European Journal of Operational Research xxx (xxxx) xxx

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

# The continuous maximum capacity path interdiction problem

Javad Tayyebi [a], Ankan Mitra [b], Jorge A. Sefair [b,*]

[a] Department of Industrial Engineering, Birjand University of Technology, Iran
[b] School of Computing and Augmented Intelligence, Arizona State University, USA

ARTICLE INFO

*Article history:*
Received 21 July 2021
Accepted 14 May 2022
Available online xxx

*Keywords:*
Networks
Maximum capacity path
Network interdiction
Stackelberg games
Newton's method
Interdiction games

ABSTRACT

This paper studies the continuous maximum capacity path interdiction problem, where two players, user and interdictor, compete in a capacitated network. The user wants to send the maximum possible amount of flow through a path, whose capacity is given by the minimum capacity among its arcs. The budget-constrained interdictor decreases arc capacities by any continuous amount to reduce the quality of the user's chosen path. We present an efficient algorithm based on a discrete version of the Newton's method, which helps us solve the problem in polynomial time. We also prove that the problem can be transformed into a zero-sum game, which has always a pure Nash equilibrium point. We demonstrate the performance of our algorithm over a set of randomly generated networks.



## 1. Introduction

The maximum capacity path problem (MCPP), also known as *widest path* problem, consists of finding a maximum capacity path between two given nodes in a network. The capacity of any given path is determined by the minimum capacity among its arcs. Because of its flow-based structure, this problem arises in many real-world applications from different domains. In telecommunication networks, packets of information must be routed between nodes subject to a network's limited transmission capacity. A transmission path consists of a sequence of links, each of which has a given available bandwidth. To route a packet, transmission mechanisms (i.e., protocols) determine the path for each packet while at the same time enforce a quality of service level, which is typically given by a guaranteed transmission bandwidth. One example is the Multiprotocol Label Switching (MPLS), which routes packets between two nodes (i.e., ingress and egress routers) through a path that is determined at the entry node (i.e., a label switched path) (Medhi & Ramasamy, 2017). This protocol requires a minimum available bandwidth to send a packet along a path (Kar, Kodialam, & Lakshman, 2000), where the path's bandwidth corresponds to the minimum bandwidth among its links. A similar situation occurs in the design of reliable data transmission paths, where the quality of a transmission path is determined by the minimum transmission reliability (or quality) across the links in the path (Ramaswamy, Orlin, & Chakravarti, 2005; Tragoudas, 2001). Other applications of MCPP include the routing of single service units (e.g., police, firefighters), where the quality of a route is determined by the maximum distance (or response time) to a potential service location (Berman & Handler, 1987). The MCPP also arises in single-winner election methods, where pairwise candidate comparisons rely on sequences (i.e., path) of intermediate candidates and the smallest number of voters in the sequence of comparisons (Schulze, 2011). The maximum capacity path problem is also a sub-problem of other higher-level problems such as the $k$-splittable flow problem (Baier, Köhler, & Skutella, 2005), the maximum flow problem (Ahuja, Magnanti, & Orlin, 1993; Edmonds & Karp, 1972), and the quickest path problem (Clímaco, Pascoal, Craveirinha, & Captivo, 2007; Martins & Santos, 1997), among others.

Multiple algorithms exist to solve MCPP, including modified versions of shortest path algorithms like Dijkstra's algorithm (in $O(m+n\log n)$ time), scaling procedures (in $O(\min\{m+n\log n, m\log_n W\})$ time), and recursive algorithms (in $O(m)$ time), where $n$ is the number of nodes, $m$ is the number arcs, and $W$ is the maximum arc cost in the network (Gabow, 1983; Pollack, 1960; Punnen, 1991). Related problems include the weighted minimax flow, which is to minimize the maximum value of arc weights multiplied by their flow while preserving a solution of maximum flow (Ichimori, Ishii, & Nishida, 1981), and other variants seeking for a path of maximum capacity between any one node to any other node (Pollack, 1960) or between *all* node pairs (Hu, 1961).

In this paper, we study the continuous maximum capacity path interdiction problem (CMCPIP). This problem consists of two players, *user* and *interdictor*, that compete in a directed and capacitated network. The user solves an MCPP, whereas the

* Corresponding author.
*E-mail addresses:* javadtayyebi@birjand.ac.ir (J. Tayyebi), amitra16@asu.edu (A. Mitra), jorge.sefair@asu.edu (J.A. Sefair).

https://doi.org/10.1016/j.ejor.2022.05.028
0377-2217/

budget-constrained interdictor decreases arc capacities to reduce the quality of the user's chosen path. The interdictor knows the cost of reducing a unit (or fraction) of each arc's capacity and is allowed to reduce them in any amount, as long as the total cost of its actions satisfies a budget constraint. We assume that information is perfect, meaning that both agents have complete knowledge of the network topology and parameters. The agents interact in a Stackelberg game fashion (von Stackelberg, 1952), where the interdictor plays first by reducing arc capacities, and then the user determines its maximum capacity path after observing the interdictor's actions. Applications of the CMCPIP include information transmission planning (e.g., telecommunication networks using MPLS) in the presence of a proactive adversary (i.e., interdictor), whose intention is to reduce the bandwidth or reliability of a subset of links. We refer the reader to Aiello, Kushilevitz, Ostrovsky, & Rosén (2000); Bhavathankar, Chatterjee, & Misra (2017); Xue & Nahrstedt (2004) and Mageswari & Baulkani (2020) for related works on packet routing in the presence of adversaries.

In CMCPIP, the interdictor may represent a hostile agent (i.e., a jammer) or a network failure event that reduces arc capacities. Because the user competes against an interdictor that acts optimally, this game describes a pessimistic vision of the possible maximum capacity path. If the interdictor adopts a sub-optimal strategy, then it is possible to find a path with a larger capacity than that provided by CMCPIP. Under the assumption that the interdictor plays first in this two-stage Stackelberg setting, this game guarantees that in the presence of an adversary equipped with a given disruption budget, it is possible to send an amount of flow of *at least* the optimal objective function of CMCPIP. Variations in the interdictor's budget help modeling different scenarios for the uncertain network attacks and their corresponding impact on the maximum capacity path. As a result, the role of the interdictor budget is to acknowledge that some arcs capacities may be reduced but neither many of them at the same time nor by drastic amounts, which is a concept that also appears in the budgeted uncertainty paradigm from robust optimization (Bertsimas & Sim, 2003, 2004).

Unlike most of the interdiction works available in the literature (see Smith & Song, 2020 for a survey), in this paper we focus on *continuous* interdiction actions. This is not only a generalization of the existing methods (e.g., Mohammadi & Tayyebi, 2019), but also a more realistic approach from an application point of view. Existing models typically assume *binary* interdictor decisions, reflecting an *all-or-nothing* disruption regime when attacks are executed. This is unrealistic if the adversary has a wider range of options available when planning the disruption of a network component. The use of continuous decision variables allows us to model realistic situations in which the interdictor not only selects the location of the attacks but also their intensity (e.g., jammer strength). Moreover, continuous decisions can be used to model the unknown outcome of the interdictor's actions by admitting a range of values for the possible impact rather than a single value. In this case, the interdictor unveils the worst-case set of (continuous) actions that cause the most detrimental damage under the given budget.

The proposed CMCPIP belongs to the category of network interdiction problems. These problems are widely studied, as they are a natural way to describe the operation of a flow-based system subject to disruptions. Well known problems in this class are the maximum flow and the shortest path interdiction problem, both of which are NP-hard (Israeli & Wood, 2002; Wood, 1993). Although network interdiction problems are typically hard to solve (see Smith & Song, 2020 for a survey), in this paper we present a polynomial time algorithm for solving CMCPIP. Our algorithm consists of two phases. The first phase finds an interval of values containing the optimal CMCPIP value, whereas the second phase uses a discrete version of the Newton's method to search for the optimal solution. Methodologically speaking, related works include a binary search to find most vital arcs in a network (Ball, Golden, & Vohra, 1989), an application of the Newton's method to solve maximum flow interdiction problem (Matuschke, McCormick, Oriolo, Peis, & Skutella, 2017), and a successive minimum cut algorithm to solve maximum flow interdiction problem (McMasters & Mustin, 1970). The closest work to ours is that of Mohammadi & Tayyebi (2019), which assumes that the cost of interdicting an arc is fixed regardless of the attack intensity, i.e., the interdiction actions are discrete and the interdictor follows an *all-or-nothing* rationale.

We transform CMCPIP into a zero-sum game in normal form and prove that such game always has a pure Nash equilibrium. This allows us to devise optimal strategies for each player. This is an important theoretical result because zero-sum normal games not always admit a pure Nash-equilibrium (Mazalov, 2014). Additionally, this transformation provides a justification for the assumption that the user has complete information of the interdictor's actions. In zero-sum normal games, players act in a pessimistic way to secure a minimum payoff, i.e., they want to maximize their minimum guaranteed payoff. Our approach is similar to the work of Washburn & Wood (1995), whose zero-sum game transformation combining network optimization and game theoretical results allows them to efficiently solve a probabilistic version of a path-selection interdiction problem.

This paper is organized as follows. In Section 2, we review the existing literature on network interdiction. Section 3 provides a bi-level programming formulation of CMCPIP in which the interdictor (outer level) variables are continuous, while the user (inner level) variables are discrete (i.e., whether an arc belongs to the path). This feature differentiates CMCPIP from some of the existing network interdiction problems, whose formulations contain continuous variables at the user level and assume integer interdictor decisions (e.g., Israeli & Wood, 2002; Wood, 1993). Indeed, this feature prevents us from using the KKT conditions and duality results to directly convert CMCPIP into a single-level problem (e.g., Allende & Still, 2013; Golden, 1978). Section 4 presents the preliminary results needed in our strongly polynomial algorithm, which is described in Section 5. Section 6 provides a reduction of CMCPIP to a zero-sum normal game, which always has a Nash equilibrium point. We present our computational experience in Section 7. Section 8 presents our concluding remarks and future work.

## 2. Literature review

Network interdiction models are widely applied, as they can be used to optimize the operation of flow-based systems under disruptions. They have been applied in a wide range of domains, including interdicting criminal in an illegal drug supply chains (Malaviya, Rainwater, & Sharkey, 2012), energy delivery (Rocco et al., 2010), nuclear smuggling (Dimitrov et al., 2008; Morton, Pan, & Saeger 2007), infection spread control (Assimakopoulos, 1987), military planning (Ghare, Montgomery, & Turner, 1971), conservation planning (Acevedo, Sefair, Smith, Reichert, & Fletcher, 2015; Sefair, Smith, Acevedo, & Fletcher, 2017), and protecting electric power grids against terrorist attacks (Salmeron, Wood, & Baldick, 2004). Depending on the players' objectives and decisions, various types of network interdiction problems arise, including

- **Shortest path interdiction** (Fulkerson & Harding, 1977; Golden, 1978; Israeli & Wood, 2002; Sefair & Smith, 2016): The objective of the user is to find a shortest path to move between two known nodes, while the objective of the interdictor is to perturb the arc set to deteriorate the user's objective. Actions that could achieve this objective are removing or increasing the length (or cost) of a subset of arcs.
- **Maximum flow interdiction** (Akgun, Tansel, & Wood, 2011; Altner, Ergun, & Uhan, 2010; Royset & Wood, 2007; Wood, 1993): The user maximizes the flow sent from a source node to a sink

node, whereas the interdictor seeks to reduce such flow by decreasing the capacity of a subset of arcs.

- **Maximum reliability path interdiction** (Pan, 2005; Pan & Morton, 2008): The user chooses a maximum reliability path to move between two known nodes, while the interdictor decreases the overall path reliability by decreasing the individual reliability of a subset of arcs.

Interdiction problems have been formulated for other network-oriented problems including matching and assignment (Laroche, Marchetti, Martin, & Roka, 2014; Sefair & Smith, 2017; Zenklusen, 2010), minimum spanning tree (Frederickson & Solis-Oba, 1999; Zenklusen, 2015), hub design (Ghaffarinasab & Motallebzadeh, 2018; Ramamoorthy, Jayaswal, Sinha, & Vidyarthi, 2018), and facility location (Church & Scaparra, 2007; Liberatore, Scaparra, & Daskin, 2011, 2012). For a comprehensive review of network interdiction problems, we refer the reader to the survey papers Smith, Prince, & Geunes (2013) and Smith & Song (2020).

Robust optimization problems are closely related to interdiction problems. In some variants, the order in which the game is played is interchanged, meaning that the problem is solved from the perspective of the user (Matuschke et al., 2017). That is, the user makes its decisions first and then the interdictor, which represents the uncertainty in the input parameters, perturbs a subset of parameters seeking the maximum possible deterioration of the user problem's objective value. Examples of constraints limiting the interdictor when choosing such perturbations are the ellipsoidal or polyhedral uncertainty sets in Bertsimas & Sim (2003, 2004). Unlike some network interdiction problems, Bertsimas & Sim (2003) show that the robust counterpart of some polynomially solvable zero-one optimization problems remains polynomially solvable. Further, they present an algorithm for robust network flow problems to obtain an optimal solution by solving a polynomial number of minimum cost flow problems. In a related area, Alves Pessoa, Di Puglia Pugliese, Guerriero, & Poss (2015) prove that the robust constrained shortest path problem under resource uncertainty is strongly NP-hard, which can be solved in pseudo-polynomial time whenever the uncertainty set is determined only by capacity constraints.

The closest work to ours is that of Mohammadi & Tayyebi (2019) on the MCPIP, which assumes that the cost of interdicting an arc is fixed regardless of the attack intensity. This limiting assumption implies that the interdictor decreases arc capacities by a fixed (known) amount if an arc is attacked, which is unrealistic in some applications. We propose a more general model that allows continuous interdiction decisions with costs that are proportional to the intensity of the attacks. These additional problem features pose new algorithmic challenges that prevent us from using the simpler algorithmic framework in Mohammadi & Tayyebi (2019) in its current form. In this work, we characterize an optimal solution to the continuous interdiction problem and exploit its features (e.g., the interdictor's budget constraint is always binding). These elements provide the theoretical background to prove that the algorithm in Mohammadi & Tayyebi (2019) can be extended to the continuous case. To our knowledge, this is the first work studying the maximum capacity path interdiction problem with continuous variables for the interdictor and interdiction costs proportional to the intensity of the attacks.

## 3. Problem definition and formulation

In this section, we formally define CMCPIP and formulate it as a bi-level optimization problem. To this end, we first describe the maximum capacity path problem. Let $G = (V, A, \mathbf{c})$ be a directed graph in which $V = \{1, 2, \ldots, n\}$ is the node set, $A$ is the arc set such that $|A| = m$, and $\mathbf{c}$ is an $m$-dimensional vector of arc capacities. The capacity of arc $(i, j) \in A$ is given by parameter $c_{ij} \geq 0$. The network contains two distinguished nodes, $s$ and $t$, representing the origin and the destination of a maximum capacity path, respectively. We refer to a path from $s$ to $t$ as an $st$-path. Moreover, we define the capacity of any $st$-path $P$ as the minimum capacity of its arcs, i.e., $\min_{(i,j)\in P} c_{ij}$. Using these definitions, the maximum capacity path problem is to find an $st$-path of maximum capacity. This problem can be formulated as the combinatorial optimization problem

$$\max_{P\in\mathcal{P}} \min_{(i,j)\in P} c_{ij}, \tag{1}$$

where $\mathcal{P}$ is the set of all $st$-paths in $G$. We formulate Problem (1) as a zero-one linear programming problem with binary decision variables $x_{ij}$ for each $(i, j) \in A$. Variable $x_{ij}$ is equal to one if arc $(i, j) \in A$ is chosen for the maximum capacity path and is equal to zero otherwise. Using these definitions, the MIP formulation in (2) describes MCPP.

$$\max z = c_{min} \tag{2a}$$

$$\text{s.t.} \quad c_{ij} + M_{ij}(1 - x_{ij}) \geq c_{min}, \quad \forall (i, j) \in A \tag{2b}$$

$$\sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = \begin{cases} 1 & i = s \\ 0 & i \notin \{s, t\}, \\ -1 & i = t \end{cases} \quad \forall i \in V \tag{2c}$$

$$x_{ij} \in \{0, 1\} \quad \forall (i, j) \in A, \tag{2d}$$

where $M_{ij} = \max_{(k,\ell)\in A}\{c_{k\ell}\} - c_{ij}$ acts like a big-M parameter. The objective function in (2a) and the Constraints in (2b) guarantee that $c_{\min} = \min_{(i,j)\in A}\{c_{ij} : x_{ij} = 1\}$. Constraints (2c) are typical flow balance requirements and Constraints (2d) enforce the binary nature of the decision variables. We note that alternative formulations to the MIP in (2) may exist. However, we are unaware of any compact linear programming formulation for MCPP, as in the case for other combinatorial optimization problems (e.g., minimum spanning tree (Magnanti & Wolsey, 1995)).

The MIP in (2) can be solved using traditional integer programming methods, such as branch-and-bound and cutting planes (Schrijver, 1998), or zero-one programming approaches such as Balas' additive method (Balas, 1965). However, such approaches have exponential complexity in the worst case. Due to its structure, the MCPP described in (2) is solvable by efficient polynomial-time algorithms (Medhi & Ramasamy, 2017).

The MIP formulation in (2) describes the user's problem only. To formulate the interdiction problem, we introduce the continuous decision variables $d_{ij} \in [0, c_{ij}]$ for each $(i, j) \in A$. These variables capture the reduction in the capacity of arc $(i, j) \in A$ induced by the interdictor actions. The interdictor is subject to the budget constraint $\sum_{(i,j)\in A} w_{ij} d_{ij} \leq W$, where $w_{ij} \geq 0$ is the cost of decreasing $c_{ij}$ by one unit and $W \geq 0$ is the available budget. Using these definitions, we formulate CMCPIP as the bi-level programming problem in (3).

$$\min \ z \tag{3a}$$

$$\text{s.t.} \quad \sum_{(i,j)\in A} w_{ij} d_{ij} \leq W \tag{3b}$$

$$0 \leq d_{ij} \leq c_{ij} \quad \forall (i, j) \in A, \tag{3c}$$

$$z = \max \ c_{min} \tag{3d}$$

$$\text{s.t.} \quad c_{ij} - d_{ij} + M_{ij}(1 - x_{ij}) \geq c_{min}, \quad \forall (i, j) \in A \tag{3e}$$

$$\sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = \begin{cases} 1 & i = s \\ 0 & i \notin \{s, t\}, \\ -1 & i = t \end{cases} \quad \forall i \in V \tag{3f}$$

$$x_{ij} \in \{0, 1\} \quad \forall (i, j) \in A \tag{3g}$$

The interdictor's objective function in (3a) seeks to reduce the user's objective function subject to the budget constraint (3b) and the bounds on the $d$-variables enforced by (3c). Constraints (3d)–(3g) correspond to the user's problem and describe the MCPP in (2) with updated arc capacities given the interdictor's decisions. A feasible solution to Problem (3) takes the form $(\mathbf{d}, \mathbf{x})$, where $\mathbf{d}$ is feasible for the interdictor (i.e., satisfies constraints (3b) and (3c)) and $\mathbf{x}$ is optimal for the user given $\mathbf{d}$. We emphasize that the user problem in (3d)–(3g) is always feasible.

Problem (3) cannot be solved using the traditional dualize-and-combine approach (i.e., dualizing the user's problem to produce a single-level formulation) or a KKT-based reformulation because the user-level variables are binary (Allende & Still, 2013; Bazaraa, Jarvis, & Sherali, 2011). However, this bi-level problem can be transformed into a single-level integer linear problem using a *value function* reformulation and then solved using a branch-and-cut algorithm (Fischetti, Ljubić, Monaci, & Sinnl, 2017, 2018; Tahernejad, Ralphs, & DeNegre, 2020). Other solution approaches to solve bi-level optimization problems include decomposition techniques based on column-and-constraint generation (Zeng & An, 2014) and tailored algorithms exploiting the problem's structure (Contardo & Sefair, 2021; Xu & Wang, 2014). A compact linear programming reformulation to Problem (2) may allow the use of other solution approaches for the solution of Problem (3). However, we are unaware of the existence of such reformulation for the MCPP. Although it is possible to solve Problem (3) via existing methods, using such approaches will ignore the subjacent properties of the problem that in this particular case admit a polynomial time algorithm.

In the remainder of this paper, we use the notation $G = (V, A, \mathbf{c})$ to denote a graph with node set $V$, arc set $A$, and arc capacities $\mathbf{c} = (c_{ij})_{(i,j)\in A}$. Moreover, we represent a solution to Problem (3) as $(\mathbf{d}, \mathbf{x})$, where $\mathbf{d} = (d_{ij})_{(i,j)\in A}$, which can be simplified to only $\mathbf{d}$ because $\mathbf{x}$ can be determined by solving MCPP on $G = (V, A, \mathbf{c} - \mathbf{d})$ using the MIP in (2) or any other method. In other words, the optimal objective function value corresponding to an optimal solution $\mathbf{d}^*$ to Problem (3) is the same as that obtained by solving Problem (2) with arc capacities given by $c_{ij} - d^*_{ij}$ for each arc $(i, j) \in A$.

We assume that all parameters in Problem (3) are positive integers. This is a common assumption in the network optimization literature and is not restrictive in practice (Ahuja et al., 1993). Rational numbers can be transformed into integer numbers by multiplying them by a suitably large number. Moreover, irrational numbers need to be converted into rational numbers to be stored on a computer, which allows us to transform them into integer numbers as well. We assume that *all* parameters are positive. If the problem contains some arc $(i, j)$ with $c_{ij} = 0$, then the user would never select an $st$-path including $(i, j)$ and the arc can be removed. If $w_{ij} = 0$ for some $(i, j) \in A$, then the optimal interdictor's action is to set $d_{ij} = c_{ij}$ at no cost, which means that arc $(i, j)$ can be removed from the network because its capacity becomes zero. If $W = 0$, then the interdictor cannot decrease any arc capacity and consequently, $\mathbf{d} = \mathbf{0}$ is its optimal decision.

The following example shows that Problem (3) may have a fractional optimal solution even if all parameters are integer numbers.

**Example 1.** Consider the graph shown in Fig. 1, where the numbers in parenthesis correspond to $(c_{ij}, w_{ij})$ for each arc $(i, j) \in A$. The available budget is $W = 1$ and the user travels from node 1 to node 4, so the possible maximum capacity $st$-paths are $1-2-4$, $1-3-4$, and $1-3-2-4$ with capacity $c_{\min} = 3$. If the interdictor reduces $c_{24}$ to 2, then the user will select $1-3-4$ with capacity $c_{\min} = 3$. Similarly, if the interdictor decreases $c_{13}$ from 3 to 2, then the user will select $1-2-4$ with capacity $c_{\min} = 3$. Therefore, the interdictor must divide the budget equality among both paths, changing $c_{13}$ and $c_{24}$ from 3 to 2.5, which results in the capacity of all $st$-paths being at most $c_{\min} = 2.5$. This is the optimal solution to this instance of CMCPIP.

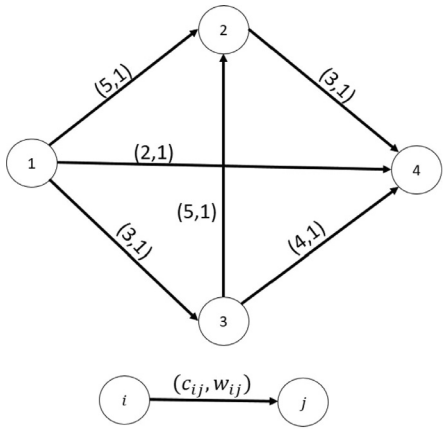



**Fig. 1.** Instance of CMCPIP with total budget $W = 1$.

## 4. Problem properties

In this section, we derive some preliminary results that serve as the foundation of our algorithm and that will be used throughout the paper. Although the user (inner) level feasible region in Problem (3) is discrete, we prove that the set of objective values corresponding to all feasible solutions (i.e., the *objective region*) is a closed interval. For a given maximum capacity path $P$ in $G = (V, A, \mathbf{c})$, let $z_{\max} = \min_{(i,j)\in P}\{c_{ij}\}$ and $\mathbf{x}_{\max} = (x_{ij})_{(i,j)\in A}$, where $x_{ij} = 1$ if $(i, j) \in P$ and $x_{ij} = 0$ otherwise. Note that $z_{\max}$ is the same objective function value of $(\mathbf{d}, \mathbf{x}) = (\mathbf{0}, \mathbf{x}_{\max})$ in Problem (3). Therefore, the objective value of any feasible solution belongs to the interval $[0, z_{\max}]$.

We use the notion of an $st$-cut in $G$, which is a minimal set of arcs whose removal from $G$ disconnects $s$ and $t$, i.e., $G$ has no $st$-paths. The removal of all arcs in an $st$-cut will partition $G$ into two sub-graphs with node sets $S$ and $\bar{S}$, one containing $s$ and the other containing $t$, respectively (Ahuja et al., 1993). We denote an $st$-cut by $C = [S, \bar{S}]$ where $s \in S$ and $t \in \bar{S}$. We refer to an arc $(i, j)$ with $i \in S$ and $j \in \bar{S}$ as a forward and an arc $(i, j)$ with $i \in \bar{S}$ and $j \in S$ as a backward arc of $C$. The sets $(S, \bar{S})$ and $(\bar{S}, S)$ denote the sets of forward and backward arcs, respectively, in an $st$-cut $C = [S, \bar{S}]$. The weight of an $st$-cut is the sum of weights of its *forward* arcs. Moreover, a minimum $st$-cut is an $st$-cut of minimum weight. We refer the reader to Ahuja et al. (1993) for further details on finding minimum $st$-cuts. Property 1 states the relationship between an $st$-cut and the optimal solution to CMCPIP.

**Property 1.** *The set of modified arcs in an optimal solution belongs to an st-cut.*

Suppose that a feasible solution to Problem (3) exists with objective value equal to $z$. This means that the capacity of any $st$-path is at most $z$, i.e., for every $st$-path there exists at least one arc whose initial or modified capacity is less than or equal to $z$. Because the initial capacity of some $st$-paths may be greater than $z$, the interdictor has to reduce the capacity of at least one arc in those $st$-paths to $z$. In this case, it is sufficient to modify the capacity of exactly one arc in each of these $st$-paths. As a result, the set of modified arcs in any optimal solution belongs to an $st$-cut. Moreover, it is nonstrategic for the interdictor to incur in the additional cost of reducing the initial capacity of an arc $(i, j)$ with $c_{ij} > z$ to any value strictly less than $z$. The following definition and theorem formalize the arguments in Property 1.

**Definition 1.** For any *st*-cut $C$ and any $z \in [0, z_{\max}]$, we define vector $\mathbf{d}^{(C,z)} = \left(d_{ij}^{(C,z)}\right)_{(i,j)\in A}$ as

$$d_{ij}^{(C,z)} = \begin{cases} c_{ij} - z & \text{If } (i,j) \in C \text{ and } c_{ij} > z \\ 0 & \text{Otherwise} \end{cases}, \quad \forall (i,j) \in A. \tag{4}$$

The following properties are immediate.

**Property 2.** *If $\mathbf{d}^{(C,z)}$ satisfies the budget constraint (3b), then $\mathbf{d}^{(C,z')}$ also satisfies it for every $z' \in [z, z_{\max}]$.*

**Property 3.** *Vector $\mathbf{d}^{(C,z)}$ is a feasible solution to Problem (3) if and only if it satisfies the budget constraint (3b).*

**Theorem 1.** *Consider an instance of Problem (3) and suppose that a feasible solution exists with objective value equal to z. Then, there exists an st-cut C such that $\mathbf{d}^{(C,z)}$ satisfies Constraints (3b) and (3c).*

**Proof.** Suppose that $\bar{\mathbf{d}}$ is a feasible solution to Problem (3) with objective value equal to $z$. Define the set $\bar{D} = \{(i,j) \in A : c_{ij} - \bar{d}_{ij} \le z\}$ and note that at least one member of this set determines the user's optimal path with capacity $z$. Set $\bar{D}$ contains at least one arc from each *st*-path $P \in \mathcal{P}$, where $\mathcal{P}$ is the set of all *st*-paths. Otherwise, there is an *st*-path whose capacity is more than $z$, which contradicts the optimality of the user's problem given $\bar{\mathbf{d}}$. Because $\bar{D}$ contains at least one arc from each *st*-path, it follows that it contains at least one *st*-cut. Denote any of the *st*-cuts in $\bar{D}$ by $C$ and define $\mathbf{d}^*$ as in (4). This implies that the objective value of $\mathbf{d}^*$ is also equal to $z$ and that $d_{ij}^* \le \bar{d}_{ij} \le c_{ij}$ for every $(i,j) \in A$. Using this relationship and the nonnegativity of the $w$-parameters, we obtain that

$$\sum_{(i,j)\in A} w_{ij} d_{ij}^* \le \sum_{(i,j)\in A} w_{ij} \bar{d}_{ij},$$

which means that $\mathbf{d}^*$ satisfies the budget constraint (3b). We complete the proof by noting that $\mathbf{d}^*$ is nonnegative by construction. □

From a practical point of view, Theorem 1 states that the interdictor should only focus on reducing the capacity of *st*-cuts in order to reduce the capacity of *any* of the user's *st*-paths. We leverage on this result to search for an optimal solution to Problem (3) only among the *st*-cuts of $G$. Definition 2 describes the costs incurred by the interdictor when attacking a subset of arcs in $G$ in order to achieve an objective function value equal to $z$.

**Definition 2.** For any network $G = (V, A, c)$ and any $z \in [0, z_{\max}]$, we define a cost vector $\bar{\mathbf{w}}^z = \left(\bar{w}_{ij}^z\right)_{(i,j)\in A}$ as

$$\bar{w}_{ij}^z = \begin{cases} w_{ij}(c_{ij} - z) & \text{If } c_{ij} > z \\ 0 & \text{If } c_{ij} \le z \end{cases}, \quad \forall (i,j) \in A. \tag{5}$$

The value $\bar{w}_{ij}^z$ can be seen as the cost of reducing the capacity of arc $(i,j)$ from $c_{ij}$ to $z$ when $c_{ij} > z$, which is equal to zero if $c_{ij} \le z$ because an interdiction action on arc $(i,j)$ will not improve the objective function value $z$. Using (5), we can calculate the cost of attacking any *st*-cut $C$ as $\bar{\mathbf{w}}^z(C) = \sum_{(i,j)\in C} \bar{w}_{ij}^z$. Property 4 follows directly from Definitions 1 and 2.

**Property 4.** *Vector $\mathbf{d}^{(C,z)}$ satisfies the budget constraint (3b) if and only if $\bar{\mathbf{w}}^z(C) \le W$.*

Theorem 2 states that the interdictor can reduce the capacity of all *st*-paths to $z$ as long as the cost of an *st*-cut is less than or equal to $W$.

**Theorem 2.** *Let $C^z$ be a minimum st-cut with respect to arc costs $\bar{\mathbf{w}}^z$, where $z \in [0, z_{\max}]$.*

- *If $\bar{\mathbf{w}}^z(C^z) \le W$, then $\mathbf{d}^{(C^z,z)}$ is a feasible solution to Problem (3).*
- *If $\bar{\mathbf{w}}^z(C^z) > W$, then there is no feasible solution to Problem (3) with objective value equal to z.*

**Proof.** The proof of the first part follows directly from Properties 3 and 4. To prove the second part, we show that there is no feasible solution with objective value $z$. Assume for contradiction that a feasible solution to Problem (3) with objective value $z$ exists. By Theorem 1, there exists an *st*-cut, $C$, such that $\mathbf{d}^{(C,z)}$ is a feasible solution with objective value $z$. Using the budget feasibility of $\mathbf{d}^{(C,z)}$, we obtain

$$\begin{aligned} W &\ge \sum_{(i,j)\in A} w_{ij} d_{ij}^{(C,z)} \\ &= \sum_{(i,j)\in C \cap \{(i,j):c_{ij}>z\}} w_{ij}(c_{ij} - z) \\ &= \bar{\mathbf{w}}^z(C), \end{aligned}$$

where the first and second equalities hold from the definitions of $\mathbf{d}^{(C,z)}$ and $\mathbf{w}^z$, respectively. However, the weight of any *st*-cut $C$ is at least the weight of the minimum *st*-cut $C^z$, thus $\bar{\mathbf{w}}^z(C^z) \le \bar{\mathbf{w}}^z(C) \le W$, which contradicts the assumption that $\bar{\mathbf{w}}^z(C^z) > W$. This implies that $\mathbf{d}^{(C,z)}$ is not feasible and that there is no feasible solution with objective value equal to $z$. □

Theorem 3 states an important feature of any optimal solution to Problem (3).

**Theorem 3.** *If the optimal objective function value to Problem (3) is positive, then the budget constraint is binding at any optimal solution.*

**Proof.** Assume for contradiction that $\mathbf{d}^*$ is an optimal solution with objective function value equal to $z^* > 0$ and such that $\sum_{(i,j)\in A} w_{ij} d_{ij}^* < W$. Define $D^*$ as the set of arcs whose modified capacity is positive, i.e., $D^* = \{(i,j) \in A : c_{ij} - d_{ij}^* > 0\}$, and let $\delta^* = \min_{(i,j)\in D^*}\{c_{ij} - d_{ij}^*\}$. Because $z^*$ is positive, it follows that there is at least one *st*-path whose capacity is positive. Thus, every arc on such *st*-paths belong to $D^*$ and $D^* \ne \emptyset$. Define $\mathbf{d}'$ as

$$d'_{ij} = \begin{cases} d_{ij}^* & (i,j) \notin D^* \\ d_{ij}^* + \epsilon & (i,j) \in D^* \end{cases},$$

where $\epsilon = \min\left\{\frac{W - \sum_{(i,j)\in A} w_{ij} d_{ij}^*}{\sum_{(i,j)\in D^*} w_{ij}}, \delta^*\right\} > 0$. It follows that $\mathbf{d}'$ is a feasible solution to Problem (3) because it satisfies both the budget constraint in (3b) and also $0 \le d'_{ij} \le c_{ij}$ for every $(i,j) \in A$. By construction, the capacity of any *st*-path is decreased by $\epsilon$, meaning that $\mathbf{d}'$ leads to a solution with objective function value strictly less than $z^*$, which contradicts the optimality of $\mathbf{d}^*$. □

These preliminary results lead us to reformulate Problem (3) as follows:

$$\min \quad z \tag{6a}$$

$$\text{s.t.} \ \min_{C\in\mathcal{C}} \left\{\bar{\mathbf{w}}^z(C) = \sum_{(i,j)\in C} \bar{w}_{ij}^z\right\} = W \tag{6b}$$

where $\mathcal{C}$ is the set of all *st*-cuts in $G = (V, A)$. Now, consider the function

$$\begin{aligned} f(z) &= \min_{C\in\mathcal{C}}\{\bar{\mathbf{w}}^z(C)\} - W \\ &= \min_{C\in\mathcal{C}} \left\{\sum_{(i,j)\in C} \max\{0, w_{ij}(c_{ij} - z)\} - W\right\}. \end{aligned}$$

Define the set $\mathcal{Z} = \{0\} \cup \{c_{ij} : c_{ij} \le z_{\max}, (i,j) \in A\}$ and suppose that its elements are organized in nondecreasing order such that $(0 =) z_0 \le z_1 \le \ldots \le z_k (= z_{\max})$, where $k \ge 1$ is an integer number. The following lemma states some properties of $f(z)$.

**Lemma 1.** *The function $f(z):\mathbb{R}\mapsto\mathbb{R}$ is nonincreasing and piecewise-linear. Furthermore, $f(z):[z_{l-1},z_l]\mapsto\mathbb{R}$ is a decreasing concave piecewise-linear function for any $l=1,2,\ldots,k$.*

**Proof.** Because $W$ is a constant, it follows that $f(z)+W=\min_{C\in\mathcal{C}}\{\bar{\mathbf{w}}^z(C)\}$ is the minimum of the functions $\bar{\mathbf{w}}^z(C)$. For any st-cut $C$ in $G$, the function $\bar{\mathbf{w}}^z(C)=\sum_{(i,j)\in C}\max\{0,w_{ij}(c_{ij}-z)\}$ is a convex nonincreasing piecewise-linear function on $z$.

Let $z\in[z_{l-1},z_l]$ for any $l=1,2,\ldots,k$. Define $S=\{(i,j)\in A: c_{ij}\geq z\}$, $b_C=\sum_{(i,j)\in C\cap S}w_{ij}c_{ij}-W$, and $a_C=\sum_{(i,j)\in C\cap S}w_{ij}$. Then, function $f(z)$ can be re-written in the following form.

$$\begin{aligned} f(z) &= \min_{C\in\mathcal{C}}\left\{\sum_{(i,j)\in C\cap S} w_{ij}(c_{ij}-z)-W\right\} \\ &= \min_{C\in\mathcal{C}}\{b_C-a_C z\}. \end{aligned} \tag{7}$$

The concavity and decreasing nature follow because $f(z)$ is defined in (7) as the minimum of a set of linear functions, $a_C>0$, and $a_C$ and $b_C$ are constants for $z\in[z_{l-1},z_l]$. □

Lemma 1 implies that $f(z)$ has only one root $z^*\in[0,z_{\max}]$, which is the optimal value to Problem (6). Further, Theorem 2 implies that if $z$ is an infeasible objective function value, then $f(z)>0$. Similarly, if $z$ is a feasible objective value, then $f(z)\leq 0$. The following Lemma builds on this observation.

**Lemma 2.** *For any fixed value $z\in[0,z_{\max}]$, if Problem (3) has no feasible solution with objective value z, then it contains no feasible solution whose objective value is less than z.*

**Proof.** Suppose that Problem (3) has no feasible solution with objective value $z'\in[0,z_{\max}]$ such, which implies that $f(z')>0$. Consider a solution with objective value $z''<z'$. Because $f(z)$ is a nonincreasing function, it follows that $f(z'')\geq f(z')>0$, which implies that there is no feasible solution with objective value $z''<z'$. □

Lemmas 1 and 2 form the basis of our polynomial-time algorithm, which we describe in Section 5.

## 5. Polynomial-time solution algorithm

In this section, we present an algorithm to solve Problem (3) in polynomial time. The proposed algorithm consists of two phases. The first phase uses a binary search to find an interval $[z_{k-1},z_k]$ containing the optimal objective value $z^*$. Formally, the goal is to find the smallest index $k\in\{1,2,\ldots,m\}$ so that $\mathbf{d}^{(C^{z_{k-1}},z_{k-1})}$ is infeasible while $\mathbf{d}^{(C^{z_k},z_k)}$ is feasible, where $C^{z_{k-1}}$ and $C^{z_k}$ are two minimum st-cuts in $G=(V,A,\bar{\mathbf{w}}^{z_{k-1}})$ and $G=(V,A,\bar{\mathbf{w}}^{z_k})$, respectively. Using Theorem 2, the first phase looks for an index $k$ so that the minimum st-cut cost in $G=(V,A,\bar{\mathbf{w}}^{z_{k-1}})$ is greater than $W$ but less than or equal to $W$ in $G=(V,A,\bar{\mathbf{w}}^{z_k})$. The second phase uses a discrete version of the Newton's method to find the value of $z^*$ inside the interval provided by the first phase, and obtains a corresponding st-cut, $C^*$. The condition $z^*\in[z_{l-1},z_l]$ means that there is no feasible solution with objective value $z_{l-1}$, whereas a solution with objective value $z_l$ exists. Accordingly, we find the root of $f(z)$ in $[z_{l-1},z_l]$ using Algorithm 1, which is based on a discrete version of the Newton's method (Radzik, 2013). The second phase also constructs an optimal solution to Problem (3) using $z^*$ and $C^*$. Depending on the context, in Algorithm 1 we use the notation $G=(V,A,\boldsymbol{\alpha})$ with $\boldsymbol{\alpha}=\mathbf{c}$ or $\boldsymbol{\alpha}=\mathbf{w}$ because we invoke subroutines to find a maximum capacity path using the $c$-parameters or to find a minimum st-cut with arc weights given by the $w$-parameters.

Phase I of Algorithm 1 starts in Line 1 by calculating a minimum st-cut in $G=(V,A,\mathbf{w}')$, denoted by $\tilde{C}$, where $w'_{ij}=c_{ij}w_{ij}$ for each $(i,j)\in A$. This can be done by finding the maximum flow between $s$ and $t$ in $G$ and then using the results from the classic max-flow min-cut theorem (Ahuja et al., 1993). The `If` condition in Lines 2–4 verifies whether the weight of $\tilde{C}$ is less than or equal to $W$. If so, it is feasible for the interdictor to reduce all the arc capacities in $\tilde{C}$ to zero. In this case, Line 3 calculates the optimal interdictor attack as well as its corresponding objective value. The algorithm then moves to Line 24 and terminates. If the weight of $\tilde{C}$ is more than $W$, Line 5 solves a MCPP with the initial capacities as input (i.e., $\mathbf{c}$) and obtains the maximum capacity $z_{\max}$. This can be done using any existing algorithm for the MCPP (see Medhi & Ramasamy, 2017 for widest path algorithms on directed networks). Following our mathematical development from Section 4, Line 6 creates a sorted list of $z$-values ranging from 0 to $z_{\max}$, which is used in Lines 7–16 to search for the interval $[z_{l-1},z_l]$ that contains $z^*$. To this end, Algorithm 1 uses a binary search approach that iteratively solves maximum flow problems using the $\bar{\mathbf{w}}$-weights defined in (5). If the maximum flow found in Line 10, denoted by $v$, is less than or equal to $W$, then there is a feasible solution to Problem (3) according to Theorem 2. If $v>W$, then there is no feasible solution for the interdictor and the corresponding $z_k$-value is unachievable. Both cases are used to update the interval where $z^*$ lies in the binary search. Line 12 records the index $k_U$ such that interval $[z_{k_U-1},z_{k_U}]$ contains $z^*$, as well as other parameters required in Phase II.

Phase II of Algorithm 1 begins in Line 17 by constructing the set of arcs whose capacities are greater than or equal to $\tilde{z}$, which is the upper bound of the interval containing $z^*$. The `While` loop in Lines 18–22 generates a sequence $\{\tilde{z}_k\}_{k=0,1,\ldots}$ converging to the root $z^*$, where $\tilde{z}_0=\tilde{z}$. Following the results in Radzik (2013), we calculate the $k$-th element in this sequence, $\tilde{z}_k$ as a function of $\tilde{z}_{k-1}$ by computing

$$\begin{aligned} \tilde{z}_k &= \tilde{z}_{k-1}-\frac{f(\tilde{z}_{k-1})}{f'(\tilde{z}_{k-1})} \\ &= \tilde{z}_{k-1}-\frac{b_{C^{\tilde{z}_{k-1}}}-a_{C^{\tilde{z}_{k-1}}}\tilde{z}_{k-1}}{-a_{C^{\tilde{z}_{k-1}}}} \\ &= \frac{\sum_{(i,j)\in C^{\tilde{z}_{k-1}}\cap S}w_{ij}c_{ij}-W}{\sum_{(i,j)\in C^{\tilde{z}_{k-1}}\cap S}w_{ij}}, \end{aligned} \tag{8}$$

where $C^{\tilde{z}_{k-1}}$ is a minimum st-cut in $G=(V,A,\bar{\mathbf{w}}^{\tilde{z}_{k-1}})$. Because $f(z)$ is non differentiable at its breakpoints, we calculate $f'(\tilde{z}_{k-1})$ using its right-hand derivative. Line 19 computes $\tilde{z}_k$ using (8) and Line 20 updates the corresponding set of weights $\bar{\mathbf{w}}^{\tilde{z}}$ using (5). Using these weights, Line 21 solves a minimum st-cut problem on $G=(V,A,\bar{\mathbf{w}}^{\tilde{z}})$. The `While`-loop continues until $v^*=W$, which is equivalent to finding the root of $f(z)$ as a result of Lemma 1. Note that the stopping condition also means that the interdictor depletes its budget, which is a necessary condition for optimality according to Theorem 3.

The following lemma and theorem prove that Algorithm 1 solves Problem (3) in a finite number of iterations. The correctness of Algorithm 1 is given by the results in Radzik (2013), for which Remark 1 provides more details.

**Lemma 3.** *If there is an index k such that points $(\tilde{z}_{k-1},f(\tilde{z}_{k-1}))$ and $(\tilde{z}_k,f(\tilde{z}_k))$ lie on the same line segment of $f(z)$, then $f(\tilde{z}_k)=0$, i.e., $\tilde{z}_k$ is the optimal value to Problem (3).*

**Proof.** Because $C^*=C^{\tilde{z}_{k-1}}$ in Line 21 is a minimum st-cut in $G=(V,A,\bar{\mathbf{w}}^{\tilde{z}_{k-1}})$, it follows that

$$\begin{aligned} f(\tilde{z}_{k-1}) &= \min_{C\in\mathcal{C}}\{\bar{\mathbf{w}}^{\tilde{z}_{k-1}}(C)\}-W \\ &= \bar{\mathbf{w}}^{\tilde{z}_{k-1}}(C^{\tilde{z}_{k-1}})-W \\ &= b_{C^{\tilde{z}_{k-1}}}-a_{C^{\tilde{z}_{k-1}}}\tilde{z}_{k-1}. \end{aligned}$$

By assumption, $(\tilde{z}_k, f(\tilde{z}_k))$ lies on the line segment, implying that $f(\tilde{z}_k) = b_{C^{\tilde{z}_{k-1}}} - a_{C^{\tilde{z}_{k-1}}}\tilde{z}_k$. From (8) we know that $\tilde{z}_k = \frac{b_{C^{\tilde{z}_{k-1}}}}{a_{C^{\tilde{z}_{k-1}}}}$, which results in $f(\tilde{z}_k) = 0$. □

**Theorem 4.** *Algorithm 1 solves Problem (3) in a finite number of iterations.*

**Proof.** The binary search in Phase I explores a finite set of intervals, thus it finishes in a finite number of iterations. The proof for Phase II is based on the results in Radzik (2013), where the goal is to find the root of a convex nonincreasing piecewise-linear function given by

$$h(\delta) = \max\{f(\mathbf{x}) - \delta g(\mathbf{x}) \mid \mathbf{x} \in \mathcal{X}\}, \tag{9}$$

where $\mathcal{X}$ is the domain of the $f$- and $g$-functions. It is shown that the Newton's method finishes in a finite number of iterations while visiting a sequence of increasing values. This result is analogous for a concave decreasing function like $f(z)$ defined as in (7) with $f(\mathbf{x}) = b_C$, $g(\mathbf{x}) = a_C$, $\delta = z$, and $\mathcal{X} = \mathcal{C}$ (see Lemma 1) and where the update in Line 19 using (8) results in a sequence of decreasing $\tilde{z}$-values. Because $f(z)$ contains a finite number of segments, Lemma 3 implies that Phase II terminates after a finite number of iterations. □

**Remark 1.** Radzik (2013) proved that the number of iterations of the Newton's method for finding the root of a convex nonincreasing piecewise-linear function defined as in (9), has a strongly polynomial bound given by $\mathcal{O}(p^2 \log^2 p)$, where $p$ is the dimension of vector $\mathbf{x}$. As established in Theorem 4, the Newton's method is applicable in a similar way to find the root of $f(z)$. As a result, the number of iterations of Phase II has a strongly polynomial bound given by $\mathcal{O}(m^2 \log^2 m)$, where $m$ is the number of arcs in the network. This leads to an overall time complexity of Algorithm 1 equal to $\mathcal{O}(m^2 \log^2 m \Lambda(n, m))$, where $\Lambda(n, m)$ is the complexity for finding a minimum cut in a network, which is polynomially solvable.

The value $z_{\max}$ in Line 5 can be quickly initialized with $c_{\max} = \max_{(i,j)\in A}\{c_{ij}\}$, avoiding the solution of a MCPP. This is convenient when the input network is very large because the complexity of any algorithm for MCPP is no less than $O(m)$. This alternative initialization comes at the expense of additional iterations of the binary search procedure in Lines 8–16. Remark 2 states that this modification has no effect in the algorithm's correctness and that each extra step in the binary search requires the solution of a trivial sub-problem. Our implementation uses this initialization as it reduced the solution times in the large-scale instances in our computational experiments.

**Remark 2.** Let $z_{\max}$ be the capacity of the maximum capacity path between $s$ and $t$ in $G = (V, A, \mathbf{c})$. Therefore, the maximum flow between $s$ and $t$ in $G = (V, A, \bar{\mathbf{w}}^z)$ must be zero for any $z \geq z_{\max}$ because at least one arc in every $(s, t)$ path has a modified weight of zero according to (5). Otherwise, $z_{\max}$ would not be the true maximum capacity. As a result, using $c_{\max}$ in Line 5 implies that Line 6 admits more values in the sorted list, which will be discarded when executing Line 12 for any $z_k > z_{\max}$ as the sub-problem in Line 10 returns $v = 0$. The algorithm will return the same results after this modification because at some iteration $k$ of the binary search we will have $z_k \leq z_{\max}$, which is the initialization in the original version of the algorithm.

Fig. 2 shows an example of the step-by-step execution of Algorithm 1 on a network with 6 nodes and 9 arcs. Fig. 2(a) shows the network components with arc capacities and interdiction costs next to each arc. Fig. 2(b) shows the execution of Line 1, where arc capacities are given by $w'_{ij} = w_{ij}c_{ij}$, for each $(i, j) \in A$, which is the same as using (5) with $z = 0$. Executing this line returns

**Algorithm 1** Two-phase algorithm for CMCPIP.

**Input:** Graph $G = (V, A, \mathbf{c})$ with two prescribed nodes, $s$ and $t$, interdiction cost vector $\mathbf{w}$, and budget $W$
**Output:** Optimal interdiction vector $\mathbf{d}^*$ and its corresponding optimal value $z^*$
**Phase I:**
1: Find a minimum $st$-cut $\tilde{C}$ in $G = (V, A, \mathbf{w}')$ with arc weights $w'_{ij} = w_{ij}c_{ij}$, $(i, j) \in A$
2: **if** $\sum_{(i,j)\in\tilde{C}} w'_{ij} \leq W$ **then**
3: Calculate $d^*_{ij} = \begin{cases} c_{ij} & (i,j) \in \tilde{C} \\ 0 & (i,j) \in A \setminus \tilde{C} \end{cases}$ and set $z^* = 0$. Go to Line 24
4: **end if**
5: Find a maximum capacity path in $G = (V, A, \mathbf{c})$ and set $z_{\max}$ to its capacity
6: Sort the values in $\{c_{ij} : c_{ij} \leq z_{\max}, (i, j) \in A\} \cup \{0\}$ in nondecreasing order and let $z_0 = 0 \leq z_1 \ldots \leq z_\ell = z_{\max}$ be the sorted list
7: Set $k_L = 0$ and $k_U = \ell$
8: **while** $k_U - k_L > 1$ **do**
9: Set $k = \lceil \frac{k_L + k_U}{2} \rceil$
10: Find the maximum flow value, $v$, between $s$ and $t$ in $G = (V, A, \bar{\mathbf{w}}^{z_k})$, where $\bar{\mathbf{w}}^{z_k}$ is defined by (5)
11: **if** $v \leq W$ **then**
12: Set $k_U = k$, $\tilde{z} = z_{k_U}$, $v^* = v$, and let $C^*$ be the $st$-cut corresponding to $v$ in $G = (V, A, \bar{\mathbf{w}}^{z_k})$
13: **else**
14: Set $k_L = k$
15: **end if**
16: **end while**
**Phase II:**
17: Construct set $S^* = \{(i, j) \in A : c_{ij} \geq \tilde{z}\}$
18: **while** $v^* \neq W$ **do**
19: Set $\tilde{z} = \frac{\sum_{(i,j)\in C^*\cap S^*} w_{ij}c_{ij} - W}{\sum_{(i,j)\in C^*\cap S^*} w_{ij}}$
20: Calculate $\bar{\mathbf{w}}^{\tilde{z}}$ using (5)
21: Find a minimum $st$-cut, $C^*$, in $G = (V, A, \bar{\mathbf{w}}^{\tilde{z}})$ and set $v^*$ to its weight
22: **end while**
23: Set $z^* = \tilde{z}$, calculate $\mathbf{d}^{(C^*, z^*)}$ using (4), and set $\mathbf{d}^* = \mathbf{d}^{(C^*, z^*)}$
24: **Return** $\mathbf{d}^*$ and $z^*$

the cut $\tilde{C} = \{(1, 2), (1, 3)\}$ with value $\sum_{(i,j)\in\tilde{C}} w'_{ij} = 60$ (shown in red dotted lines). Because this value is greater than $W = 15$, the algorithm skips Lines 2–4. We use the initialization $z_{\max} = c_{\max}$ (see Remark 2) in Line 5 and create a sorted list of arc capacities (including 0) according to Line 6. Fig. 2(c) shows the first execution of Line 10 within the binary search with $k = 4$, $z_4 = 7$, and updated arc weights given by $\bar{\mathbf{w}}^{z_4}$. This line returns the flow value $v = 4$ with corresponding cut $C = \{(3, 5), (4, 6)\}$ (shown in red dotted lines). Because $v \leq W$, Line 12 updates $k_U = 4$, $\tilde{z} = z_4 = 7$, $v^* = 4$, and $C^* = \{(3, 5), (4, 6)\}$, and Line 9 sets $k = 2$. Fig. 2(d) shows the execution of Line 10 in the binary search with $k = 2$, $z_2 = 5$, and updated arc weights given by $\bar{\mathbf{w}}^{z_2}$. This line returns the flow value $v = 12$ with corresponding cut $C = \{(3, 5), (2, 4)\}$ (shown in red dotted lines). Line 12 updates $k_U = 2$, $\tilde{z} = z_2 = 5$, $v^* = 12$, and $C^* = \{(2, 4), (3, 5)\}$, and Line 9 sets $k = 1$. The next iteration of the binary search is shown in Fig. 2(e), where Line 10 returns the flow value $v = 24$ with corresponding cut $C = \{(1, 2), (3, 2), (3, 5)\}$ (shown in red dotted lines). Because $v > W$, Line 14 updates $k_L = 1$ and Phase 1 terminates because the condition of the While loop in Line 8 is no longer satisfied. Fig. 2(f) shows the execution of Phase 2, where Line 17 constructs the set $S^* = \{(1, 2), (1, 3), (2, 4), (3, 5), (4, 3), (5, 4), (4, 6), (5, 6)\}$ and

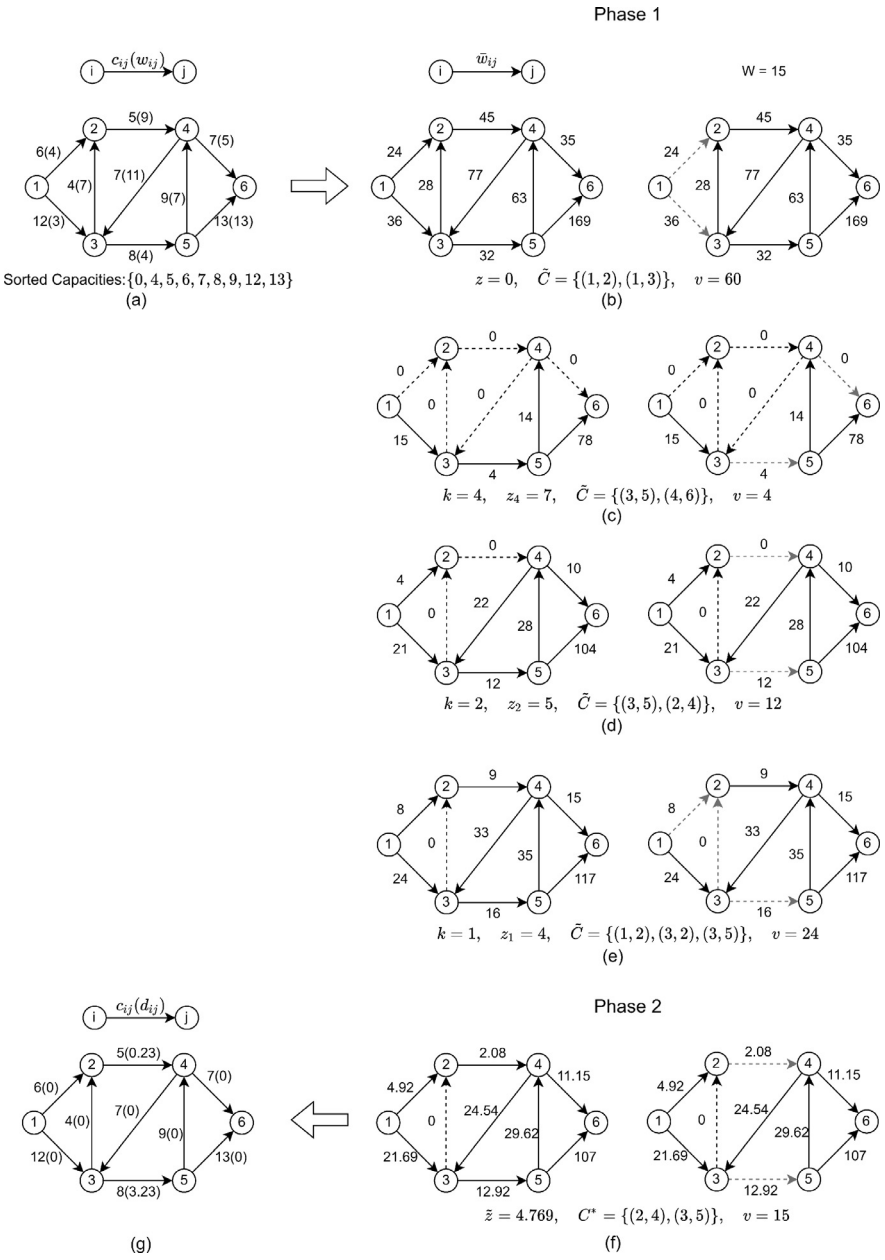



**Fig. 2.** Step-by-step example of the execution of Algorithm 1.

Line 20 sets $\tilde{z} = 4.769$ because $12 = v^* \neq 15$. The minimum *st*-cut in the modified network with weights $\bar{\mathbf{w}}^{\tilde{z}}$ (shown in Fig. 2(f)) obtained in Line 21 is $C^* = \{(2,4),(3,5)\}$ with value $v^* = 15$. The While loop terminates because $v^* = W$ and the condition in Line 18 is no longer satisfied. Line 23 calculates the attacks $\mathbf{d}^*$ using (4), which are shown in parenthesis in Figure Fig. 2(g). The algorithm returns $z^* = 4.769$ and $\mathbf{d}^*$ in Line 24.

## 6. A reduction to a zero-sum game

Algorithm 1 solves Problem (3) in polynomial time assuming that user and interdictor are not subject to any side constraint beyond those in Problem 3. However, Algorithm 1 cannot solve other problem variants involving additional constraints, for instance any constraint making a cut or path infeasible. In this section, we show that Problem (3) can be reduced to a zero-sum noncooperative game. We prove that the game always has a pure Nash-equilibrium point. This is an interesting result from a theoretical point of view because in general zero-sum games may not always admit a pure Nash-equilibrium point (see Mazalov, 2014 for further details). This reduction allows us to convert Problem (3) into a constrained zero-sum game, where some side constraints can be easily incorporated.

We first determine the set of pure strategies for each agent in Problem (3). The user's strategy set consists of all *st*-paths in $G$. To determine a strategy set for the interdictor, we use the decision variables $\mathbf{d}^{(C,z)}$ from Definition 1 and remove their dependency on $z$. Let $\check{C}$ be the set of all *st*-cuts in $G$ such that $\bar{\mathbf{w}}^{z_{\max}}(C) \leq W$ for each $C \in \check{C}$. We show that there is a unique number $z_C \in [0, z_{\max}]$ for each *st*-cut $C$ such that $\mathbf{d}^{(C,z_C)}$ satisfies the budget constraint of Problem (3) in equality form. This construction allows us to define the interdictor's strategy set as all *st*-cuts $C \in \check{C}$ in $G$. As a result, if the user chooses an *st*-path $P$ and the interdictor attacks an *st*-cut $C$, then the user's payoff is the leftover capacity of $P$ after the interdictor reduces the capacity of some arcs in $C$. Because this is a zero-sum game, the interdictor's payoff is the capacity loss of the user. This implies that the user's and interdictor's payoffs are

at most $z_C$. To remove dependency of $\mathbf{d}^{(C,z)}$ on $z$, we define $z^C$ as follows.

**Definition 3.** Define $C$ as an $st$-cut in $G$ such that $\bar{\mathbf{w}}^{z_{\max}}(C) \leq W$. Define $g^C(z) = \bar{\mathbf{w}}^z(C) - W$, where $g^C : [0, z_{\max}] \to \mathbb{R}$ and let $z^C$ be the solution of $g^C(z) = 0$.

From Lemma 1, $g^C(z)$ is a convex decreasing piecewise-linear function for $z \in [0, z_{\max}]$.

**Lemma 4.** *For each st-cut $C \in \check{\mathcal{C}}$ in $G$, $z^C$ is the best attainable objective value to Problem (3) when the interdictor reduces the arc capacities in cut $C$ only.*

**Proof.** Suppose for a contradiction that $z^C$ is not the best attainable objective that the interdictor can achieve by reducing the arc capacities in cut $C$. Then, let $z' < z^C$ be the best objective the interdictor can achieve. Because $g^C$ is decreasing, then $g^C(z') > g^C(z^C) = 0$. This implies $g^C(z') > 0$, i.e., $\bar{\mathbf{w}}^{z'}(C) > W$, which means that $z'$ is unattainable (i.e., infeasible) with respect to cut $C$. □

The following properties of $z^C$ are immediate.

**Property 5.** *For each st-cut $C \in \check{\mathcal{C}}$ in $G$, $z^C \in [0, z_{\max}]$ and it is unique.*

The uniqueness of $z^C$ follows from the fact that $g^C$ is decreasing in the interval $[0, z_{\max}]$, so the root of the function must be unique. Furthermore, $z^C \in [0, z_{\max}]$ since $g^C(z_{\max}) \leq 0$ by Definition 3.

Because $g^C$ is a convex decreasing piecewise-linear function in the interval $[0, z_{\max}]$, we find the root of $g^C$ using the same principles from the Newton's method described in Section 4. In particular, we generate a nondecreasing sequence $\{\bar{z}_k\}_{k=0,1,\dots}$ converging to the root $z^C$, where $\bar{z}_0 = 0$. Define $S_k(C) = \{(i,j) \in C : c_{ij} \geq \bar{z}_k\}$ and the $k$-th element in this sequence, $\bar{z}_k$, using following update rule.

$$\begin{aligned}\bar{z}_k &= \bar{z}_{k-1} - \frac{g^C(\bar{z}_{k-1})}{g'^C(\bar{z}_{k-1})} \\ &= \bar{z}_{k-1} - \frac{\sum_{(i,j)\in S_{k-1}(C)} w_{ij}(c_{ij} - \bar{z}_{k-1}) - W}{-\sum_{(i,j)\in S_{k-1}(C)} w_{ij}} \\ &= \frac{\sum_{(i,j)\in S_{k-1}(C)} w_{ij}c_{ij} - W}{\sum_{(i,j)\in S_{k-1}(C)} w_{ij}} \end{aligned} \tag{10}$$

**Theorem 5.** *The update rule in (10) converges to $z^C$ if $S_k(C) = S_{k-1}(C)$ for any given $k \in \{0, 1, 2, \dots\}$.*

**Proof.** When $S_k(C) = S_{k-1}(C)$ we have that

$$\begin{aligned}\bar{z}_k &= \frac{\sum_{(i,j)\in S_{k-1}(C)} w_{ij}c_{ij} - W}{\sum_{(i,j)\in S_{k-1}(C)} w_{ij}} \\ &= \frac{\sum_{(i,j)\in S_k(C)} w_{ij}c_{ij} - W}{\sum_{(i,j)\in S_k(C)} w_{ij}}.\end{aligned}$$

Reorganizing terms, we obtain $\sum_{(i,j)\in S_k(C)} w_{ij}(c_{ij} - \bar{z}_k) = W$, which implies that $g^C(\bar{z}_k) = 0$ by definition. Hence, $\bar{z}_k = z^C$. □

Property 6 describes a feature of function $g^C(z)$ that guarantees the convergence of the update rule in (10) in a finite number of iterations.

**Property 6.** *The breakpoints of the convex decreasing piecewise-linear function*

$$\begin{aligned}g^C(z) &= \bar{w}^z(C) - W \\ &= \sum_{(i,j)\in C} \max\{0, w_{ij}(c_{ij} - z)\} - W\end{aligned}$$

*correspond to the capacity values for arcs in $C$. Hence, the Newton's method based procedure converges in a finite number of iterations, which is bounded by the number of unique arc capacities in cut $C$.*

**Table 1**
Calculation of $z^C$ for a given $st$-cut.

| $k$ | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| $\bar{z}_k$ | 0 | 7.40 | 10.20 | **10.50** |
| $S_k(C)$ | $\{e_1, e_2, \dots, e_{10}\}$ | $\{e_6, e_7, e_8, e_9, e_{10}\}$ | $\{e_7, e_8, e_9, e_{10}\}$ | $\{e_7, e_8, e_9, e_{10}\}$ |

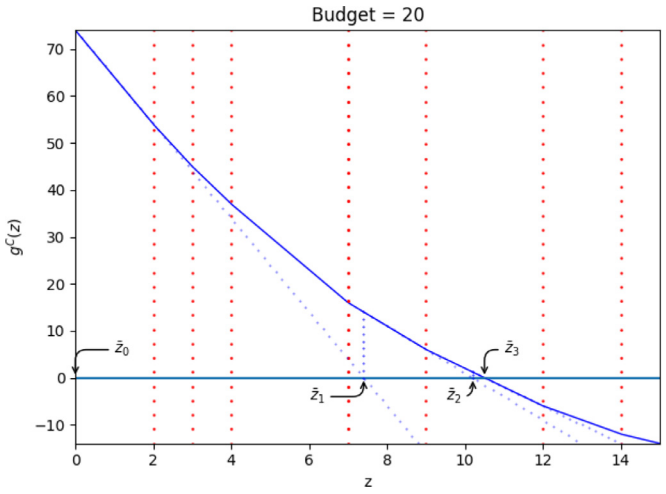



**Fig. 3.** Sequence of $\{\bar{z}_k\}$ for a given cut $C$.

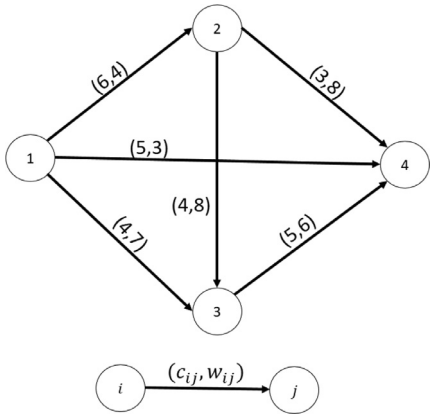



**Fig. 4.** Instance of Problem (3).

We illustrate this root-finding procedure using Example 2.

**Example 2.** Consider an instance of Problem (3) with $z_{\max} = 15$, with an $st$-cut consisting of 10 arcs, $C = \{e_1, e_2, \dots, e_{10}\}$. The initial arc capacities are $\{2, 3, 4, 7, 7, 9, 12, 14, 18, 18\}$ and the interdiction budget is $W = 20$. For simplicity, we assume that all $w$-values are equal to 1. Table 1 shows the values of $\bar{z}_k$ calculated by (10).

Table 1 shows that the sequence $\{\bar{z}_k\}_{k=0}^3$ is increasing in $k$, given the update from (10). At $k = 3$, we have that $S_3(C) = S_2(C)$, which is the convergence condition.

Fig. 3 illustrates the corresponding function $g^C(z)$, its breakpoints and root, as well as the $\bar{z}_k$-values until convergence.

The following example illustrates the reduction of an instance of Problem (3) to a zero-sum noncooperative game.

**Example 3.** Consider the instance of Problem (3) depicted in Fig. 4, with $s = 1$ and $t = 4$. Recall that the strategy set for the user consists of all $st$-paths, while for the interdictor consists of all $st$-cuts. Table 2 shows the payoff matrix for the user perspective, for all combinations of user $st$-paths and interdictor's $st$-cuts. The payoff is the capacity of a path $P$ after the interdictor optimally decreases

**Table 2**
User's payoff matrix for the zero-sum game in Fig. 4.

| | $C_1 = [S_1, \bar{S}_1]$<br>$S_1 = \{1\}$<br>$\bar{S}_1 = \{2,3,4\}$<br>$z^{C_1} = \frac{47}{14}$ | $C_2 = [S_2, \bar{S}_2]$<br>$S_2 = \{1,2\}$<br>$\bar{S}_2 = \{3,4\}$<br>$z^{C_2} = \frac{55}{18}$ | $C_3 = [S_3, \bar{S}_3]$<br>$S_3 = \{1,3\}$<br>$\bar{S}_3 = \{2,4\}$<br>$z^{C_3} = \frac{49}{13}$ | $C_4 = [S_4, \bar{S}_4]$<br>$S_4 = \{1,2,3\}$<br>$\bar{S}_4 = \{4\}$<br>$z^{C_4} = 3$ |
|---|---|---|---|---|
| $P_1 : 1-2-4$ | 3 | 3 | 3 | 3 |
| $P_2 : 1-2-3-4$ | $\frac{47}{14}$ | $\frac{55}{18}$ | $\frac{49}{13}$ | 3 |
| $P_3 : 1-3-4$ | $\frac{47}{14}$ | $\frac{55}{18}$ | $\frac{49}{13}$ | 3 |
| $P_4 : 1-4$ | $\frac{47}{14}$ | $\frac{55}{18}$ | $\frac{49}{13}$ | 3 |

the arc capacities in cut $C$. In this example, the strategy profile $(P_1, C_4)$ is a pure Nash equilibrium solution with payoff equal to 3.

In the same setting of Example 3, suppose that there are side constraints that make cuts $C_1$ and $C_3$ infeasible. This new problem cannot be solved directly by Algorithm 1. However, this can be solved using the proposed reduction by removing strategies $C_1$ and $C_3$ because they do not satisfy the new constraint. This reduces the size of the game, and strategy profile $(P_1, C_4)$ remains optimal.

The game described in Example 3 contains a pure equilibrium solution, which indeed is always the case. Theorem 6 establishes that the proposed game always contains a pure Nash equilibrium solution. Remark 3 states that a sequential game can be depicted as a simultaneous game in normal form.

**Theorem 6.** *The corresponding zero-sum game to any instance of Problem (3) has always a pure Nash equilibrium solution.*

**Proof.** Let $A = (a_{ij})$ be the payoff matrix for the user. Consider the ($st$-cut) strategy $C_j$ for the interdictor, where $z^{C_j} = \min_{C \in \check{\mathcal{C}}}\{z^C\}$, and the strategy $P_i$ for the user, which corresponds to the maximum value of elements belonging to the $j$-th column of $A$. Formally, the payoff value of strategy $(P_i, C_j)$ corresponds to $a_{ij}$, where $i = \arg\max_{r=1,\ldots,|\mathcal{P}|} \min\{c(P_r), z^{C_j}\}$, where $c(P_r)$ is the capacity of path $P_r$ (i.e., $c(P_r) = \min_{(k,\ell)\in P_r} c_{k\ell}$). Next, we prove that strategy $(P_i, C_j)$ is a pure Nash equilibrium. By construction, we have that the user has no incentive to deviate from $P_i$ given $C_j$ as there is no strictly better payoff in the corresponding column of the payoff matrix. To prove that the interdictor has no incentive to change strategy $C_j$, suppose for a contradiction that the $st$-cut $C_l$ is chosen instead of $C_j$, with $l \neq j$, and that the optimal payoff in this case is $a_{\hat{\ell}l}$, corresponding to strategy $(P_{\hat{\ell}}, C_l)$ and where $\hat{\ell} = \arg\max_{r=1,\ldots,|\mathcal{P}|} \min\{c(P_r), z^{C_l}\}$. We examine three cases for $a_{\hat{\ell}l}$:

- **Case 1**: $a_{\hat{\ell}l} = z^{C_l}$. In this case, the user's payoff is increased because $z^{C_j} = \min\{z^C : C \in \check{\mathcal{C}}\} \leq z^{C_l}$, hence the interdictor has no incentive to adopt the new strategy $C_l$.
- **Case 2**: $a_{ij} \leq a_{\hat{\ell}l} < z^{C_l}$. In this case, the interdictor has no incentive to change strategy $C_j$ because it provides a payoff that is at least as good as $C_l$.
- **Case 3**: $a_{\hat{\ell}l} < a_{ij}$. This means that $c(P_{\hat{\ell}}) = a_{\hat{\ell}l} < a_{ij} \leq c(P_i)$, which is impossible because the user can choose path $P_i$ and obtain a payoff $a_{\hat{\ell}l} \geq c(P_i)$.

Piecing together these cases leads to the conclusion that there is no cut $C_l$ better than $C_j$ for the interdictor, with $l \neq j$, thus strategy $(P_i, C_j)$ is a pure Nash equilibrium. □

**Remark 3.** The CMCPIP is a sequential game, where the interdictor plays first and then the user, fully aware of the interdictor's choice decides the path to follow. However, this game can be equally represented as a simultaneous game in normal form because regardless of the path chosen, the user's payoff will be at-most $z^C$ for any choice of $st$-cut $C$ of the interdictor. That is, the payoff of the user's choice is not affected by the order of play.

We also point out that in this zero-sum game the user wants to maximize their payoff (maximum capacity of the path), while the interdictor wants to minimize the user's payoff. This means that the interdictor's payoff is the negative of the user's payoff, so it suffices to depict only one payoff.

Although the result of this section is theoretically interesting, its practical applicability is limited to small networks because the strategy sets of the user ($st$-paths) and the interdictor ($st$-cuts) grow exponentially with the size of network.

## 7. Computational experiments

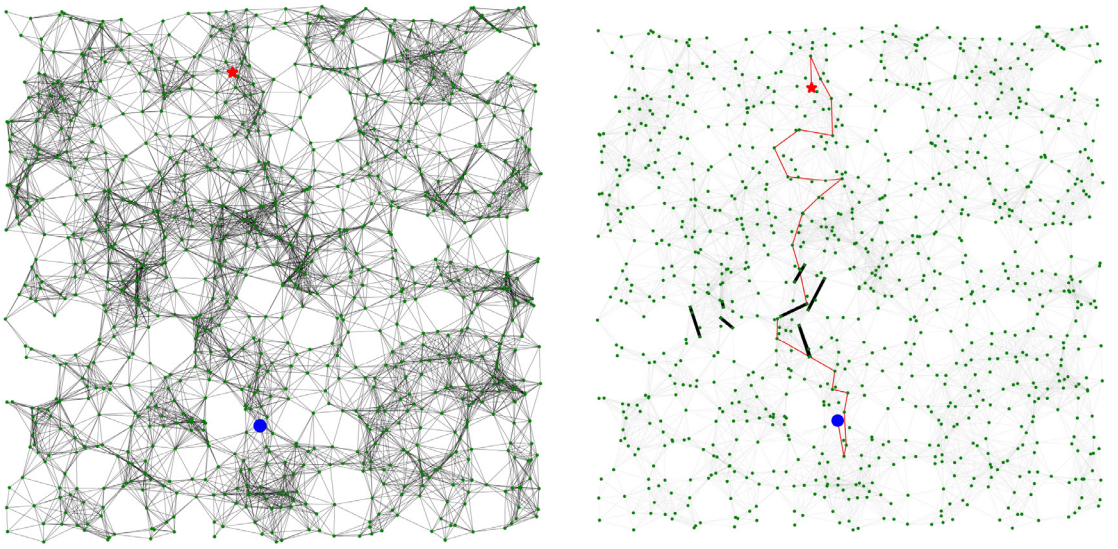
This section illustrates the computational performance of our solution approach. We implemented Algorithm 1 in a computer with an Intel Xeon E5-2680 v4 CPU running at 2.40 GHz, 16 GB of RAM, and Linux 3.10.0. We used Python 3.6 and the embedded routines in Matplotlib and NetworkX for the code development and instance generation (Batagelj & Brandes, 2005). Section 7.1 presents an example on the interdiction of wireless sensor networks, where an interdictor affects the transmission bandwidth between sensors. Section 7.2 demonstrates the performance of our algorithm on a set of randomly generated networks of various sizes based on Erdős & Rényi (1976) and scale-free graphs (Barabási & Albert, 1999).

### 7.1. Interdiction of wireless sensor networks

Wireless Sensor Networks (WSNs) consist of a group of sensors located in a space with the purpose of collecting physical or environmental data of interest, which is then transmitted wirelessly from sensor to sensor until reaching a base station (or sink) for processing. Sensors must be strategically located in the surveyed area due to their limited transmission range in order to ensure a continuous communication to the base station. The use of WSNs includes environmental monitoring, infrastructure surveillance, precision agriculture, fire detection, and supply chain management, among others (Othman & Shazali, 2012; Xu, 2002).

Consider the randomly generated 1000-sensor WSN shown in Fig. 5a. Sensor locations are depicted as nodes and an arc between two nodes means that the corresponding sensors are within range and can communicate. The network in Fig. 5a is undirected for illustrative purposes, but the communication between sensors is bidirectional and the CMCPIP is solved over a directed network with two arcs—$(i, j)$ and $(j, i)$—for every pair of communicating sensors $i$ and $j$. This leads to a network of 16,578 arcs. The network user is interested in finding the maximum transmission capacity path between a given sensor (source) and the base station, which are depicted in Fig. 5a with a circle and a star, respectively. Arc capacities represent the transmission bandwidth between sensors, which we generate using an integer uniform distribution in the range [10,50]. We generate arc interdiction costs using an integer uniform distribution in the range [10, 100], except for those arcs close to the source or base node, which we assume are prohibitively expensive to attack. This reflects that such locations are highly protected to deter any interdiction action. We define the minimum cost of totally isolating the source sensor as the value of the minimum-cost cut between source and base station in the same WSN but using arc costs given by $c_{ij}w_{ij}$, for each $(i, j) \in A$ (i.e., using the costs of reducing each arc capacity to zero). We calculate the interdictor's budget as a percentage of the total isolation cost.

Fig. 5 b shows the optimal interdiction plan when the interdictor's budget is equal to 1% of the total isolation cost. The thicker black arcs are attacked, i.e., arcs such that $d^*_{ij} > 0$ in the solution returned in Line 24 of Algorithm 1. Fig. 5b also shows a maximum transmission path between the source sensor and the base node using an attacked arc. Note that this path may not be unique.



(a) WSN input network

(b) Arcs attacked and maximum capacity (bandwidth) path

**Fig. 5.** Interdiction of a WSN.

The maximum transmission capacity in the absence of the interdictor is equal to 45, which is reduced given the attacks to 42.66 (5.21% decrease). Solving CMCPIP on the same WSN and using interdiction budgets of 2%, 5%, and 10% of the total isolation cost results in maximum transmission capacities of 40.78 (9.37% decrease), 37.15 (17.44% decrease), and 32.56 (27.64% decrease), respectively. These results illustrate that even with a small budget the interdictor is able to disrupt the user operations to some degree. Algorithm 1 solved these WSN instances in less than 2 seconds.

### 7.2. Randomly generated instances

We use the undirected binomial random graphs introduced by Erdõs & Rényi (1976) to create random instances of various arc densities. Binomial graphs are generated using two parameters, $n$ and $p \in [0, 1]$, where $n$ is the number of nodes and any edge $(i, j)$ exists with probability $p$. As $p$ increases, the graph becomes denser. In particular, $p = 1$ corresponds to a complete graph. We also generate random graphs using the Barabási-Albert (BA) model (Barabási & Albert, 1999), which are scale-free and approximate the behavior of systems in which few nodes have a relatively high degree with respect to other nodes (e.g., internet, social networks). Under the BA model, graphs are generated by sequentially adding new nodes and arcs using a *preferential attachment rule* in which highly connected nodes are more likely to receive more arc connections. This is controlled by parameter $h$, which is the number of initially connected nodes that have a higher chance to be connected to new nodes as the graph construction progresses.

In all instances, we assume that the origin is Node 1 (i.e., $s = 1$) and the destination is Node $n$ (i.e., $t = n$). We generate three groups of $c$-parameters to evaluate the influence of different capacities on the algorithm's performance. These capacities are generated using integer uniform distributions with ranges $[50, 500]$, $[50, 1000]$, and $[50, 2000]$, respectively. In all instances, the $w$-parameters are integer and uniformly distributed in the range [1,1000], and the interdiction budget $W$ is a percentage of the total isolation cost of node $s$. To construct a directed network, we first generate an undirected graph and then create arcs $(i, j)$ and $(j, i)$ for each existing edge $\{i, j\}$ in the input graph. We avoid arc duplicates and assign both arcs the same capacity and interdiction cost as the edge in the initial undirected graph. The results presented are averages across five randomly generated networks of the same size.

We test our algorithm on multiple networks with different values of $p$ and $h$, resulting in a number of nodes between 100 and 2000 and a number of arcs between 1600 and 3.6M. Tables 3 and 4 summarize the performance of Algorithm 1 on networks generated from Erdõs-Rényi binomial graphs and Barabási-Albert scale-free graphs, respectively. In both cases, the solution times ($t$) increase with the network size and the length of the capacity range as the number of (possibly) distinct capacity values increase. This is because the input set of distinct $z$-values in the binary search procedure tends to be larger as the capacity range increases, requiring more iterations of Phase 1. Moreover, a budget increase allows the interdictor to achieve a smaller objective function value via more arcs attacked or more intense attacks. This results in more time-consuming subproblems in Lines 10 and 21 as the size of the input networks (and number of arcs with 0 capacity) depend on the arc weights given by (5). We also report the impact of the interdictor actions as $\Delta z = (z^0 - z^*)/z^0$, where $z^*$ is the optimal value of CMCPIP returned by Algorithm 1 in Line 24 and $z^0$ is the capacity of the maximum capacity path in the absence of interdictor. As expected, the interdiction impact increases with the budget. Attacks reduce the capacity of the maximum capacity path by at least 7.30% (1% budget) and up to 32.96% (10% budget) in the networks generated from binomial graphs and by at least 5.62% (1% budget) and up to 33.23% (10% budget) in the networks generated from scale-free graphs. These results illustrate that the interdictor can cause significant damage to the user operations even with a small budget. Tables 3 and 4 demonstrate the scalability of Algorithm 1, which is able to solve very-large scale instances (e.g., 2000 nodes and 3.6M arcs) in less than 500 seconds on average given its polynomial time nature.

Tables 5 and 6 report the run time of each Phase of Algorithm 1 for the binomial- and scale-free graphs, respectively. In both cases, the run time of each phase increases with the input network size and the interdictor's budget. Increasing the length of the capacity range results in longer solution times ($t$) due to more iterations performed in Phase 1. This is because the number

**Table 3**
Solution time and interdiction impact on networks generated from Erdős-Rényi binomial graphs.

| | | Budget % | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 1% | | 2% | | 5% | | 10% | |
| $(n, m, p)$ | Capacity | $t$ (s) | $\Delta z$ (%) | $t$ (s) | $\Delta z$ (%) | $t$ (s) | $\Delta z$ (%) | $t$ (s) | $\Delta z$ (%) |
| (100, 5054, 0.3) | [50, 500] | 0.35 | 7.44 | 0.44 | 12.41 | 0.45 | 20.73 | 0.51 | 32.96 |
| | [50, 1000] | 0.39 | 7.30 | 0.42 | 10.20 | 0.45 | 23.85 | 0.47 | 29.64 |
| | [50, 2000] | 0.44 | 8.26 | 0.42 | 13.19 | 0.51 | 22.41 | 0.55 | 29.71 |
| (100, 9008, 0.7) | [50, 500] | 0.68 | 9.61 | 0.66 | 13.70 | 0.78 | 23.02 | 0.84 | 30.95 |
| | [50, 1000] | 0.70 | 9.03 | 0.69 | 13.26 | 0.82 | 22.55 | 0.92 | 31.25 |
| | [50, 2000] | 0.73 | 9.46 | 0.71 | 13.12 | 0.84 | 23.01 | 0.92 | 32.79 |
| (500, 126996, 0.3) | [50, 500] | 7.32 | 9.41 | 7.73 | 13.68 | 8.68 | 22.29 | 9.71 | 30.44 |
| | [50, 1000] | 8.20 | 10.21 | 8.14 | 14.00 | 9.48 | 23.13 | 10.31 | 30.46 |
| | [50, 2000] | 8.13 | 10.05 | 9.15 | 14.39 | 10.04 | 22.93 | 11.18 | 30.97 |
| (500, 226936, 0.7) | [50, 500] | 13.81 | 11.29 | 14.62 | 14.89 | 17.76 | 22.14 | 18.52 | 32.09 |
| | [50, 1000] | 13.89 | 9.50 | 15.26 | 14.60 | 17.10 | 22.89 | 19.39 | 32.07 |
| | [50, 2000] | 15.16 | 10.32 | 16.33 | 13.89 | 17.91 | 22.68 | 20.78 | 32.11 |
| (1000, 509468, 0.3) | [50, 500] | 30.23 | 10.09 | 32.83 | 13.99 | 38.45 | 23.03 | 42.31 | 31.62 |
| | [50, 1000] | 32.50 | 9.83 | 36.50 | 13.97 | 41.74 | 23.49 | 44.21 | 31.56 |
| | [50, 2000] | 35.44 | 10.60 | 37.58 | 14.13 | 42.14 | 23.10 | 48.53 | 32.07 |
| (1000, 909596, 0.7) | [50, 500] | 55.38 | 9.84 | 60.76 | 13.92 | 74.14 | 22.90 | 77.33 | 31.29 |
| | [50, 1000] | 59.11 | 10.50 | 64.93 | 14.01 | 77.24 | 22.38 | 84.35 | 32.22 |
| | [50, 2000] | 63.10 | 9.87 | 68.44 | 13.76 | 76.55 | 22.33 | 89.46 | 31.81 |
| (2000, 2039320, 0.3) | [50, 500] | 128.60 | 10.13 | 142.81 | 14.59 | 163.39 | 22.54 | 183.96 | 32.00 |
| | [50, 1000] | 143.29 | 10.39 | 152.74 | 14.36 | 168.16 | 22.44 | 198.23 | 31.85 |
| | [50, 2000] | 146.57 | 10.39 | 158.66 | 14.69 | 174.93 | 22.66 | 214.90 | 31.54 |
| (2000, 3637428, 0.7) | [50, 500] | 242.12 | 9.79 | 262.14 | 14.26 | 310.88 | 22.46 | 352.48 | 31.51 |
| | [50, 1000] | 261.45 | 10.02 | 311.25 | 14.09 | 321.55 | 22.98 | 372.05 | 32.00 |
| | [50, 2000] | 274.28 | 10.28 | 317.19 | 14.08 | 332.15 | 22.48 | 388.82 | 31.58 |

**Table 4**
Solution time and interdiction impact on networks generated from Barabási-Albert scale-free graphs.

| | | Budget % | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 1% | | 2% | | 5% | | 10% | |
| $(n, m, h)$ | Capacity | $t$ (s) | $\Delta z$ (%) | $t$ (s) | $\Delta z$ (%) | $t$ (s) | $\Delta z$ (%) | $t$ (s) | $\Delta z$ (%) |
| (100, 1600, 20) | [50, 500] | 0.26 | 7.26 | 0.28 | 13.38 | 0.33 | 21.73 | 0.36 | 33.23 |
| | [50, 1000] | 0.27 | 5.75 | 0.31 | 14.14 | 0.32 | 17.64 | 0.35 | 30.88 |
| | [50, 2000] | 0.31 | 8.87 | 0.31 | 11.13 | 0.30 | 18.59 | 0.37 | 30.74 |
| (100, 2500, 50) | [50, 500] | 0.35 | 5.62 | 0.36 | 10.87 | 0.39 | 20.81 | 0.49 | 29.82 |
| | [50, 1000] | 0.39 | 6.36 | 0.41 | 12.68 | 0.43 | 21.42 | 0.46 | 30.80 |
| | [50, 2000] | 0.42 | 7.20 | 0.38 | 10.59 | 0.45 | 19.73 | 0.51 | 30.11 |
| (500, 40000, 100) | [50, 500] | 4.51 | 9.26 | 4.94 | 12.47 | 5.81 | 21.92 | 6.09 | 32.53 |
| | [50, 1000] | 5.04 | 10.74 | 5.11 | 13.23 | 6.14 | 22.19 | 6.58 | 31.24 |
| | [50, 2000] | 5.26 | 10.13 | 5.43 | 13.71 | 6.27 | 22.02 | 6.94 | 30.73 |
| (500, 62500, 250) | [50, 500] | 7.31 | 8.94 | 7.52 | 12.56 | 8.61 | 21.35 | 9.58 | 32.89 |
| | [50, 1000] | 7.71 | 9.52 | 7.89 | 14.19 | 8.54 | 20.81 | 10.59 | 31.36 |
| | [50, 2000] | 7.93 | 9.38 | 8.38 | 13.60 | 9.63 | 22.76 | 10.66 | 31.51 |
| (1000, 160000, 200) | [50, 500] | 19.44 | 9.56 | 20.36 | 13.45 | 23.13 | 21.08 | 25.95 | 30.21 |
| | [50, 1000] | 19.56 | 9.17 | 22.09 | 13.22 | 23.62 | 21.94 | 27.45 | 30.78 |
| | [50, 2000] | 21.01 | 8.68 | 22.93 | 13.63 | 25.64 | 22.73 | 29.54 | 31.59 |
| (1000, 250000, 500) | [50, 500] | 29.18 | 9.05 | 31.35 | 14.02 | 34.47 | 20.80 | 40.89 | 30.88 |
| | [50, 1000] | 31.26 | 10.46 | 33.15 | 12.70 | 37.79 | 22.34 | 42.75 | 31.30 |
| | [50, 2000] | 32.66 | 9.74 | 37.04 | 13.42 | 38.86 | 21.61 | 47.72 | 31.81 |
| (2000, 640000, 400) | [50, 500] | 80.38 | 9.92 | 87.18 | 14.68 | 98.95 | 22.92 | 110.84 | 31.10 |
| | [50, 1000] | 83.34 | 9.94 | 92.30 | 13.75 | 102.31 | 22.41 | 118.29 | 31.99 |
| | [50, 2000] | 91.48 | 10.41 | 100.22 | 14.50 | 115.21 | 21.60 | 126.64 | 31.42 |
| (2000, 1000000, 1000) | [50, 500] | 128.94 | 9.55 | 135.21 | 13.68 | 154.03 | 22.05 | 185.61 | 32.24 |
| | [50, 1000] | 132.85 | 10.03 | 144.30 | 14.00 | 160.69 | 21.89 | 186.74 | 31.74 |
| | [50, 2000] | 137.87 | 9.69 | 152.41 | 13.66 | 171.48 | 21.34 | 196.85 | 31.61 |

of (possibly) distinct capacity values increase with the length of the capacity interval. The performance of Phase 2 is similar across capacity values. Phase 1 is consistently more time-consuming as it performs more iterations until identifying the range of $z$-values containing the optimal objective function value. Phase 2 performs very few iterations as it only needs to identify the optimal objective function value and the corresponding set of arcs to attack within the already narrowed range of $z$-values (and subset of arcs) provided by Phase 1. In our experiments, Phase 1 performs between 8 to 11 iterations, whereas Phase 2 performs no more than 3 iterations (2 for most instances). These experiments show that variations in input parameters and network size affect the solution times in a similar manner in the examined networks regardless of their structure.

## 8. Final remarks

In this paper, we study the maximum capacity path interdiction problems with continuous interdiction. We propose an efficient algorithm for its solution, combining a binary search procedure and a discrete-type Newton's method. The algorithm first obtains an interval that contains the optimal objective function and

**Table 5**
Solution time and iterations per phase on networks generated from Erdõs-Rényi binomial graphs.

| | | Budget % | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1% | | | | 5% | | | | 10% | | | |
| | | Phase 1 | | Phase 2 | | Phase 1 | | Phase 2 | | Phase 1 | | Phase 2 | |
| $(n, m, h)$ | Capacity | $t$ (s) | iter. | $t$ (s) | iter. | $t$ (s) | iter. | $t$ (s) | iter. | $t$ (s) | iter. | $t$ (s) | iter. |
| (100, 5054, 0.3) | [50, 500] | 0.29 | 8.8 | 0.07 | 2.0 | 0.36 | 9.0 | 0.09 | 2.0 | 0.41 | 8.8 | 0.10 | 2.0 |
| | [50, 1000] | 0.32 | 10.0 | 0.07 | 2.0 | 0.36 | 9.6 | 0.08 | 2.0 | 0.38 | 9.8 | 0.09 | 2.0 |
| | [50, 2000] | 0.37 | 10.6 | 0.07 | 2.0 | 0.42 | 10.6 | 0.09 | 2.0 | 0.45 | 10.6 | 0.10 | 2.0 |
| (100, 9008, 0.7) | [50, 500] | 0.55 | 9.0 | 0.12 | 2.0 | 0.66 | 9.0 | 0.12 | 2.0 | 0.70 | 8.8 | 0.14 | 2.0 |
| | [50, 1000] | 0.59 | 10.0 | 0.10 | 2.0 | 0.70 | 9.8 | 0.12 | 2.0 | 0.77 | 10.0 | 0.15 | 2.0 |
| | [50, 2000] | 0.64 | 11.0 | 0.09 | 2.0 | 0.72 | 11.0 | 0.12 | 2.0 | 0.78 | 10.6 | 0.13 | 2.0 |
| (500, 126996, 0.3) | [50, 500] | 6.03 | 9.0 | 1.29 | 2.0 | 7.00 | 8.8 | 1.68 | 2.0 | 7.78 | 9.0 | 1.93 | 2.0 |
| | [50, 1000] | 6.83 | 10.0 | 1.37 | 2.0 | 7.80 | 10.0 | 1.68 | 2.0 | 8.43 | 9.8 | 1.89 | 2.0 |
| | [50, 2000] | 6.88 | 11.0 | 1.25 | 2.0 | 8.38 | 10.6 | 1.66 | 2.0 | 9.31 | 11.0 | 1.86 | 2.0 |
| (500, 226936, 0.7) | [50, 500] | 11.37 | 9.0 | 2.45 | 2.0 | 14.45 | 9.0 | 3.31 | 2.0 | 14.75 | 8.6 | 3.76 | 2.0 |
| | [50, 1000] | 11.64 | 9.8 | 2.24 | 2.0 | 14.09 | 10.0 | 3.01 | 2.0 | 15.87 | 10.0 | 3.52 | 2.0 |
| | [50, 2000] | 12.86 | 11.0 | 2.30 | 2.0 | 14.96 | 11.0 | 2.94 | 2.0 | 17.33 | 11.0 | 3.46 | 2.0 |
| (1000, 509468, 0.3) | [50, 500] | 24.91 | 8.8 | 5.32 | 2.0 | 31.19 | 8.8 | 7.27 | 2.0 | 33.98 | 9.0 | 8.33 | 2.0 |
| | [50, 1000] | 27.30 | 10.0 | 5.20 | 2.0 | 34.38 | 10.0 | 7.36 | 2.0 | 36.08 | 9.8 | 8.13 | 2.0 |
| | [50, 2000] | 29.94 | 11.0 | 5.50 | 2.0 | 35.12 | 11.0 | 7.03 | 2.0 | 40.36 | 11.0 | 8.17 | 2.0 |
| (1000, 909596, 0.7) | [50, 500] | 45.91 | 8.8 | 9.47 | 2.0 | 60.54 | 9.0 | 13.60 | 2.0 | 62.13 | 8.8 | 15.20 | 2.0 |
| | [50, 1000] | 49.55 | 9.8 | 9.57 | 2.0 | 63.88 | 9.8 | 13.36 | 2.0 | 69.23 | 9.8 | 15.12 | 2.0 |
| | [50, 2000] | 53.66 | 11.0 | 9.44 | 2.0 | 63.35 | 10.8 | 13.20 | 2.0 | 74.41 | 10.8 | 15.05 | 2.0 |
| (2000, 2039320, 0.3) | [50, 500] | 105.84 | 8.4 | 22.76 | 2.0 | 132.25 | 8.6 | 31.14 | 2.0 | 147.05 | 8.8 | 36.91 | 2.0 |
| | [50, 1000] | 120.10 | 9.8 | 23.19 | 2.0 | 137.74 | 9.8 | 30.42 | 2.0 | 162.34 | 10.0 | 35.89 | 2.0 |
| | [50, 2000] | 124.40 | 11.0 | 22.17 | 2.0 | 145.55 | 10.8 | 29.38 | 2.0 | 179.23 | 11.0 | 35.67 | 2.0 |
| (2000, 3637428, 0.7) | [50, 500] | 200.99 | 8.8 | 41.14 | 2.0 | 253.78 | 9.0 | 57.10 | 2.0 | 284.93 | 9.0 | 67.56 | 2.0 |
| | [50, 1000] | 220.89 | 10.0 | 40.56 | 2.0 | 264.41 | 10.0 | 57.14 | 2.0 | 306.76 | 10.0 | 65.29 | 2.0 |
| | [50, 2000] | 234.08 | 11.0 | 40.20 | 2.0 | 278.06 | 11.0 | 54.08 | 2.0 | 324.74 | 11.0 | 64.08 | 2.0 |

**Table 6**
Solution time and iterations per phase on networks generated from Barabási-Albert scale-free graphs.

| | | Budget % | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1% | | | | 5% | | | | 10% | | | |
| | | Phase 1 | | Phase 2 | | Phase 1 | | Phase 2 | | Phase 1 | | Phase 2 | |
| $(n, m, h)$ | Capacity | $t$ (s) | iter. | $t$ (s) | iter. | $t$ (s) | iter. | $t$ (s) | iter. | $t$ (s) | iter. | $t$ (s) | iter. |
| (100, 1600, 20) | [50, 500] | 0.21 | 8.8 | 0.06 | 2.2 | 0.27 | 8.8 | 0.06 | 2.0 | 0.29 | 8.8 | 0.08 | 2.0 |
| | [50, 1000] | 0.22 | 9.6 | 0.05 | 2.0 | 0.26 | 9.6 | 0.06 | 2.0 | 0.28 | 9.8 | 0.07 | 2.0 |
| | [50, 2000] | 0.26 | 10.2 | 0.05 | 2.0 | 0.25 | 10.2 | 0.05 | 2.0 | 0.30 | 10.2 | 0.07 | 2.0 |
| (100, 2500, 50) | [50, 500] | 0.28 | 8.4 | 0.07 | 2.0 | 0.31 | 8.8 | 0.08 | 2.0 | 0.39 | 9.0 | 0.10 | 2.0 |
| | [50, 1000] | 0.31 | 9.6 | 0.08 | 2.2 | 0.35 | 10.0 | 0.08 | 2.0 | 0.38 | 10.0 | 0.08 | 2.0 |
| | [50, 2000] | 0.34 | 10.4 | 0.08 | 2.2 | 0.37 | 10.6 | 0.08 | 2.0 | 0.41 | 10.6 | 0.09 | 2.0 |
| (500, 40000, 100) | [50, 500] | 3.71 | 9.0 | 0.80 | 2.0 | 4.70 | 9.0 | 1.12 | 2.0 | 4.82 | 8.4 | 1.27 | 2.0 |
| | [50, 1000] | 4.21 | 10.0 | 0.82 | 2.0 | 5.05 | 10.0 | 1.09 | 2.0 | 5.33 | 9.8 | 1.25 | 2.0 |
| | [50, 2000] | 4.42 | 11.0 | 0.84 | 2.0 | 5.20 | 11.0 | 1.06 | 2.0 | 5.74 | 11.0 | 1.20 | 2.0 |
| (500, 62500, 250) | [50, 500] | 6.08 | 8.8 | 1.23 | 2.0 | 6.98 | 9.0 | 1.62 | 2.0 | 7.62 | 8.8 | 1.96 | 2.0 |
| | [50, 1000] | 6.46 | 10.0 | 1.25 | 2.0 | 6.98 | 10.0 | 1.56 | 2.0 | 8.63 | 10.0 | 1.96 | 2.0 |
| | [50, 2000] | 6.70 | 11.0 | 1.23 | 2.0 | 8.02 | 10.8 | 1.61 | 2.0 | 8.83 | 11.0 | 1.84 | 2.0 |
| (1000, 160000, 200) | [50, 500] | 15.89 | 8.8 | 3.55 | 2.0 | 18.70 | 9.0 | 4.43 | 2.0 | 20.83 | 9.0 | 5.13 | 2.0 |
| | [50, 1000] | 16.30 | 9.8 | 3.25 | 2.0 | 19.34 | 10.0 | 4.28 | 2.0 | 22.47 | 10.0 | 4.97 | 2.0 |
| | [50, 2000] | 17.77 | 11.0 | 3.24 | 2.0 | 21.37 | 11.0 | 4.27 | 2.0 | 24.50 | 11.0 | 5.04 | 2.0 |
| (1000, 250000, 500) | [50, 500] | 24.14 | 9.0 | 5.04 | 2.0 | 27.89 | 8.8 | 6.57 | 2.0 | 32.83 | 9.0 | 8.06 | 2.0 |
| | [50, 1000] | 26.18 | 10.0 | 5.08 | 2.0 | 31.12 | 10.0 | 6.67 | 2.0 | 34.88 | 10.0 | 7.87 | 2.0 |
| | [50, 2000] | 27.76 | 11.0 | 4.89 | 2.0 | 32.41 | 11.0 | 6.45 | 2.0 | 39.62 | 11.0 | 8.10 | 2.0 |
| (2000, 640000, 400) | [50, 500] | 66.28 | 8.6 | 14.10 | 2.0 | 80.06 | 9.0 | 18.89 | 2.0 | 88.64 | 8.8 | 22.19 | 2.0 |
| | [50, 1000] | 69.59 | 9.8 | 13.75 | 2.0 | 84.03 | 10.0 | 18.28 | 2.0 | 96.66 | 10.0 | 21.63 | 2.0 |
| | [50, 2000] | 77.25 | 11.0 | 14.24 | 2.0 | 96.12 | 11.0 | 19.09 | 2.0 | 105.35 | 11.0 | 21.29 | 2.0 |
| (2000, 1000000, 1000) | [50, 500] | 106.86 | 9.0 | 22.08 | 2.0 | 124.43 | 9.0 | 29.60 | 2.0 | 148.90 | 9.0 | 36.72 | 2.0 |
| | [50, 1000] | 111.45 | 10.0 | 21.39 | 2.0 | 132.23 | 10.0 | 28.45 | 2.0 | 152.96 | 10.0 | 33.78 | 2.0 |
| | [50, 2000] | 117.31 | 11.0 | 20.56 | 2.0 | 143.46 | 11.0 | 28.02 | 2.0 | 163.77 | 11.0 | 33.08 | 2.0 |

then constructs an optimal solution. By exploiting the properties of an optimal solution and the problem's network structure, our proposed algorithm runs in polynomial time and becomes one of the first known algorithms to exhibit such performance on a continuous network interdiction problem. To our knowledge, network interdiction problems have mostly focused on discrete decisions for the adversary and the literature on continuous interdiction is very limited.

We also show that the problem can be converted into a zero-sum noncooperative game which always has a pure Nash-equilibrium point. This is an interesting result from a theoretical point of view because in general zero-sum games may not always admit a pure Nash-equilibrium point (see Mazalov, 2014 for further details). Moreover, this reduction can be used to solve some versions of the problem that include side constraints on the interdictor that make a cut infeasible. These cannot be incorporated in

our proposed algorithm, but can be easily included in the game-theoretical form of the game as it enumerates all the interdictor's strategies. Naturally, this approach is not suitable for large-scale instances because it requires the enumeration of the set of strategies for the user and interdictor.

Our work shows that CMCPIP is one of the simplest forms of network interdiction problems because it admits a polynomial time algorithm, while various—and more sophisticated—variants exist in the literature. Future work can be directed towards including more realistic features on the user's path design, such as multiple objectives (Ramirez-Marquez, 2010), asymmetric information (Bayrak & Bailey, 2008), and dynamic (multi-stage) interactions between user and interdictor (Borrero, Prokopyev, & Sauré, 2015; Sefair & Smith, 2016, 2017). Moreover, it is worth exploring a compact linear programming formulation to MCPP, as its existence may allow the solution of other deterministic and stochastic problem variants. Because of the polynomial nature of the proposed algorithm, there are multiple research opportunities where CMCPIP can be used as a (fast) subproblem. For instance, it is possible to embed the proposed algorithm into an defender-attacker-defender framework, where the user first *fortifies* arcs to prevent any interdictor attempt to reduce their capacity, and then a CMCPIP game develops over the fortified network.

## Acknowledgments

This material is based upon work supported by the National Science Foundation under grant no. 2039917.

## References

Acevedo, M. A., Sefair, J. A., Smith, J. C., Reichert, B., & Fletcher Jr, R. J. (2015). Conservation under uncertainty: Optimal network protection strategies for worst-case disturbance events. *Journal of Applied Ecology, 52*(6), 1588–1597.
Ahuja, R., Magnanti, T., & Orlin, J. (1993). *Network flows: Theory, algorithms, and applications*. NJ: Prentice-Hall, Englewood Cliffs.
Aiello, W., Kushilevitz, E., Ostrovsky, R., & Rosén, A. (2000). Adaptive packet routing for bursty adversarial traffic. *Journal of Computer and System Sciences, 60*(3), 482–509.
Akgun, I., Tansel, B. C., & Wood, R. K. (2011). The multi-terminal maximum-flow network-interdiction problem. *European Journal of Operational Research, 211*(2), 241–251.
Allende, G. B., & Still, G. (2013). Solving bilevel programs with the KKT-approach. *Mathematical Programming, 138*(1), 309–332.
Altner, D. S., Ergun, O., & Uhan, N. A. (2010). The maximum flow network interdiction problem: Valid inequalities, integrality gaps, and approximability. *Operations Research Letters, 38*(1), 33–38.
Alves Pessoa, A., Di Puglia Pugliese, L., Guerriero, F., & Poss, M. (2015). Robust constrained shortest path problems under budgeted uncertainty. *Networks, 66*(2), 98–111.
Assimakopoulos, N. (1987). A network interdiction model for hospital infection control. *Computers in Biology and Medicine, 17*(6), 413–422.
Baier, G., Köhler, E., & Skutella, M. (2005). The k-splittable flow problem. *Algorithmica, 42*(3–4), 231–248.
Balas, E. (1965). An additive algorithm for solving linear programs with zero-one variables. *Operations Research, 13*(4), 517–546.
Ball, M. O., Golden, B. L., & Vohra, R. V. (1989). Finding the most vital arcs in a network. *Operations Research Letters, 8*(2), 73–76.
Barabási, A. L., & Albert, R. (1999). Emergence of scaling in random networks. *Science, 286*(5439), 509–512.
Batagelj, V., & Brandes, U. (2005). Efficient generation of large random networks. *Physical Review E, 71*(3), 036113.
Bayrak, H., & Bailey, M. D. (2008). Shortest path network interdiction with asymmetric information. *Networks, 52*(3), 133–140.
Bazaraa, M. S., Jarvis, J. J., & Sherali, H. D. (2011). *Linear programming and network flows*. John Wiley & Sons.
Berman, O., & Handler, G. Y. (1987). Optimal minimax path of a single service unit on a network to nonservice destinations. *Transportation Science, 21*(2), 115–122.
Bertsimas, D., & Sim, M. (2003). Robust discrete optimization and network flows. *Mathematical Programming, 98*(1), 49–71.
Bertsimas, D., & Sim, M. (2004). The price of robustness. *Operations Research, 52*(1), 35–53.
Bhavathankar, P., Chatterjee, S., & Misra, S. (2017). Link-quality aware path selection in the presence of proactive jamming in fallible wireless sensor networks. *IEEE Transactions on Communications, 66*(4), 1689–1704.
Borrero, J. S., Prokopyev, O. A., & Sauré, D. (2015). Sequential shortest path interdiction with incomplete information. *Decision Analysis, 13*(1), 68–98.
Church, R. L., & Scaparra, M. P. (2007). Protecting critical assets: The r-interdiction median problem with fortification. *Geographical Analysis, 39*(2), 129–146.
Clímaco, J. C. N., Pascoal, M. M. B., Craveirinha, J. M. F., & Captivo, M. E. V. (2007). Internet packet routing: Application of a k-quickest path algorithm. *European Journal of Operational Research, 181*(3), 1045–1054.
Contardo, C., & Sefair, J. A. (2021). A progressive approximation approach for the exact solution of sparse large-scale binary interdiction games. *INFORMS Journal on Computing*. Forthcoming
Dimitrov, N. B., Gonzalez, M. A., Michalopoulos, D. P., Morton, D. P., Nehme, M. V., Popova, E., … Thoreson, G. G. (2008). Interdiction modeling for smuggled nuclear material. *In Proceedings of the 49th annual meeting of the Institute of Nuclear Materials Management*.
Edmonds, J., & Karp, R. M. (1972). Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACMJ, 19*(2), 248–264.
Erdõs, P., & Rényi, A. (1976). On the evolution of random graphs. *Selected Papers of Alfréd Rényi, 2*, 482–525.
Fischetti, M., Ljubić, I., Monaci, M., & Sinnl, M. (2017). A new general-purpose algorithm for mixed-integer bilevel linear programs. *Operations Research, 65*(6), 1615–1637.
Fischetti, M., Ljubić, I., Monaci, M., & Sinnl, M. (2018). On the use of intersection cuts for bilevel optimization. *Mathematical Programming, 172*(1–2), 77–103.
Frederickson, G. N., & Solis-Oba, R. (1999). Increasing the weight of minimum spanning trees. *Journal of Algorithms, 33*(2), 244–266.
Fulkerson, D. R., & Harding, G. C. (1977). Maximizing the minimum source-sink path subject to a budget constraint. *Mathematical Programming, 13*(1), 116–118.
Gabow, H. N. (1983). Scaling algorithms for network problems. *In Proceedings of the IEEE 24th Annual Symposium on Foundations of Computer Science (SFCS)*, 248–258.
Ghaffarinasab, N., & Motallebzadeh, A. (2018). Hub interdiction problem variants: Models and metaheuristic solution algorithms. *European Journal of Operational Research, 267*(2), 496–512.
Ghare, P. M., Montgomery, D. C., & Turner, W. C. (1971). Optimal interdiction policy for a flow network. *Naval Research Logistics (NRL), 18*(1), 37–45.
Golden, B. (1978). A problem in network interdiction. *Naval Research Logistics (NRL), 25*(4), 711–713.
Hu, T. C. (1961). The maximum capacity route problem. *Operations Research, 9*(6), 898–900.
Ichimori, T., Ishii, H., & Nishida, T. (1981). Weighted minimax real-valued flows. *Journal of the Operations Research Society of Japan, 24*(1), 52–60.
Israeli, E., & Wood, R. K. (2002). Shortest-path network interdiction. *Networks, 40*(2), 97–111.
Kar, K., Kodialam, M., & Lakshman, T. V. (2000). Minimum interference routing of bandwidth guaranteed tunnels with MPLS traffic engineering applications. *IEEE Journal on Selected Areas in Communications, 18*(12), 2566–2579.
Laroche, P., Marchetti, F., Martin, S., & Roka, Z. (2014). Bipartite complete matching vertex interdiction problem: Application to robust nurse assignment. In *In 2014 international conference on control, decision and information technologies (coDIT)* (pp. 182–187). IEEE.
Liberatore, F., Scaparra, M. P., & Daskin, M. S. (2011). Analysis of facility protection strategies against an uncertain number of attacks: The stochastic r-interdiction median problem with fortification. *Computers & Operations Research, 38*(1), 357–366.
Liberatore, F., Scaparra, M. P., & Daskin, M. S. (2012). Hedging against disruptions with ripple effects in location analysis. *Omega, 40*(1), 21–30.
Mageswari, R. U., & Baulkani, S. (2020). Routing technique for data transmission under jammer in multi-hop wireless network. *Journal of Ambient Intelligence and Humanized Computing, 12*, 3705–3713.
Magnanti, T. L., & Wolsey, L. A. (1995). Optimal trees. *Handbooks in Operations Research and Management Science, 7*, 503–615.
Malaviya, A., Rainwater, C., & Sharkey, T. (2012). Multi-period network interdiction problems with applications to city-level drug enforcement. *IIE Transactions, 44*(5), 368–380.
Martins, E. Q. V., & Santos, J. L. E. (1997). An algorithm for the quickest path problem. *Operations Research Letters, 20*(4), 195–198.
Matuschke, J., McCormick, S. T., Oriolo, G., Peis, B., & Skutella, M. (2017). Protection of flows under targeted attacks. *Operations Research Letters, 45*(1), 53–59.
Mazalov, V. (2014). *Mathematical game theory and applications*. John Wiley & Sons.
McMasters, A. W., & Mustin, T. M. (1970). Optimal interdiction of a supply network. *Naval Research Logistics (NRL), 17*(3), 261–268.
Medhi, D., & Ramasamy, K. (2017). *Network routing: Algorithms, protocols, and architectures, second edition*. Cambrigde, MA: Morgan Kaufmann Publishers.
Mohammadi, A., & Tayyebi, J. (2019). Maximum capacity path interdiction problem with fixed costs. *Asia-Pacific Journal of Operational Research, 36*(4). 1950018.1–1950018.21
Morton, D. P., Pan, F., & Saeger, K. (2007). Models for nuclear smuggling interdiction. *IIE Transactions, 39*, 3–14.
Othman, M. F., & Shazali, K. (2012). Wireless sensor network applications: A study in environment monitoring system. *Procedia Engineering, 41*, 1204–1210.
Pan, F. (2005). *Stochastic network interdiction: Models and methods*. PhD dissertation.
Pan, F., & Morton, D. P. (2008). Minimizing a stochastic maximum-reliability path. *Networks, 52*(3), 111–119.
Pollack, M. (1960). The maximum capacity through a network. *Operations Research, 8*(5), 733–736.
Punnen, A. P. (1991). A linear time algorithm for the maximum capacity path problem. *European Journal of Operational Research, 53*(3), 402–404.

Radzik, T. (2013). Fractional combinatorial optimization. In *In handbook of combinatorial optimization* (pp. 1311–1355). New York, NY: Springer.
Ramamoorthy, P., Jayaswal, S., Sinha, A., & Vidyarthi, N. (2018). Multiple allocation hub interdiction and protection problems: Model formulations and solution approaches. *European Journal of Operational Research, 270*(1), 230–245.
Ramaswamy, R., Orlin, J. B., & Chakravarti, N. (2005). Sensitivity analysis for shortest path problems and maximum capacity path problems in undirected graphs. *Mathematical Programming, 102*(2), 355–369.
Ramirez-Marquez, J. E. (2010). A bi-objective approach for shortest-path network interdiction. *Computers & Industrial Engineering, 59*(2), 232–240.
Rocco, S., Claudio, M., Ramirez-Marquez, J. E., Salazar, A., Daniel, E., & Zio, E. (2010). A flow importance measure with application to an italian transmission power system. *International Journal of Performability Engineering, 6*(1).
Royset, J. O., & Wood, R. K. (2007). Solving the bi-objective maximum-flow network-interdiction problem. *INFORMS Journal on Computing, 19*(2), 175–184.
Salmeron, J., Wood, R. K., & Baldick, R. (2004). Analysis of electric grid security under terrorist threat. *IEEE Transactions on power systems, 19*(2), 905–912.
Schrijver, A. (1998). *Theory of linear and integer programming*. John Wiley & Sons.
Schulze, M. (2011). A new monotonic, clone-independent, reversal symmetric, and condorcet-consistent single-winner election method. *Social Choice and Welfare, 36*(2), 267–303.
Sefair, J. A., & Smith, J. C. (2016). Dynamic shortest-path interdiction. *Networks, 68*(4), 315–330.
Sefair, J. A., & Smith, J. C. (2017). Exact algorithms and bounds for the dynamic assignment interdiction problem. *Naval Research Logistics, 64*(5), 373–387.
Sefair, J. A., Smith, J. C., Acevedo, M. A., & Fletcher, J. R. J. (2017). A defender-attacker model and algorithm for maximizing weighted expected hitting time with application to conservation planning. *IISE Transactions, 49*(12), 1112–1128.
Smith, J. C., Prince, M., & Geunes, J. (2013). Modern network interdiction problems and algorithms. In *In handbook of combinatorial optimization* (pp. 1949–1987). Springer New York.
Smith, J. C., & Song, Y. (2020). A survey of network interdiction models and algorithms. *European Journal of Operational Research, 283*(3), 797–811.
von Stackelberg, H. (1952). *The theory of the market economy*. London: William Hodge and Co.
Tahernejad, S., Ralphs, T. K., & DeNegre, S. T. (2020). A branch-and-cut algorithm for mixed integer bilevel linear optimization problems and its implementation. *Mathematical Programming Computation, 12*(4), 529–568.
Tragoudas, S. (2001). The most reliable data-path transmission. *IEEE Transactions on Reliability, 50*(3), 281–285.
Washburn, A., & Wood, R. K. (1995). Two-person zero-sum games for network interdiction. *Operations Research, 43*(2), 243–251.
Wood, R. K. (1993). Deterministic network interdiction. *Mathematical and Computer Modelling, 17*(2), 1–18.
Xu, N. (2002). A survey of sensor network applications. *IEEE communications magazine, 40*(8), 102–114.
Xu, P., & Wang, L. (2014). An exact algorithm for the bilevel mixed integer linear programming problem under three simplifying assumptions. *Computers & Operations Research, 41*, 309–318.
Xue, Y., & Nahrstedt, K. (2004). Providing fault-tolerant ad hoc routing service in adversarial environments. *Wireless Personal Communications, 29*(3–4), 367–388.
Zeng, B., & An, Y. (2014). Solving bilevel mixed integer program by reformulations and decomposition. *Optimization Online*. http://www.optimization-online.org/DB_FILE/2014/07/4455.pdf
Zenklusen, R. (2010). Matching interdiction. *Discrete Applied Mathematics, 158*(15), 1676–1690.
Zenklusen, R. (2015). An $o(1)$-approximation for minimum spanning tree interdiction. *In Proceedings of the IEEE 56th Annual Symposium on Foundations of Computer Science (SFCS)*, 709–728.