Communications in Combinatorics and Optimization
Vol. xx, No. x (xxxx), pp. 1-19
https://doi.org/10.22049/cco.2026.30488.2496

CCO
Commun. Comb. Optim.

Research Article

Dynamic maximum capacity path interdiction with asymmetric information: strongly polynomial-time algorithms

Massoud Aman[1,†], Javad Tayyebi[2,*], Abolfazl Abdolahzadeh[1,‡]

[1] Department of Mathematics, Faculty of Science, University of Birjand, Iran
[†] mamann@birjand.ac.ir
[‡] a.abdolahzadeh@birjand.ac.ir

[2] Department of Industrial Engineering, Faculty of Industrial and Computer Engineering, Birjand University of Technology, Birjand, Iran
[*] javadtayyebi@birjandut.ac.ir

Received: 2 April 2025; Accepted: 20 May 2026
Published Online: 27 May 2026

Abstract: This paper introduces a novel non-cooperative game played on a capacitated network involving a defender and an attacker. The attacker seeks to maximize the capacity of a path from an origin to a destination, but operates under the constraint of limited network visibility, necessitating a greedy path selection strategy. Conversely, the fully informed defender aims to thwart the attacker's objective by strategically reducing arc capacities within a budgetary constraint. The game unfolds in multiple rounds, with the defender making capacity reduction decisions followed by the attacker's local path extension. This dynamic setting with asymmetric information characterizes the problem as the dynamic maximum capacity path interdiction problem. We present a strongly polynomial algorithm to solve this challenging game. Subsequently, an enhanced algorithm is developed to significantly improve computational efficiency. Extensive computational experiments validate the efficacy of both algorithms, demonstrating that the latter achieves approximately half the runtime of the former.

Keywords: Interdiction problem, dynamic game, asymmetric information, maximum capacity path, polynomial-time algorithm.

AMS Subject classification: 90C70, 91A80, 90C29

1. Introduction

The maximum capacity path problem is a fundamental combinatorial optimization problem defined on a capacitated network. The objective is to identify an s-t path, connecting an origin node s to a destination node t, that maximizes the minimum

* *Corresponding Author*

capacity among its constituent arcs. This problem belongs to the class of max-min (bottleneck) optimization problems and is also referred to as the widest path or bottleneck path problem [9, 23].

In many practical applications, such as the internet, the network traffic varies over time, resulting in dynamic changes to residual path capacities [19]. Moreover, in many real-world scenarios, decision-makers lack complete network information. Under such conditions, finding a globally optimal maximum capacity path becomes infeasible due to time constraints and information limitations. A practical alternative involves local decision-making, such as selecting the arc with maximum capacity among outgoing arcs at a given node, following a greedy strategy. This study focuses on such a scenario, motivated by military applications where the attacker, the first decision-maker, may possess limited knowledge of the enemy's force distribution within the designated area.

This paper studies a game involving two players with conflicting objectives on a capacitated network: a defender and an attacker. The attacker aims to select an s-t path to maximize the flow sent. Although, in military contexts, "flow" represents the attacker's forces, we employ standard network optimization terminology for a formal and independent problem definition. Due to incomplete information, the attacker can only make greedy decisions. Conversely, the defender seeks to minimize the attacker's flow by reducing arc capacities subject to a budget constraint. In internet networks, capacity reduction can be interpreted as increased path congestion caused by dummy data transmission through the path. For instance, see [15, 29] for research on Distributed Denial of Service (DDoS) attacks on internet-connected networks.

This game exhibits a sequential or dynamic structure [12, 35, 36, 45]. In each round, upon the attacker reaching a node, the defender first attempts to reduce the capacity of some arcs. Subsequently, the attacker selects the arc with maximum capacity among outgoing arcs to proceed. As the sum of the players' payoffs equals zero, this game is considered a zero-sum game. Given the hierarchical strategy selection, this game belongs to the class of Stackelberg games [17]. In the network optimization literature, such games are referred to as network interdiction problems [37]. Furthermore, due to the attacker's complete information and limited observation to outgoing arcs from their position, this game involves asymmetric information [10, 39].

This paper considers the aforementioned game with a budget constraint and lower-bound restrictions for the defender. We assume two types of costs for reducing arc capacity: a linear cost and a fixed cost. The total cost is the sum of these costs. This paper presents a strongly polynomial time algorithm to solve the game. Subsequently, an algorithm modification is developed based on several observations. The performance of the algorithms is evaluated through computational experiments.

Let us formally state a military application of the game in more detail. Suppose a group of attackers attempts to seize an area. Among all possible paths to the area, they tend to choose a wide path as it facilitates the movement of heavy equipment and offers a safer advance. Conversely, the defenders aim to prevent this by creating obstacles, such as deploying war mines. Due to the attackers' limited knowledge of the

defenders' strategy, their network information is incomplete, necessitating a greedy path selection after examining all arcs ahead. This exemplifies a situation where a player must make greedy decisions. From the defenders' perspective, an optimal strategy involves directing the attackers towards a critical point and then concentrating all forces to interdict their advancement at that point. This fundamental concept underlies our proposed algorithms.

1.1. A short literature review

The earliest form of network interdiction problems emerged in the 1960s when US military researchers sought to disrupt the transportation of Vietnamese troops and military equipment during the Vietnam War [38]. Since then, network interdiction models have been applied to a wide range of applications, including disease control [6], counterterrorism [13], and the interdiction of weapons and nuclear materials smuggling [27, 39].
The most common interdiction problems fall into five categories:

- **Routing Interdiction** [7, 8, 20, 28, 30, 40]
- **Maximum Capacity Path Interdiction** [11, 26, 41]
- **Maximum Flow Interdiction** [21, 42, 44]
- **Minimum Cut Interdiction** [1]
- **Minimum Spanning Tree Interdiction** [2, 33, 43]

Some interdiction problems are modified versions of these five problems. The most common generalizations that have led to new problems include: arc or node elimination [22], undirected graphs [44], multi-terminal problems [4], deterministic or stochastic interdiction [5, 14, 18], static or dynamic networks [24], symmetric or asymmetric information [10], and adding a new level for fortification [25, 32].

Due to focusing closely on the subject, let us review the maximum capacity path interdiction problem in the literature. There are only two papers which consider this problem: one with fixed costs [26], and the other with linear costs [41]. In the first case, the authors proposed a successive minimum-cut algorithm to solve the problem in polynomial time. In the second case that was recently published, the authors used a discrete version of the Newton algorithm to present a polynomial-time algorithm to solve the problem. In both papers, the game is assumed to be static, namely, it contains only one round where the defender first changes arc capacities and then, the attacker choose a path. In this paper, we assume that the game contains several rounds. Such games were referred to as "dynamic games" [35, 36]. To cover both the costs, the game is designed in a way that the total cost is a linear combination of fixed and linear costs. Although it was proved that dynamic shortest path interdiction problems [35] and dynamic assignment interdiction problems [36] are NP-hard,

this paper presents two polynomial-time algorithms to show the dynamic maximum capacity path problems belong to the class P.

Focusing closely on the subject, we review the maximum capacity path interdiction problem in the literature. Only two papers address this problem: one with fixed costs [26] and the other with linear costs [41]. In the first case, the authors proposed a successive minimum-cut algorithm to solve the problem in polynomial time. In the second case, the authors used a discrete version of the Newton algorithm to present a polynomial-time algorithm for the problem. Both papers assume a static game with a single round where the defender first modifies arc capacities, followed by the attacker's path selection. This paper considers a multi-round game, referred to as "dynamic games" [35, 36]. To encompass both cost types, the game is designed with a total cost as a linear combination of fixed and linear costs. While dynamic shortest path interdiction [35] and dynamic assignment interdiction [36] problems are NP-hard, this paper presents two polynomial-time algorithms, demonstrating that dynamic maximum capacity path problems belong to the class P.

The remainder of this paper is organized as follows. Section 1.1 reviews significant works in the field of network interdiction problems. Section 2 provides initial concepts and formally states the game. The proposed algorithms are described in Section 3. Computational results are reported in Section 4. Finally, Section 5 offers concluding remarks.

2. Preliminaries and problem statement

This section presents fundamental concepts and notations from graph theory that will be utilized throughout the paper. It subsequently introduces the formal framework of the game and provides an illustrative example.

Let $G(V, A)$ be a connected directed graph, where V represents the set of n nodes and A denotes the set of m arcs. Within this graph, two specific nodes are identified: the origin node s and the destination node t.

A path P is defined as a sequence of nodes $v_1 - v_2 - \ldots - v_k$ such that there exists an arc $(v_l, v_{l+1}) \in A$ for every $l = 1, 2, \ldots, k-1$. An arc (i, j) is said to be part of the path P, denoted as $(i, j) \in P$, if the nodes i and j are consecutive elements in the sequence. A cycle is defined as a path $v_1 - v_2 - \ldots - v_k$ that also includes the arc (v_k, v_1). Lastly, a path is referred to as an s-t path if its first and last nodes are s and t, respectively.

Assume that each arc (i, j) is associated with a nonnegative capacity u_{ij}. The capacity of an s-t path P, denoted by u_P, is defined as the minimum capacity of its arcs, i.e., $u_P = \min_{(i,j)\in P} u_{ij}$. The maximum capacity path problem is a well-known combinatorial optimization problem that aims to find an s-t path with maximum capacity. Several algorithms solve this problem in polynomial time, including a modified version of Dijkstra's algorithm [34] and a recursive algorithm [31].

In this paper, we assume that the decision maker, the attacker, lacks complete in-

formation about the network structure. Instead, the attacker can only observe the outgoing arcs at their current location. Consequently, the attacker makes local decisions by selecting the arc with the maximum capacity among all visible outgoing arcs. This greedy strategy is not necessarily optimal. However, this paper focuses on such a strategy due to its relevance in certain applications discussed in Section 1. For example, Figure 1 illustrates a network with arc capacities. The maximum capacity path is $s-1-4-t$ with a capacity of 5. In contrast, the attacker traverses the path $s-1-2-t$ with a capacity of 4, as they choose arc $(1,2)$, which has a greater capacity than $(1,4)$ whenever they are at node 1.

Now, suppose there is another decision-maker, called the defender, whose goal conflicts completely with that of the attacker. The defender is capable of decreasing arc capacities to reduce the capacity of the attacker's desired path as much as possible. Let $\tilde{u}_{ij}$ be the reduced capacity of (i,j).
It is assumed that the defender faces two types of constraints:

Budget Constraint: There are two types of costs for decreasing the capacity of an arc (i,j): (I) a fixed cost f_{ij} and (II) a linear cost r_{ij}. If the reduced capacity of an arc (i,j) is denoted by $\tilde{u}_{ij}$, then its cost is $f_{ij}H(u_{ij},\tilde{u}_{ij}) + r_{ij}(u_{ij}-\tilde{u}_{ij})$, where $H(u_{ij},\tilde{u}_{ij})$ is the Hamming distance between u_{ij} and $\tilde{u}_{ij}$, defined as:

$$H(u_{ij},\tilde{u}_{ij}) = \begin{cases} 0 & \text{if } u_{ij} = \tilde{u}_{ij}, \\ 1 & \text{otherwise.} \end{cases}$$

Let R be the total available budget. The budget constraint states that the total consumed cost must not exceed the total budget, i.e.,

$$\sum_{(i,j)\in A} \big(f_{ij}H(u_{ij},\tilde{u}_{ij}) + r_{ij}(u_{ij}-\tilde{u}_{ij})\big) \leq R.$$

Bound Restrictions: Let $\underline{u}_{ij}$ be a lower bound on the capacity of (i,j). The reduced capacity of each arc (i,j) cannot be less than $\underline{u}_{ij}$. Thus,

$$\underline{u}_{ij} \leq \tilde{u}_{ij} \leq u_{ij}$$

for every $(i,j) \in A$.

Remark 1. From the viewpoint of the application stated in Section 1, the fixed cost can be interpreted as the cost of transporting equipment to the arc, and the linear cost is proportional to the magnitude of the capacity reduction. The lower bound represents the minimum possible capacity to which the defender can reduce the capacity of an arc using available equipment.

This game unfolds sequentially, involving multiple rounds. Initially, the attacker is positioned at the origin node s. The defender begins by decreasing arc capacities. Subsequently, the attacker observes the defender's actions and selects the outgoing arc from s with the maximum capacity for advancement. Assuming the attacker chooses (s, i), the second round commences with the attacker at node i. The defender then reduces arc capacities again, and the attacker selects the preferred outgoing arc from i.

Although the defender possesses complete network information and can modify the capacity of any arc in each round, they can restrict their strategy to decreasing outgoing arcs from the attacker's current node, as the attacker's decisions are based on these capacities. Consequently, both players concentrate solely on the outgoing arcs of a node in each round. This game is termed the dynamic maximum capacity path interdiction (DMCPI) problem with asymmetric information. An illustrative example is provided in Figure 1.

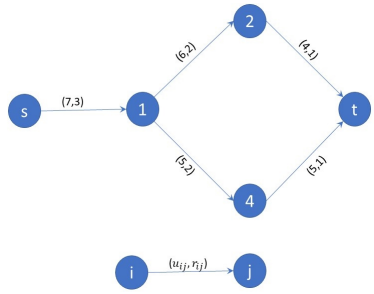

Figure 1. An example of the interdiction problem with zero lower bounds, zero fixed costs, and $R = 3$

Example 1. Figure 1 depicts an instance of the dynamic maximum capacity path problem. For this instance, the lower bound and fixed cost are assumed to be zero for every arc, and the total budget is 3.

Due to budget limitations, the defender can only reduce the capacity of $(s, 1)$ by one unit, which is ineffective. Thus, no budget is expended in the first round, and the attacker advances to node 1. In the second round, if the defender attempts to reduce the capacity of $(1, 2)$ by using the entire budget, the attacker selects arc $(1, 3)$ for advancement, resulting in the s-t path $s - 1 - 3 - t$ with capacity 5. Conversely, if the defender remains inactive in the second round, the attacker proceeds to node 2 as the capacity of $(1, 2)$ exceeds that of $(1, 3)$. In this scenario, the defender can reduce the capacity of $(2, t)$ from 4 to 1. It is evident that this strategy is optimal for the attacker. Therefore, the optimal value of the problem in this case is 1.

3. Algorithm design

In this section, we design an efficient algorithm that solves the problem in strongly polynomial time. Subsequently, we propose a modification to enhance the overall

running time.

For the moment, assume the defender knows the s-t path P that the attacker will select. As the arc in P with minimum capacity determines the attacker's objective value, the defender can concentrate on reducing the capacity of this arc. The following lemma establishes the attacker's strategy under these circumstances.

Lemma 1. *For path networks, it suffices to reduce the capacity of the arc* (i_0, j_0) *selected as follows:*

$$(i_0, j_0) = \arg \min_{(i,j)\in A} \left\{ \max\{\underline{u}_{ij}, u_{ij} - \frac{R}{r_{ij}}\} \right\}. \tag{3.1}$$

Proof. Given the preceding argument, the defender selects a single arc to reduce its capacity, thereby allocating the entire budget to an arc (i, j). This implies the inequality $\tilde{u}_{ij} \geq u_{ij} - \frac{R}{r_{ij}}$ from the budget constraint. Consequently, the maximum possible reduction of an arc (i, j) is $\max\{\underline{u}_{ij}, u_{ij} - \frac{R}{r_{ij}}\}$. This justifies the lemma. □

Lemma 1 presents a straightforward strategy for path networks, where the defender focuses on reducing the capacity of a single critical arc. However, this approach is limited to such simple graph structures. In more complex networks, the attacker can adapt their path in response to capacity reductions, necessitating a more sophisticated defense strategy.

Despite the increased complexity, the fundamental principle that a single arc's capacity determines the attacker's objective value remains valid. This observation suggests a potential approach: defining a feasible defense strategy for each arc and then selecting the optimal strategy from this set. A feasible strategy, in this context, is one that respects both the budget and capacity constraints.

To simplify the problem further, we shift our focus from arcs to nodes. Consider the following observation.

Observation 1. If the objective value is equivalent to the modified capacity of an arc (i, j), then the capacities of all outgoing arcs from node i must be less than or equal to that of (i, j).

This observation implies that by focusing on node-based strategies, the defender can indirectly control the capacities of outgoing arcs and, consequently, the attacker's objective value.

Building on this insight, we introduce the concept of a modified capacity vector, denoted by $\tilde{\mathbf{u}}^{(v)}$, for each node v in the network. This vector represents a potential defense strategy that aims to direct the attacker towards node v and subsequently reduce the capacities of its outgoing arcs. As we will demonstrate, at least one optimal solution can be found among these node-based strategies.

The vector $\tilde{\mathbf{u}}^{(v)}$ must be constructed to direct the attacker towards node v. Subsequently, the capacities of the outgoing arcs from v are reduced to minimize the

attacker's objective value. Thus, the defender initially incurs the minimum possible cost to guide the attacker towards node v. Then, the remaining budget is entirely allocated to reducing the capacities of the outgoing arcs from v. The process of defining the solution $\tilde{\mathbf{u}}^{(v)}$ is divided into two primary steps:

- Directing the attacker towards node v.
- Decreasing the outgoing arcs of v.

We provide detailed explanations of these steps in the following subsections.

3.1. Directing the attacker toward a node

To direct the attacker towards a node v, the defender must modify capacities to gradually steer the attacker along a path from s to v. However, capacity changes incur costs. Thus, the defender aims to find the least costly path from s to v to maximize the remaining budget for subsequent capacity reductions. To this end, a cost is assigned to each arc (i, j), representing the cost of incentivizing the attacker to choose (i, j) over other outgoing arcs from node i. A shortest path algorithm is then employed to determine the minimum-cost path from s to v.

To define a cost vector for a node v, the arc set A is partitioned into two subsets: A_v, consisting of arcs that belong to at least one path from s to v, and $\bar{A}_v = A \setminus A_v$, containing the remaining arcs. This partition can be achieved by executing a traversal algorithm twice: once from s and again from v in the reverse direction. Algorithm 1 provides a formal description of this procedure.

Algorithm 1 Computing the set A_v

Require: A graph $G(V, A)$ with two specified nodes s and v.
Ensure: The arc set A_v.
1: Run Breadth-First Search (BFS) from node s to identify all accessible arcs. Let $\tilde{A}_s$ denote the set of these arcs.
2: Run BFS from node v in the reverse direction to find all arcs with paths to v. Let $\tilde{A}_v$ denote the set of these arcs.
3: Compute $A_v = \tilde{A}_s \cap \tilde{A}_v$.

Now, let us define a cost vector $\mathbf{c}^v$ as

$$c_{ij}^v = \begin{cases} +\infty & (i,j) \in \bar{A}_v, \\ \sum_{k \in V:(i,k)\in A \text{ and } u_{ik} > u_{ij}} \big(f_{ik} + r_{ik}(u_{ik} - u_{ij})\big) & (i,j) \in A_v, \end{cases} \quad \forall (i,j) \in A. \tag{3.2}$$

To justify this definition, observe that the attacker selects arc $(i, j) \in A_v$ at node i only if the defender modifies capacities such that the capacity of (i, j) becomes greater than or equal to the other outgoing arcs of i. Consequently, the defender can set the capacities of the outgoing arcs of i as follows:

$$\tilde{u}_{ik} = \begin{cases} u_{ij} & k \neq j \text{ and } u_{ik} \geq u_{ij}, \\ u_{ik} & k = j \text{ or } u_{ik} < u_{ij}. \end{cases} \tag{3.3}$$

This incurs a cost of $\sum_{(i,k)\in A:u_{ik}\geq u_{ij}} f_{ik} + r_{ik}(u_{ik} - u_{ij})$ to the defender. To prevent the attacker from selecting an arc $(i,j) \in \bar{A}_v$ (since there is no path from such an arc to v), we assign an infinite cost to these arcs.

To complete this step, a shortest path algorithm is applied to the cost vector $\mathbf{c}^v$ to identify a minimum-cost path from s to v, representing the optimal route to direct the attacker towards node v.

Algorithm 2 Directing the attacker towards a specific node

Require: A graph $G(V, A)$ with arc capacities u_{ij}, arc costs f_{ij} and r_{ij}, and two specified nodes s and v.

Ensure: A modified capacity vector $\tilde{\mathbf{u}}$, a path P from s to v, the capacity of P with respect to the modified capacity vector, and the total cost R_1 of directing the attacker towards v.

1: Apply Algorithm 1 to determine the set A_v.
2: Compute the cost vector $\mathbf{c}^v$ using Equation (3.2).
3: Find a shortest path P from s to v with respect to the cost vector $\mathbf{c}^v$. Let R_1 be its cost.
4: Compute the modified capacity vector $\tilde{\mathbf{u}}$ for the outgoing arcs of nodes in P.
5: Calculate the capacity of path P based on the modified capacities.

Lemma 2. *Algorithm 2 finds a path with minimum cost in $O(mn)$ time.*

Proof. Computing the set A_v using breadth-first search requires linear time, $O(m)$. As the arc costs are non-negative, the Fibonacci-heap implementation of Dijkstra's algorithm can be employed to find the shortest path in $O(m + n \log n)$ time [3]. The most computationally intensive step is the calculation of the cost vector $\mathbf{c}^v$, which has a time complexity of $O(mn)$. Consequently, the overall time complexity of Algorithm 2 is dominated by the computation of the cost vector, resulting in a total running time of $O(mn)$. □

3.2. Decreasing the outgoing arcs

Having established a method for directing the attacker towards a specific node v at minimal cost, we now focus on the second step: utilizing the remaining budget to reduce the capacities of the outgoing arcs of v in a way that minimizes the attacker's objective value. This can be achieved by solving the following optimization problem:

$$\min \quad z = \max_{j\in\bar{V}_v}\{u_{vj} - x_{vj}\} \tag{3.4a}$$

$$\text{s.t.} \quad \sum_{j\in\bar{V}_v} \left(f_{vj}y_{vj} + r_{vj}x_{vj}\right) \leq \bar{R}, \tag{3.4b}$$

$$0 \leq x_{vj} \leq (u_{ij} - \underline{u}_{ij})y_{vj} \quad \forall j \in \bar{V}_v, \tag{3.4c}$$

$$y_{vj} \in \{0,1\} \quad \forall j \in \bar{V}_v, \tag{3.4d}$$

where $\bar{V}_v = \{j \in V : (v,j) \in A\}$, x_{vj} represents the capacity reduction of arc (v,j) (i.e., $x_{vj} = u_{ij} - \tilde{u}_{ij}$), and y_{vj} is a binary decision variable indicating whether a capacity reduction is applied to arc (v,j). The parameter $\bar{R}$ denotes the budget remaining after the first step.

Problem (3.4) can be classified as a knapsack problem with continuous and fixed costs, featuring a bottleneck objective function. Consequently, it can be efficiently solved using a greedy approach. Algorithm 3 outlines the solution procedure.

Lemma 3. *The worst-case time complexity of Algorithm 3 is* $O(n^2)$.

Proof. The While loop terminates if either $\delta = \delta_1$ or $\delta = \frac{\bar{R}-f_{total}}{r_{total}}$. Therefore, the loop iterates only when $\delta = z^1_{max} - z^2_{max}$, implying that the values z^1_{max} and z^2_{max} decrease in each iteration except the last. As these values are selected from a set of at least n distinct values, the number of iterations is bounded by $O(n)$. Since the For loops and the calculation of z^2_{max} also have a time complexity of $O(n)$, the overall time complexity of the algorithm is $O(n^2)$. □

Algorithm 3 Decreasing the outgoing arcs of v

Require: An instance of problem (3.4)
Ensure: Optimal decision variables x_{ij}, y_{ij} and the optimal value z^1_{max}.
1: Initialize $x_{vj} = 0$ and $y_{vj} = 0$ for all $j \in \bar{V}_v$.
2: Determine the maximum and second maximum values among the distinct values $u_{vj} - x_{vj}$, denoting them as z^1_{max} and z^2_{max}, respectively.
3: **while** True **do**
4: Initialize $S = \emptyset$, $\delta_1 = M$, $\delta = 0$, $r_{total} = 0$, and $f_{total} = 0$.
5: **for** $k \in \bar{V}_v$ **do**
6: **if** $u_{vk} - x_{vk} = z^1_{max}$ **then**
7: Add k to S.
8: Update $r_{total} = r_{total} + r_{vk}$ and $f_{total} = f_{total} + f_{vk}$.
9: **if** $\delta_1 > u_{vk} - \underline{u}_{vk}$ **then**
10: Set $\delta_1 = u_{vk} - \underline{u}_{vk}$.
11: **end if**
12: **end if**
13: **end for**
14: Calculate $\delta = \min\{\delta_1, z^1_{max} - z^2_{max}, \frac{\bar{R}-f_{total}}{r_{total}}\}$.
15: **for** $k \in S$ **do**
16: Set $y_{vk} = 1$ and $x_{vk} = x_{vk} + \delta$.
17: **end for**
18: **if** $\bar{R} = 0$ or $\delta = \delta_1$ **then**
19: Break
20: **end if**
21: Set $z^1_{max} = z^2_{max}$ and recompute z^2_{max}.
22: **end while**

3.3. Algorithm Description

We now formally present our proposed algorithm for solving the DMCPI problem. The algorithm iteratively computes a solution for each node $v \in V$ using Algorithms 2 and 3, and ultimately selects the optimal solution among these candidates. Algorithm 4 provides a detailed description.

Algorithm 4 Solving the DMCPI problem

Require: An instance of the DMCPI problem defined by a graph $G(V, A)$, capacity vector $\mathbf{u}$, lower bound vector $\underline{\mathbf{u}}$, cost vector $\mathbf{r}$, and budget value R.
Ensure: An optimal strategy profile $(\tilde{\mathbf{u}}^*, P^*)$ and the corresponding payoff value z^*.
1: Initialize $z^* = M$.
2: **for** $v \in V \setminus \{t\}$ **do**
3: Apply Algorithm 2 to find a path P from s to v, a modified capacity vector $\tilde{\mathbf{u}}$, the capacity of P with respect to $\tilde{\mathbf{u}}$, and the total cost R_1 of directing the attacker towards v.
4: **if** $R > R_1$ **then**
5: Set $\bar{R} = R - R_1$.
6: Apply Algorithm 3 to compute z^1_{max}.
7: **if** $z^* > z^1_{max}$ **then**
8: Set $z^* = z^1_{max}$ and $\tilde{\mathbf{u}}^* = \tilde{\mathbf{u}}$.
9: **end if**
10: **end if**
11: **end for**

The correctness of Algorithm 4 is evident as it exhaustively examines all nodes to identify the one whose outgoing arc determines the optimal value. The following theorem establishes the algorithm's time complexity:

Theorem 2. *Algorithm 4 solves the DMCPI problem in $O(n^2m)$ time.*

Proof. The algorithm iterates through each node, resulting in an outer loop with $O(n)$ iterations. Since Algorithms 2 and 3 have time complexities of $O(mn)$ and $O(n^2)$, respectively, the overall time complexity of Algorithm 4 is dominated by the nested loop structure, leading to a total running time of $O(n^2m)$. □

Algorithm 4 employs a straightforward approach to address the Dynamic Maximum Capacity Path Interdiction (DMCPI) problem. While the overall structure of the algorithm is effective, a crucial observation can lead to improvements in its running time. To clarify this point, it is essential to recognize that the bottleneck operation within the algorithm is the process of finding the shortest path, which is carried out by Algorithm 2.

This step of calculating the shortest path can become computationally intensive, particularly if it has to be performed repeatedly for many scenarios. Therefore, if we can identify cases where certain shortest paths do not need to be computed, we can significantly enhance the efficiency of Algorithm 4.

This optimization can be implemented through the following proposition: by analyzing the characteristics of the network and the specific conditions of the DMCPI problem, we can pre-determine which paths are likely to be optimal and which can be excluded from consideration. This selective exclusion reduces the number of computations required since we focus only on those paths that have a reasonable chance of being the shortest. Consequently, by minimizing unnecessary calculations, we can substantially decrease the overall running time of the algorithm, allowing it to perform more efficiently and effectively in solving the DMCPI problem.

Proposition 1. *Let P be the output of Algorithm 2, representing a shortest path from the source node s to a node v. If a node w is part of the path P, then the output of Algorithm 2 when searching for a shortest path from s to w is the segment of P that connects s to w.*

Proof. By definition, if there exists at least one path from the source s to the destination node v that includes the node w, then the set of arcs associated with w (denoted as A_w) must be a subset of the arcs associated with v (denoted as A_v). This relationship can be expressed as $A_w \subseteq A_v$.
This observation leads us to conclude that the costs associated with the arcs must also reflect this relationship. Specifically, for every arc $(i, j) \in A_w$, the cost function must satisfy $c_{ij}^w = c_{ij}^v$. Conversely, for arcs not belonging to A_w (but possibly included in A_v), the cost will be represented as $c_{ij}^w = \infty$. This indicates that these arcs cannot be traversed in the context of finding the shortest path from s to w.
Now, assume that there exists an alternative path $\bar{P}$ from s to w which has a total length, when measured according to the cost structure c_{ij}^w, that is less than that of the portion of P connecting s to w (denoted as P'). This leads us to consider the combined path $\bar{P} \cup (P \setminus P')$ which forms a path from s to v.
Given that $\bar{P}$ was found to be shorter than the segment P', this combined path $\bar{P} \cup (P \setminus P')$ must consequently have a length that is shorter than the length of P when evaluated under the cost structure c_{ij}^v. This result directly contradicts the definition of P being the shortest path from s to v. Consequently, our assumption that such a path $\bar{P}$ exists must be incorrect. □

From Proposition 1, it is unnecessary to execute Algorithm 2 for every node that lies on the shortest path to another node. Algorithm 5 incorporates this optimization within the framework of Algorithm 4.

Algorithm 5 A modified version of Algorithm 4

Require: An instance of the DMCPI problem defined by a graph $G(V, A)$, capacity vector $\mathbf{u}$, lower bound vector $\underline{\mathbf{u}}$, cost vector $\mathbf{r}$, and budget value R.
Ensure: An optimal strategy profile $(\tilde{\mathbf{u}}^*, P^*)$ and the corresponding payoff value z^*.
Initialize $z^* = M$.
Let $L = V \setminus \{t\}$.
Perform a BFS traversal from node t in the reverse direction to sort the elements of L.
while $L \neq \emptyset$ **do**
    Select the first element v from L.
    Apply Algorithm 2 to find a path P from s to v, a modified capacity vector $\tilde{\mathbf{u}}$, the capacity of P with respect to $\tilde{\mathbf{u}}$, and the total cost R_1 of directing the attacker towards v.
    for each node v' in P **do**
        Let P' be the subpath of P from s to v' and R_1' be its length.
        if $R > R_1'$ **then**
            Set $\bar{R} = R - R_1'$.
            Apply Algorithm 3 to compute z_{max}^1.
            if $z^* > z_{max}^1$ **then**
                Set $z^* = z_{max}^1$ and $\tilde{\mathbf{u}}^* = \tilde{\mathbf{u}}$.
            end if
        end if
        Remove v' from L.
    end for
end while

Algorithm 5 maintains a worst-case time complexity comparable to Algorithm 4. This is primarily due to the dominance of repeated calls to Algorithm 2 and breadth-first search (BFS) traversals within both algorithms. While the theoretical worst-case performance remains unchanged, empirical observations indicate that Algorithm 5 generally outperforms Algorithm 4. This improvement can be attributed to the selective computation of shortest paths, which reduces unnecessary calculations. The subsequent section will present computational results supporting this performance enhancement.

4. Computational experiments

This section presents a comparative analysis of the computational performance of Algorithms 4 and 5. To conduct this analysis, we implemented both algorithms in Python 3.10.0, utilizing the NetworkX 2.6.3 library for network manipulation and visualization. Experiments were performed on a Windows 10 64-bit laptop equipped with an Intel Core i7-8550U CPU (1.80 GHz - 2.00 GHz) and 8 GB of RAM.

To assess algorithm efficiency, we generated random instances of various sizes and densities. For each instance size and density combination, ten network instances were created, and the average running time of each algorithm was recorded.

To evaluate the performance of Algorithms 4 and 5, we employed two graph models: ladder graphs and random binomial graphs.

Ladder graphs, denoted by L_k, are planar undirected graphs with $n = 2k$ nodes and $m = 3k - 2$ edges. Constructed as the Cartesian product of two path graphs (one with a single edge), ladder graphs are inherently acyclic when directed from lower to higher node indices. To convert it to a directed one, each edge (i, j) with $i < j$ is oriented from i to j. Note that the graph obtained by this way is an acyclic graph. So, it is easy to find a shortest path in it.

Random binomial graphs, introduced by Erdös and Rényi [16], are characterized by the number of nodes, n, and an edge probability, $p \in [0, 1]$. Each potential edge is included in the graph independently with probability p. To create directed binomial graphs, edges are randomly oriented, resulting in potentially cyclic structures.

To construct random instances of the DMCPI problem, data are generated according to the following uniform random distributions:

$$\begin{aligned}
\underline{u}_{ij} &\sim U(0, n) && \forall (i, j) \in A, \\
u_{ij} - \underline{u}_{ij} &\sim U(0, n) && \forall (i, j) \in A, \\
r_{ij} &\sim U(0, n) && \forall (i, j) \in A, \\
f_{ij} &\sim U(0, n) && \forall (i, j) \in A, \\
R &\sim U(0, n^2).
\end{aligned}$$

Table 1 presents the computational results for various ladder graphs. Due to the acyclic nature of ladder graphs and the fact that a shortest path from the source to the

Table 1. Comparison of Running Times for Algorithms 4 and 5 on Various Ladder Graphs

k	n	m	Algorithm 4			Algorithm 5			Ratio
			Iter.	Time (sec.)	Obj.	Iter.	Time (sec.)	Obj.	of times
100	200	298	100	0.078	4.72	3.5	0.011	4.72	7.091
200	400	598	200	0.328	7.89	2.8	0.031	7.89	10.581
300	600	898	300	0.734	10.34	3.2	0.063	10.34	11.651
400	800	1198	400	1.327	15.67	3.2	0.1093	15.67	12.141
500	1000	1498	500	2.108	12.45	2.9	0.187	12.45	11.273
600	1200	1798	600	3.281	8.12	2.9	0.289	8.12	11.353
700	1400	2098	700	4.655	13.89	3.4	0.468	13.89	9.947
800	1600	2398	800	8.457	6.54	3.2	0.689	6.54	12.274
900	1800	2698	900	10.812	11.23	3.2	0.896	11.23	12.067
1000	2000	2998	1000	13.912	9.98	3	1.13	9.98	12.312

Table 2. Comparison of Running Times for Algorithms 4 and 5 Across Various Binomial Graphs with Different Values of p

n	p	m	Algorithm 4			Algorithm 5			Ratio
			Iter.	Time (sec.)	Obj.	Iter.	Time (sec.)	Obj.	of times
100	0.3	1485	100	13.365	80.8703	47.6	6.633	80.8703	2.014
100	0.4	1980	100	19.560	87.0290	48.2	9.640	87.0290	2.0289
100	0.5	2475	100	32.716	113.4869	48.8	17.514	113.4869	1.868
100	0.6	2970	100	47.409	105.0200	49.8	25.369	105.0200	1.868
100	0.7	3465	100	65.599	110.8032	51.8	34.500	110.8032	1.901
100	0.8	3960	100	85.714	130.7976	54	46.857	130.7976	1.829
100	0.9	4455	100	115.479	126.0424	50.4	59.397	126.0424	1.944
100	1	4950	100	127.692	128.3876	49.2	65.338	128.3876	1.954

last node encompasses approximately $n/2$ shortest paths to other nodes, Algorithm 5 exhibits significantly superior running time compared to Algorithm 4. The last column of Table 1 reveals that the running time of Algorithm 4 is approximately 10 times greater than that of Algorithm 5.

Tables 2 and 3 present the computational results for randomly generated binomial graphs. In both tables, the running time of Algorithm 4 is approximately twice that of Algorithm 5. Importantly, for every instance in all three tables, the objective function values obtained by both algorithms are identical. This consistency confirms that both algorithms correctly locate the optimal solution and that their implementations are accurate.

While both algorithms effectively address the DMCPI problem, the proposed Algorithm 5 provides a significant computational advantage, especially for structured graph topologies such as ladder graphs. These results highlight the potential of algorithm-specific optimizations tailored to graph characteristics for enhancing the efficiency of solving the DMCPI problem.

5. Concluding remarks

In conclusion, this paper examined the dynamics of a sequential Stackelberg game in the context of the dynamic maximum capacity path interdiction problem, characterized by asymmetric information among the players. Specifically, it was found that while the defender has complete knowledge of the network structure, the attacker op-

Table 3. Comparison of Running Times for Algorithms 4 and 5 Across Various Binomial Graphs with Different Values of n

n	p	m	Algorithm 4			Algorithm 5			Ratio
			Iter.	Time (sec.)	Obj.	Iter.	Time (sec.)	Obj.	of times
50	0.5	612.5	50	1.493	41.8198	23.6	0.852	41.8198	1.750
75	0.5	1387.5	75	9.457	88.1253	35.8	4.255	88.1253	2.222
100	0.5	2475	100	28.305	94.1948	50.8	14.876	94.1948	1.902
125	0.5	3875	125	70.390	157.3706	59.6	35.579	157.3706	1.978
150	0.5	5587.5	150	177.463	212.3628	72	87.380	212.3628	2.030

erates under limitations and can access the capacities of arcs only upon reaching their starting points. This disparity in information access creates a complex landscape for analyzing strategic interactions between the defender and attacker.

To address this problem, we proposed a polynomial-time algorithm that effectively utilizes the fact that the objective is influenced solely by a single arc on the desired path. Additionally, we introduced an improved version of the algorithm that showed better performance in computational experiments and demonstrated its potential for broader applications in network security strategies.

Looking ahead, several avenues for future research emerge from this study. It is crucial to investigate scenarios where both players possess complete information, as this could provide insights into different strategic behaviors and outcomes. Furthermore, extending this framework to cover other network optimization problems, such as spanning tree issues and maximum flow challenges, presents valuable opportunities to deepen our understanding of network security dynamics across various contexts.

By continuing to refine and adapt our approaches to these complex interactions, we can contribute to more effective defense strategies for critical network infrastructures and, consequently, enhance overall security resilience.

Acknowledgements: We would like to extend our sincerest gratitude to the editor and the referees who dedicated their time and expertise to review and provide valuable feedback on this manuscript. Their insightful comments, constructive criticism, and thorough review have significantly contributed to improving the quality and clarity of this paper.

Conflict of Interest: The authors declare that they have no conflict of interest regarding the publication of this paper. They affirm that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest. Furthermore, any affiliations or funding sources that contributed to the research were disclosed, ensuring transparency in the research process. The integrity and objectivity of the work presented in this paper remain intact, free from outside influence or personal bias.

Data Availability: Data sharing is not applicable to this article as no datasets were generated or analyzed during the current study.

References

[1] A. Abdolahzadeh, M. Aman, and J. Tayyebi, *Minimum st-cut interdiction problem*, Computers and Industrial Engineering **148** (2020), 106708. https://doi.org/10.1016/j.cie.2020.106708.

[2] _____, *Bottleneck spanning tree interdiction problem with fixed and linear costs*, Bulletin of the Transilvania University of Brasov. Series III: Mathematics and Computer Science **3** (2023), no. 1, 185–198. https://doi.org/10.31926/but.mif.2023.3.65.1.14.

[3] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin, *Network Flows: Theory, Applications and Algorithms*, Prentice-Hall, Englewood Cliffs, New Jersey, USA, 1993.

[4] İ. Akgün, B.Ç. Tansel, and R.K. Wood, *The multi-terminal maximum-flow network-interdiction problem*, Eur. J. Oper. Res. **211** (2011), no. 2, 241–251. https://doi.org/10.1016/j.ejor.2010.12.011.

[5] U. Anjarassuk and J. Linderoth, *Reformulation and sampling to solve a stochastic network interdiction problem*, Networks **52** (2008), no. 3, 120–132. https://doi.org/10.1002/net.20237.

[6] N. Assimakopoulos, *A network interdiction model for hospital infection control*, Computers in Biology and Medicine **17** (1987), no. 6, 413–422. https://doi.org/10.1016/0010-4825(87)90060-6.

[7] E. Azizi and A. Seifi, *Solution algorithms for shortest path network interdiction with symmetric and asymmetric information*, Int. J. Sys. Sci.: Operations and Logistics **10** (2023), no. 1, 2247964. https://doi.org/10.1080/23302674.2023.2247964.

[8] _____, *Shortest path network interdiction with incomplete information: a robust optimization approach*, Ann. Oper. Res. **335** (2024), no. 2, 727–759. https://doi.org/10.1007/s10479-023-05350-1.

[9] M. Backhaus and G. Schaefer, *Towards the complexity of the widest path problem in hybrid multi-channel WMNs*, 2020 16th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), IEEE, 2020, pp. 378–381.

[10] H. Bayrak and M.D. Bailey, *Shortest path network interdiction with asymmetric information*, Networks **52** (2008), no. 3, 133–140. https://doi.org/10.1002/net.20236.

[11] H. Bigdeli, S.M.S. Mirdamadi, and J. Tayyebi, *A meta-heuristic method for maximum capacity path interdiction problem with multiple attackers*, Journal of Modeling in Engineering **20** (2022), no. 70, 133–146. https://doi.org/10.22075/jme.2022.26026.2213.

[12] J.S. Borrero, O.A. Prokopyev, and D. Sauré, *Sequential shortest path interdiction with incomplete information*, Decis. Anal. **13** (2016), no. 1, 68–98. https://doi.org/10.1287/deca.2021.0426.

[13] G. Brown, M. Carlyle, J. Salmerón, and K. Wood, *Defending critical infrastruc-*

ture, Interfaces **36** (2006), no. 6, 530–544.
https://doi.org/10.1287/inte.1060.0252.

[14] K.J. Cormican, D.P. Morton, and R.K. Wood, *Stochastic network interdiction*, Oper. Res. **46** (1998), no. 2, 184–197.
https://doi.org/10.1287/opre.46.2.184.

[15] S. Dong, K. Abbas, and R. Jain, *A survey on distributed denial of service (DDoS) attacks in SDN and cloud computing environments*, IEEE Access **7** (2019), 80813–80828.
https://doi.org/10.1109/ACCESS.2019.2922196.

[16] P. Erdõs and A. Rényi, *On the evolution of random graphs*, Selected Papers of Alfréd Rényi **2** (1976), 482–525.

[17] F. Fang, S. Liu, A. Basak, Q. Zhu, C.D. Kiekintveld, and C.A. Kamhoua, *Introduction to game theory*, Game Theory and Machine Learning for Cyber Security (2021), 21–46.

[18] R. Hemmecke, R. Schultz, and D.L. Woodruff, *Interdicting stochastic networks with binary interdiction effort*, Network interdiction and stochastic integer programming, vol. 22, Springer, Boston, MA, 2003, pp. 69–84.
https://doi.org/10.1007/0-306-48109-X_4.

[19] L. Hu and L. Zhang, *Real-time internet traffic identification based on decision tree*, World Automation Congress 2012, IEEE, June 2012, pp. 1–3.

[20] D. Huang, Z. Mao, K. Fang, and L. Chen, *Solving the shortest path interdiction problem via reinforcement learning*, International Journal of Production Research **61** (2023), no. 1, 31–48.
https://doi.org/10.1080/00207543.2021.2002962.

[21] K.T. Kennedy, R.F. Deckro, J.T. Moore, and K.M. Hopkinson, *Nodal interdiction*, Math. Comput. Modell **54** (2011), no. 11-12, 3116–3125.
https://doi.org/10.1016/j.mcm.2011.07.041.

[22] L. Khachiyan, E. Boros, K. Borys, K. Elbassioni, V. Gurvich, G. Rudolf, and J. Zhao, *On short paths interdiction problems: Total and node-wise limited interdiction*, Theory of Comput. Syst. **43** (2008), no. 2, 204–233.
https://doi.org/10.1007/s00224-007-9025-6.

[23] F. Ljunggren, K. Persson, A. Peterson, and C. Schmidt, *Railway timetabling: a maximum bottleneck path algorithm for finding an additional train path*, Public Transp. **13** (2021), no. 3, 597–623.
https://doi.org/10.1007/s12469-020-00253-x.

[24] B.J. Lunday and H.D. Sherali, *A dynamic network interdiction problem*, Informatica **21** (2010), no. 4, 553–574.
https://doi.org/10.15388/Informatica.2010.305.

[25] M. Mahmoodjanloo, S.P. Parvasi, and R. Ramezanian, *A tri-level covering fortification model for facility protection against disturbance in r-interdiction median problem*, Computers and Industrial Engineering **102** (2016), 219–232.
https://doi.org/10.1016/j.cie.2016.11.004.

[26] A. Mohammadi and J. Tayyebi, *Maximum capacity path interdiction problem with fixed costs*, Asia-Pac. J. Oper. Res. **36** (2019), no. 04, 1950018.

https://doi.org/10.1142/S0217595919500180.

[27] D.P. Morton, F. Pan, and K.J. Saeger, *Models for nuclear smuggling interdiction*, IIE Transactions **39** (2007), no. 1, 3–14. https://doi.org/10.1080/07408170500488956.

[28] H. Nguyen-Thu, J. Tayyebi, K.T. Nguyen, and N.T. Luan, *The quickest root-leaf interdiction problem on tree networks*, Discrete Appl. Math. **368** (2025), 91–104. https://doi.org/10.1016/j.dam.2025.02.020.

[29] O. Osanaiye, K.K.R. Choo, and M. Dlodlo, *Distributed denial of service (DDoS) resilience in cloud: Review and conceptual cloud DDoS mitigation framework*, Journal of Network and Computer Applications **67** (2016), 147–165. https://doi.org/10.1016/j.jnca.2016.01.001.

[30] S.I.Z. Punla-Green, J.E. Mitchell, J.L. Gearhart, W.E. Hart, and C.A. Phillips, *Shortest path network interdiction with asymmetric uncertainty*, Networks **83** (2024), no. 3, 605–623. https://doi.org/10.1002/net.22208.

[31] A.P. Punnen, *A linear time algorithm for the maximum capacity path problem*, European J. Oper. Res. **53** (1991), no. 3, 402–404. https://doi.org/10.1016/0377-2217(91)90073-5.

[32] S. Sadeghi, A. Seifi, and E. Azizi, *Trilevel shortest path network interdiction with partial fortification*, Computers and Industrial Engineering **106** (2017), 400–411. https://doi.org/10.1016/j.cie.2017.02.006.

[33] L. Salazar-Zendeja, *Models and algorithms for the minimum spanning tree interdiction problem*, Doctoral dissertation, Centrale Lille Institut, 2022.

[34] A. Schrijver, *Combinatorial Optimization: Polyhedra and Efficiency*, vol. 24, Springer Science and Business Media, 2003.

[35] J.A. Sefair and J.C. Smith, *Dynamic shortest-path interdiction*, Networks **68** (2016), no. 4, 315–330. https://doi.org/10.1002/net.21712.

[36] ———, *Exact algorithms and bounds for the dynamic assignment interdiction problem*, Naval Research Logistics (NRL) **64** (2017), no. 5, 373–387. https://doi.org/10.1002/nav.21753.

[37] J.C. Smith and Y. Song, *A survey of network interdiction models and algorithms*, European J. Oper. Res. **283** (2020), no. 3, 797–811. https://doi.org/10.1016/j.ejor.2019.06.024.

[38] R. Steinrauf, *A network interdiction model*, Master's thesis, Naval Postgraduate School, Monterey, CA, 1991.

[39] K.M. Sullivan, D.P. Morton, F. Pan, and J. Cole Smith, *Securing a border under asymmetric information*, Naval Research Logistics (NRL) **61** (2014), no. 2, 91–100. https://doi.org/10.1002/nav.21567.

[40] J. Tayyebi, A.M. Deaconu, H. Bigdeli, and M. Niksirat, *Shortest path interdiction problem with convex piecewise-linear costs*, Comput. Appl. Math. **42** (2023), no. 7, 309. https://doi.org/10.1007/s40314-023-02445-0.

[41] J. Tayyebi, A. Mitra, and J.A. Sefair, *The continuous maximum capacity path interdiction problem*, European J. Oper. Res. **305** (2023), no. 1, 38–52. https://doi.org/10.1016/j.ejor.2022.05.028.

[42] A. Washburn and R.K. Wood, *Two-person zero-sum games for network interdiction*, Oper. Res. **43** (1994), no. 2, 243–251. https://doi.org/10.1287/opre.43.2.243.

[43] N. Wei, J.L. Walteros, and F.M. Pajouh, *Integer programming formulations for minimum spanning tree interdiction*, INFORMS Journal on Computing **33** (2021), no. 4, 1461–1480. https://doi.org/10.1287/ijoc.2020.1018.

[44] R.K. Wood, *Deterministic network interdiction*, Math. Comput. Model. **17** (1993), no. 2, 1–18. https://doi.org/10.1016/0895-7177(93)90236-R.

[45] J. Yang, J.S. Borrero, O.A. Prokopyev, and D. Sauré, *Sequential shortest path interdiction with incomplete information and limited feedback*, Decis. Anal. **18** (2021), no. 3, 218–244.