multiple stages. In this section, we illustrate the impact of optimal and heuristic scenario-clustering. Our goal is to highlight the effectiveness of further scenario reduction with faster runtimes and also to analyze the optimality gap that comes with smaller scenario spaces generated heuristically. We have selected to use the path-based formulation so as not to duplicate the experiments.

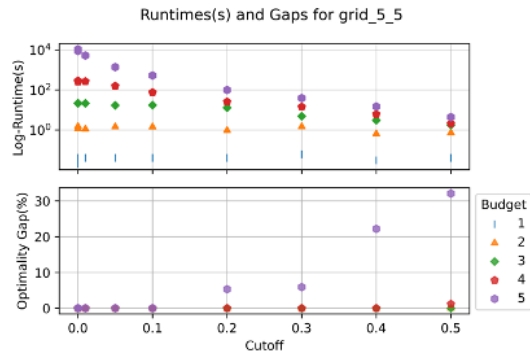
Figure 10 shows the resulting runtimes and the comparison of the heuristically found solutions with the best-found objectives for each problem instance. On the x-axis, we have varying cutoff levels for scenario clustering. Zero corresponds to the optimality-preserving pruning strategy we introduced in Section 4.1, and cutoff values greater than 0, are varying levels of thresholds for a scenario's maximum occurrence probability, which we introduced in 4.2. We also illustrate the optimality gaps and the (log-scaled) runtimes on the y-axis. Each color/shape denotes values for different budgets.



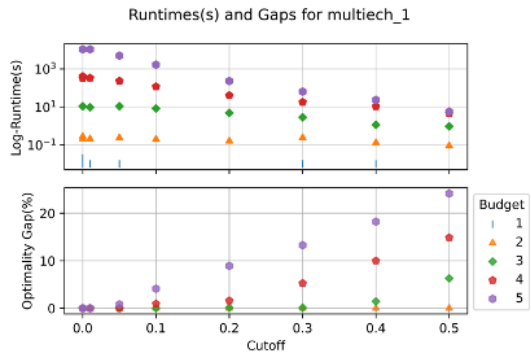
(a) Cutoff vs optimality gap for grid_3_3



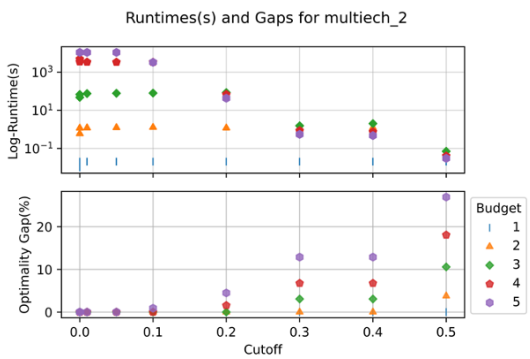
(b) Cutoff vs optimality gap for grid_3_3_1



(c) Cutoff vs optimality gap for grid_5_5



(d) Cutoff vs optimality gap for multiech_1



(e) Cutoff vs optimality gap for multiech_2

FIGURE 10 Cutoff vs (log) run time results under scenario clustering for test instances.

In all instances, we see that the cutoff value of 0.05 reduces the runtime drastically while (almost) preserving the optimality gap; the largest optimality gap when a 0.05 cutoff level is implemented is 0.008%. As the cutoff level increases, we see some deviations from the optimal solution as fewer and fewer scenarios are being used. However,

we see that experimenting with cutoffs can reduce the runtime drastically while still being close to optimality, giving the option to the user to make decisions around solution quality and runtimes. Our results suggest that for the types of instances that we have, the cutoff of 0.1 may provide a reasonable trade off between optimality and reducing the runtime by almost an order compared to the exact cutoff (of 0). In our examples, the largest optimality gap at a 0.1 cuttoff level is 0.04%.

6 | CONCLUSION

We introduced a solution methodology for solving the non-linear multi-stage decision-dependent network interdiction problem. Our contributions include two mixed integer reformulations and a scenario clustering algorithm to reduce the scenario space while preserving optimally. We also present a scenario clustering heuristic based on the maximum occurrence probability to further reduce solution times.We demonstrated the impact of our approach on various problem instances covering different network structures and parameters. The reformulations we provide to have a tractable version of the non-linear network interdiction with decision dependent success and the clustering and heuristic approaches we provide solve instances of these previously unexplored problems in reasonable times.

We also want to acknowledge potential limitations of the scenario clustering algorithm. In our implementation, we ordered the arcs corresponding to each level of the clustering algorithm by the order they were numbered in the original network (default order). We are aware that this ordering can and will affect the size of the scenario tree. Finding the optimal ordering to minimize the scenario tree is an optimization problem itself and is left for future research. However, we believe that investigating more strategic arc orderings, such as first branching on low success probabilities to cluster scenarios higher in the tree, is an interesting avenue of future research to further reduce the problem size for the proposed MILPs.

Another interesting future direction is generalizing our approach to other common inner/follower problems. Our discussion was based on the maximum flow problem, yet with some modifications, our approach is expected to be useful for shortest path and maximum reliability path problems as well.

ACKNOWLEDGMENTS

[Blinded for peer review.]

References

[1] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin, *Network flows: Theory, algorithms and applications*, title (1988).

[2] A. Atamtürk, C. Deck, and H. Jeon, *Successive quadratic upper-bounding for discrete mean-risk minimization and network interdiction*, INFORMS J. Comput. **32** (2020), 346–355.

[3] J.F. Bard, *Practical bilevel optimization: algorithms and applications* Vol. 30, Springer Science & Business Media, 2013.

[4] N.O. Baycik, T.C. Sharkey, and C.E. Rainwater, *Interdicting layered physical and information flow networks*, IISE Trans. **50** (2018), 316–331.

[5] K.J. Cormican, D.P. Morton, and R.K. Wood, *Stochastic network interdiction*, Oper. Res. **46** (1998), 184–197.

[6] H. Held and D.L. Woodruff, *Heuristics for multi-stage interdiction of stochastic networks*, J. Heuristics **11** (2005), 483–500.

[7] M. Hewitt, J. Ortmann, and W. Rei, *Decision-based scenario clustering for decision-making under uncertainty*, Ann. Oper. Res. **315** (2022), 747–771.

[8] U. Janjarassuk and J. Linderoth, *Reformulation and sampling to solve a stochastic network interdiction problem*, Networks **52** (oct 2008), 120–132.

[9] D. Kosmas, T.C. Sharkey, J.E. Mitchell, K.L. Maass, and L. Martin, *Interdicting restructuring networks with applications in illicit trafficking*, Eur. J. Oper. Res. **308** (2023), 832–851.

[10] M. Laumanns, S. Prestwich, and B. Kawas, *Distribution shaping and scenario bundling for stochastic programs with endogenous uncertainty*, Humboldt-Universität zu Berlin, Mathematisch-Naturwissenschaftliche Fakultät, 2014.

[11] A. Malaviya, C. Rainwater, and T. Sharkey, *Multi-period network interdiction problems with applications to city-level drug enforcement*, IIE Trans. (Institute Indus. Engineers) **44** (2012), 368–380.

[12] D.P. Morton, F. Pan, and K.J. Saeger, *Models for nuclear smuggling interdiction*, IIE Trans. (Institute Indus. Engineers) **39** (jan 2007), 3–14.

[13] J.R. O'Hanley, M.P. Scaparra, and S. García, *Probability chains: A general linearization technique for modeling reliability in facility location and related problems*, Eur. J. Oper. Res. **230** (2013), 63–75.

[14] S. Sadeghi and A. Seifi, *Stochastic maximum flow network interdiction with endogenous uncertainty*, Int. J. Supply Oper. Manage. **6** (2019), 200–212.

[15] J.C. Smith and Y. Song, *A survey of network interdiction models and algorithms*, Eur. J. Oper. Res. **283** (jun 2020), 797–811.

[16] B. Tezcan and K.L. Maass, *Human trafficking interdiction with decision dependent success*, Socio-Economic Planning Sci. (2023), 101521.

[17] R.K. Wood, *Deterministic network interdiction*, Math. Comput. Modelling **17** (1993), 1–18.

[18] J. Zhang, J. Zhuang, and B. Behlendorf, *Stochastic shortest path network interdiction with a case study of Arizona–Mexico border*, Reliab. Eng. System Saf. **179** (2018), 62–73.