Fröhlich, Nicolas; Ruzika, Stefan



# Interdicting facilities in tree networks

TOP

TOP (2022) 30:95–118
https://doi.org/10.1007/s11750-021-00600-6

ORIGINAL PAPER


# Interdicting facilities in tree networks

**Nicolas Fröhlich**[1] · **Stefan Ruzika**[1]

Received: 21 January 2021 / Accepted: 19 April 2021 / Published online: 10 May 2021


**Abstract**
This article investigates a network interdiction problem on a tree network: given a subset of nodes chosen as facilities, an interdictor may dissect the network by removing a size-constrained set of edges, striving to worsen the established facilities best possible. Here, we consider a reachability objective function, which is closely related to the covering objective function: the interdictor aims to minimize the number of customers that are still connected to any facility after interdiction. For the covering objective on general graphs, this problem is known to be NP-complete (Fröhlich and Ruzika In: On the hardness of covering-interdiction problems. Theor. Comput. Sci., 2021). In contrast to this, we propose a polynomial-time solution algorithm to solve the problem on trees. The algorithm is based on dynamic programming and reveals the relation of this location-interdiction problem to knapsack-type problems. However, the input data for the dynamic program must be elaborately generated and relies on the theoretical results presented in this article. As a result, trees are the first known graph class that admits a polynomial-time algorithm for edge interdiction problems in the context of facility location planning.

**Keywords** Network interdiction · Network location · Dynamic programming · Trees

**Mathematics Subject Classification** 90C27 · 90B80 · 90C39 · 05C05

✉ Nicolas Fröhlich
froehlich@mathematik.uni-kl.de

Stefan Ruzika
ruzika@mathematik.uni-kl.de

[1] Technische Universität Kaiserslautern, Gottlieb-Daimler-Str. 48, 67663 Kaiserslautern, Germany

## 1 Introduction

Location science is a well-established research area that fascinates practitioners as well as theoreticians. The most basic and likewise famous problems are known as the *p-median*, the *p-center*, and the *covering problem* (see Laporte et al. 2015). Nowadays research in this field mostly extends these basic models, often by coupling other fields of theoretical or practical interest, for example, *multicriteria location planning* (Kalcsics et al. 2014; Alzorba et al. 2015), *robust location planning* (Baron et al. 2011; Carrizosa and Nickel 2003), or combining *locational decisions* and *routing* (see for example the survey Drexl and Schneider 2015).

In this article, we study an innovative location-interdiction problem on tree networks. Given a tree $T = (V, E)$ and a set of facilities $S \subseteq V$, we are interested in finding a set of $r$ edges $R \subseteq E$ such that the number of nodes $V \setminus S$ that are still connected to some facility $s \in S$ after removing the edge set $R$ from the network is minimal. Obviously, this is equivalent to maximizing the number of non-reachable nodes after interdiction. Intuitively speaking, we aim to find an interdiction strategy such that as many nodes as possible are contained in a connected component not including a facility after interdiction. In the context of facility location planning, studying a reachability objective function is a novel approach, as this objective function is useless in a setup without interdiction: it suffices to place one facility in each connected component, no matter where it is placed. However, in the case that an interdictor may dissect the network, this new objective function becomes attractive. It can be seen as a special case of the well-known covering objective function (cf. Laporte et al. 2015), where the coverage radius is sufficiently large to cover the whole graph. Interdiction problems usually describe the interdictor's perspective, and thus, the facilities are already established, and the interdictor has full knowledge about the locator's choice. Although there are many observable applications in which the interdictor takes the role of an attacker destroying the infrastructure, there are also valuable applications in which the interdictor is the defender, for example, the prevention of poison or virus spreading, the destruction of smugglers' networks, or the mitigation of damage caused by floods (see for example Assimakopoulos 1987; Morton et al. 2007; Soleimani-Alyar et al. 2016). Besides, interdiction problems are often used to reveal weak spots of a system.

The analysis of interdiction problems dates back to the second half of the last century and has gained great attraction within the last few years. The primary research mainly has focused on *maximum flow interdiction* (cf. Burch et al. 2003; Chestnut and Zenklusen 2017; Wood 1993; Zenklusen 2010) and *shortest path interdiction* (cf. Bar-Noy et al. 1995; Boros et al. 2006; Israeli and Wood 2002; Khachiyan et al. 2008), but in the meantime, the list of combinatorial problems studied in the context of interdiction has grown rapidly (cf. Chestnut and Zenklusen 2016; Dinitz and Gupta 2013; Furini et al. 2019; Zenklusen 2014, 2010). A recent survey by Smith and Song (2020) captures the research in this area. However, work in the context of location-interdiction problems is sparse. In contrast to our approach, most articles in this area concentrate on interdicting the

facilities themselves (cf. Aksen et al. 2010; Church et al. 2004; Mahmoodjanloo et al. 2016). Commonly, the models are stated as (occasionally multilevel) integer programs. An extension is given by allowing *fortification*, a variant in which several facilities can be saved from interdiction, see for example Scaparra and Church (2008). Recently, the interdiction of facilities in a hub network has become an arising topic (see Ghaffarinasab and Atayi 2018; Ramamoorthy et al. 2018; Ullmert et al. 2020). Due to the complexity, most problems are solved by heuristic or metaheuristic approaches, for example using genetic algorithms or simulated annealing. Only a few exact algorithms to solve location-interdiction problems on general graphs apart from total enumeration are known, and the computation times are—although many times faster—still poor. This even increases the need to search for special cases that can be solved exactly in a reasonable time (nevertheless, there are exact solution methods for solving other interdiction problems in reasonable time, see for example Lozano and Smith (2017) or Baggio et al. (2021)).

On the contrary, location problems with edge interdiction gained little attention in the literature. This is quite surprising since the removal of links in networks is a noteworthy kind of interdiction in many applications such as logistics (Adenso-Diaz et al. 2018), ecology (Streib et al. 2020), or digital communications (Ahmad et al. 2017). In Fröhlich and Ruzika (2020, 2021), a systematical classification scheme for location-problems with edge interdiction is introduced, and the complexity state of the different settings is investigated. They show that in general graphs, the problem of the follower is NP-complete, whereas the problem of the leader is even $\Sigma_2^p$-complete. In Bazgan et al. (2010, 2013), inapproximability results for location-interdiction problems with weighted center and median objectives are presented.

Anyhow, there are only few interdiction problems for which approximation algorithms are known, and the gap between these approximations and the best known bound is evident (cf. Chestnut and Zenklusen 2016, 2017; Pan and Schild 2016; Phillips 1993; Zenklusen 2010, 2014). This motivates studying the threshold between exact solvability and hardness in more detail. A natural approach is to study the problems in restricted graph classes. In Shen and Smith (2011), the authors develop an exact polynomial-time algorithm based on dynamic programming to solve the critical node problem on trees and series-parallel graphs. The goal is to identify a subset of the nodes whose removal will maximally disconnect the graph. Recently, Aringhieri et al. (2019) studied a distance based variant of this problem. The authors provide polynomial and pseudo-polynomial algorithms for paths, trees and series–parallel graphs. In contrast to our work, these problems—although closely related—do not include a set of existing facilities that should be separated.

The contribution of our paper is the first polynomial-time algorithm for interdicting edges in the context of facility location planning. The presented algorithm includes several stages: As a first step, the given problem instance is decomposed into several smaller subproblems that can be solved independently. However, a modified knapsack-type approach is needed to combine the partial solutions. Each of the subproblems is solved using a dynamic programming approach whose input data is elaborately generated during the execution of the algorithm. The functionality of the dynamic programming approach is based on a subtle choice of candidates. Finally, the polynomial

runtime is ensured by bounding the number of iterations needed to compute the optimal value by a multiple of the number of edges in the tree.

The rest of the paper is structured as follows: Sect. 2 presents the formal definition and basic notation of the problem. The polynomial-time solution algorithm is developed in Sect. 3. We summarize the results and give a short outlook in Sect. 4.

## 2 Definitions and problem setup

Throughout this article, we consider a tree $T = (V, E)$. An *interdiction strategy* $R$ is a subset of the edge set $E$ of the tree. We say that $R$ satisfies the *interdiction budget* $r$, if $|R| \leq r$. The network after removing the edge set $R$ is denoted by $T \setminus R$, which is also referred to as the *interdicted network*. Given a set $S \subseteq V$ of facilities, we say that a node $v \in V$ is *reachable from* $S$, if there exists a path from $v$ to any facility in $S$. If this is not the case, we say that $v$ is *non-reachable from* $S$. If the context is clear, we omit the supplement "*from S*". We also refer to the nodes in $V \setminus S$ as *customers*. The interdictor strives to maximize the number of non-reachable nodes by deleting arcs. This interdictor's problem can be formally stated as:

**Problem 1**

Input: A tree $T = (V, E)$, a set $S \subseteq V$ of facilities, an integral interdiction budget $r > 0$

Task: Find a set $R \subseteq E$ of edges with $|R| \leq r$ such that

$$f_{T\setminus R}(S) := |\{v \in V \mid v \text{ is non-reachable from } S \text{ in } T \setminus R\}|$$

is maximized.

Instead of maximizing the number of non-reachable nodes, the problem can be equivalently stated minimizing the number of reachable nodes. It further can be formulated as an integer program. For this purpose, we introduce binary decision variables $x_v, v \in V \setminus S$, indicating whether a customer is reachable or not, and binary variables $a_e, e \in E$, indicating whether the edge $e$ is interdicted or not. By $P_{uv}$, we denote the edge set of the unique path connecting nodes $u$ and $v$. The first constraint ensures the correct assignment of the $x$-variables: for $v \in V \setminus S$, either node $v$ is reachable or the path to every facility $s \in S$ must be interdicted. The second constraint guarantees the interdiction budget restriction.

$$\min \quad \sum_{v \in V \setminus S} x_v \tag{1a}$$

$$\text{s.t.} \quad x_v + \sum_{e \in P_{sv}} a_e \geq 1 \qquad v \in V \setminus S, s \in S \tag{1b}$$

$$\sum_{e \in E} a_e \leq r \tag{1c}$$

$$a_e \in \{0,1\} \qquad e \in E \tag{1d}$$

$$x_v \geq 0 \qquad v \in V \setminus S \tag{1e}$$

Clearly, all decision variables should be binary. However, requiring $x$ to be non-negative suffices to ensure integrality of an optimal solution, as $x_v, v \in V \setminus S$, is minimized, and by constraint (1b, 1c, 1d, 1e) either greater than or equal to 0 or 1. Note that dropping the binary constraint on decision variable $a_e, e \in E$, and bounding it from below and above by 0 and 1 does not provide an integral optimal solution. A counterexample is given by the network depicted in Fig. 1 for interdiction budget 2: one (of several) optimal integral solutions is to interdict the edges (1, 2) and (2, 4) with 5 remaining reachable customers. In contrast, a fractional optimal solution is given by $a_{(1,2)} = a_{(5,6)} = a_{(8,9)} = \frac{2}{3}$ and associated solution value $2\frac{1}{3}$, as $x_v = \frac{1}{3}$ for every node $v \in V \setminus S$. In the remainder of this article, our goal is to show polynomial-time solvability of this problem presenting a constructive solution algorithm.

## 3 Algorithmic approach

First, we show how to decompose the given tree into several subproblems, also called *clusters*, each of which can be solved efficiently by the algorithm stated in the second part of this section (see Theorem 3). The partial solutions can then be composed to an overall solution by an adapted knapsack-type approach. The clusters are constructed as follows:

**Definition 1** For an arbitrary node $v \in V \setminus S$, we denote by $\mathsf{CF}(v) \subseteq S$ the set of *closest facilities connected to* $v$, i.e., for every facility $s \in \mathsf{CF}(v)$, the unique path connecting $v$ and $s$ contains no other facility than $s$.

**Definition 2** A set of nodes $V' \subseteq V \setminus S$ is said to be *cluster inducing*, if

(i) for every pair of nodes $u, v \in V'$, it holds that $\mathsf{CF}(u) = \mathsf{CF}(v)$, and
(ii) $V'$ is inclusion-wise maximal, i.e., there is no superset $V'' \supset V'$ such that $\mathsf{CF}(u) = \mathsf{CF}(v)$ for all $u, v \in V''$.

**Definition 3** The *set of clusters* is defined by

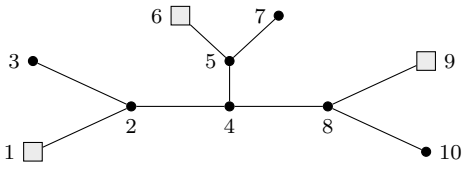


**Fig. 1** An instance with different integral / fractional optimal solutions for $r = 2$. *Circles* represent customers, *squares* represent facilities

$$\mathcal{C} := \{T[V' \cup \mathsf{CF}(V')] \mid V' \subseteq V \setminus S \text{ is cluster inducing}\},$$

where $\mathsf{CF}(V') := \mathsf{CF}(v)$ for some $v \in V'$ and $T[X]$ denotes the tree induced by $X \subseteq V$.

In Fig. 2, a network and the corresponding clusters are depicted. Note that by construction of a cluster, the facilities are located on its leaves.

Without loss of generality, we label the clusters by $C_1, C_2, \ldots, C_k$ in an arbitrary but fixed order. For $C_i \in \mathcal{C}$, we refer to the set of nodes, edges, and facilities contained in $C_i$ as $V_i \subseteq V$, $E_i \subseteq E$, and $S_i \subseteq S$, respectively. Note that by definition, a facility may be contained in several clusters, whereas the sets of edges $E_1, ..., E_k$ are disjoint sets.

For a fixed interdiction strategy $R \subseteq E$, it holds

$$\begin{aligned} f_{T\setminus R}(S) &= |\{v \in V \mid v \text{ is non-reachable from } S \text{ in } T \setminus R\}| \\ &= \sum_{i=1}^{k} |\{v \in V_i \mid v \text{ is non-reachable from } S \text{ in } T \setminus R\}| \\ &= \sum_{i=1}^{k} |\{v \in V_i \mid v \text{ is non-reachable from } \mathsf{CF}(v) \text{ in } T \setminus R\}| \\ &= \sum_{i=1}^{k} |\{v \in V_i \mid v \text{ is non-reachable from } S_i \text{ in } C_i \setminus (R \cap E_i)\}|. \end{aligned}$$

The third equality follows, since by definition of $\mathsf{CF}(v)$, if $v$ has no path to any facility of $\mathsf{CF}(v)$, $v$ does not have a path to any other facility of $S$ in the interdicted network $T \setminus R$. As a consequence, deciding whether a node of the cluster $C_i$ is reachable or non-reachable only depends on the part of the interdiction strategy that hits the edge set $E_i$.

Assume that one knows that in an optimal solution of Problem 1, $r_i^*$ edges of cluster $C_i$ are interdicted, i.e., $|R \cap E_i| = r_i^*$, where $\sum_{i=1}^{k} r_i^* \leq r$. In this case, the optimal solution of the overall problem can be obtained by composing the optimal solutions within the clusters with respect to the particular distribution of the interdiction budget. More precisely, it follows

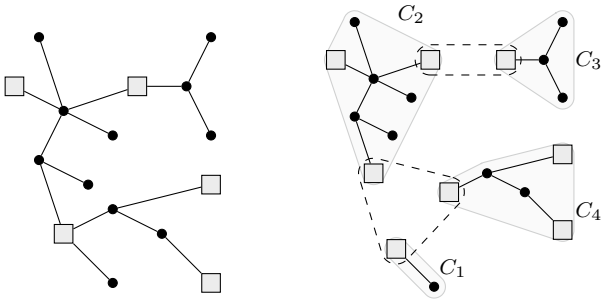


**Fig. 2** An instance of Problem 1 and corresponding clusters

$$\max_{R\subseteq E,|R|\leq r} f_{T\setminus R}(S) = \sum_{i=1}^{k} \max_{R_i\subseteq E_i,|R_i|=r_i^*} |\{v \in V_i \mid v \text{ non-reachable from } S_i \text{ in } C_i \setminus R_i\}|. \tag{2}$$

Thus, the overall solution is composed of the clusters' solutions. The cluster's problem is formally given by:

**Problem 2** (Subproblem CLUSTER − IP)

Input: A cluster $C_i$ according to Definition 3, an integral interdiction budget $r' \leq r$

Task: Find a set $R_i \subseteq E_i$ of edges with $|R_i| \leq r'$ such that

$$f_{C_i\setminus R_i}(S_i) := |\{v \in V_i \mid v \text{ is non-reachable from } S_i \text{ in } C_i \setminus R_i\}|$$

is maximized.

Before stating an algorithm to solve CLUSTER − IP efficiently, we show how these partial solutions can be combined to an optimal solution of the overall problem. To this end, assume that for some fixed interdiction budget $r' \leq r$, one can solve problem CLUSTER − IP stated in Problem 2 on a single cluster $C_i$ efficiently. According to (2), the problem reduces to finding the optimal choices of $r_1^*, ..., r_k^*$ requiring $\sum_{i=1}^{k} r_i^* \leq r$. To cope with the latter problem, we introduce a variant of the knapsack problem with additional constraints grouping the items and restricting the number of chosen items per group.

**Problem 3** (Knapsack with Bucket Constraints (KNAP-BC))

Input: A set of items $I = \{1, ..., n\}$, a weight function $w : I \longrightarrow \mathbb{N}$, a profit function $p : I \longrightarrow \mathbb{N}$, a bucket assignment $\beta : I \to \{1, ..., k\}$, and a weight bound $W$

Task: Find a set $I' \subseteq I$ of items with $\sum_{i\in I'} w(i) \leq W$ and $\beta(i) \neq \beta(j)$ for all $i, j \in I', i \neq j$ such that $\sum_{i\in I'} p(i)$ is maximized.

It can be easily seen that this problem models Problem 1, provided that CLUSTER − IP can be solved efficiently for all clusters $C_i \in \mathcal{C}$ and all interdiction budgets $r' \leq r$. For this purpose, let $R_i^*(r')$ denote an optimal solution in cluster $C_i$ with respect to budget $r'$ and define

$$z_i^*(r') := |\{v \in V_i \mid v \text{ is non-reachable from } S_i \text{ in } C_i \setminus R_i^*(r')\}|$$

to be the corresponding optimal solution value of CLUSTER − IP. Every cluster represents a bucket, and for every suitable interdiction budget $r' \leq r$, there is an item with profit $z_i^*(r')$ and weight $r'$. Clearly, only one solution of each cluster can be part of the solution of Problem 1, which justifies the bucket constraints.

KNAP-BC can be solved in time $\mathcal{O}(nW)$ by the following adaption of the well-known dynamic programming approach to solve the knapsack problem: Without loss of generality, assume that the items are ordered block by block according to their buckets, i.e., the first block of items belongs to bucket $B_1$, the next block of entries to bucket $B_2$, and so on. We define a function $\alpha : I \longrightarrow \{0, 1, ..., n\}$ mapping an item to the last item of the previous block, i.e., for $i \in I$, we define

$$\alpha(i) := \begin{cases} \max\{j \in I \mid \beta(j) = \beta(i) - 1\}, & \text{if } \beta(i) \geq 2 \\ 0, & \text{else.} \end{cases}$$

All we have to ensure when iteratively increasing the set of allowed items during dynamic programming is that a new solution containing the new item is not based on the solution of any other item of that bucket. The pseudocode to solve KNAP-BC is given in Algorithm 1. The value $P_{(i,j)}$ represents the optimal solution value restricted to items $\{0, 1, ..., i\}$ and budget limit $j$. A solution can be obtained by tracking the corresponding items that are packed. As here the maximum weight $W$ is bounded by $r \leq |E|$, Algorithm 1 runs in polynomial time with respect to the input size of Problem 1.

**Algorithm 1:** Knapsack with Bucket Constraints

**Input** : A set of items $I = \{1, ..., n\}$, a set of buckets $B_1, ..., B_k$, a weight function $w: I \longrightarrow \mathbb{N}$, a profit function $p: I \longrightarrow \mathbb{N}$, a bucket assignment $\beta: I \to \{1, ..., k\}$, and a weight bound $W$

**Output** : The optimal solution value of KNAP-BC

```
for i ∈ {0, 1, ..., n} do
    for j ∈ {0, 1, ..., n} do
        P_(i,j) ← 0
for i ∈ {1, ..., n} do
    for j ∈ {1, ..., W} do
        if P_(i−1,j) ≥ P_(α(i),j−w(i)) + p(i) then
            P_(i,j) ← P_(i−1,j)
        else
            P_(i,j) ← P_(α(i),j−w(i)) + p(i)
return P_(n,W)
```

**Theorem 1** *Algorithm 1 computes the optimal solution of* KNAP-BC *correctly in time* $\mathcal{O}(nW)$.

***Proof*** Analogous to the proof of the general 0-1-knapsack problem (cf. Gilmore and Gomory 1966) and the fact that item $i \in I$ may only be chosen if no other item $j \in I, j \neq i$, with $\beta(i) = \beta(j)$ is already packed. □

***Example 1*** Recall the network given in Fig. 2. The items of the corresponding knapsack instance can be deduced from the partial solutions within the clusters:

| $C_1$ | | $C_2$ | | $C_3$ | | $C_4$ | |
|---|---|---|---|---|---|---|---|
| $r'$ | $z_1^*(r')$ | $r'$ | $z_2^*(r')$ | $r'$ | $z_3^*(r')$ | $r'$ | $z_4^*(r')$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 3 | 1 | 0 |
| 2 | – | 2 | 2 | 2 | – | 2 | 1 |
| 3 | – | 3 | 5 | 3 | – | 3 | 2 |

For a total interdiction budget of 5, it is optimal to interdict one edge in $C_1$, three edges in $C_2$, and one edge in $C_3$.

### 3.1 Solving CLUSTERIP by dynamic programming

In this section, we develop an algorithm to solve CLUSTER – IP. In combination with Algorithm 1, this solves Problem 1. By assumption, the tree considered in this subsection is a cluster according to Definition 3. From now on, let $T = (V, E)$ denote the corresponding subtree of the cluster and let $S \subseteq V$ be the set of facilities which are, by construction, located on the leaves of $T$. Obviously, removing $|S|$ edges suffices to separate all non-facility-nodes of $T$ from $S$ by deleting the incident edges of $S$. Thus, without loss of generality, we may assume $r' \leq |S|$. We further require every interdiction strategy to be *proper*, i.e., a solution does not contain any redundant edges. An edge $e$ is redundant in $R \subseteq E$ if the set of non-reachable nodes with interdiction strategy $R$ equals the set of non-reachable nodes with respect to interdiction strategy $R \setminus \{e\}$. An arbitrary interdiction strategy $R \subseteq E$ can be easily checked for properness and transformed into a proper one.

We now give an alternative formulation of CLUSTER – IP. More precisely, instead of finding an optimal set of edges to interdict, one may also look for the nodes that should become non-reachable. Note that such a set of nodes $V' \subseteq V \setminus S$ induces a unique interdiction strategy and vice versa, as all incident edges of $V'$ must be interdicted, assuming no redundant edges. Let $\delta(V') := \{e = (u, v) \in E \mid u \in V', v \notin V'\}$ denote the set of outgoing edges of $V'$ and let $\mathcal{V}$ denote the set of all node sets $V' \subseteq V \setminus S$ such that $T[V']$ is connected. Then, Problem 2 can be reformulated to Problem 4. It is easy to see that these two problems are equivalent.

**Problem 4**

Input: A cluster $T$ according to Definition 3, an integral interdiction budget $r' \leq r$

Task: Find a set $\mathcal{V}' \subseteq \mathcal{V}$ satisfying

(1) $\sum_{V' \in \mathcal{V}'} |\delta(V')| \leq r'$ and
(2) $V' \cap V'' = \emptyset$ for all $V', V'' \in \mathcal{V}'$ with $V' \neq V''$

such that $\sum_{V' \in \mathcal{V}'} |V'|$ is maximized.

This problem can be seen as a variant of knapsack with conflict constraints (cf. Pferschy and Schauer 2009). However, the number of items (namely the cardinality of $\mathcal{V}$) is not polynomially bounded in the input size of the original problem. Thus, a naive dynamic programming approach may—provided that the conflict constraints can be addressed—lead to an exponential time algorithm. Instead, we investigate some optimality conditions of Problem 4 to facilitate an efficient dynamic program.

**Lemma 1** *There exists an optimal solution $\mathcal{V}^* = \{V_1, ..., V_l\} \subseteq \mathcal{V}$ of Problem 4 such that $\delta(V_i) \cap \delta(V_j) = \emptyset$ for all $V_i, V_j \in \mathcal{V}^*$ with $V_i \neq V_j$. Especially, all edges in $\delta(V_i) \cap \delta(V_j)$ are redundant.*

***Proof*** Suppose that $\delta(V_i) \cap \delta(V_j) = \{e\}$ for two sets $V_i, V_j \in \mathcal{V}^*$ with $V_i \neq V_j$. Replacing the sets $V_i$ and $V_j$ in $\mathcal{V}^*$ by $V_i \cup V_j$ remains feasible and provides the same objective value. Especially, it holds $\delta(V_i \cup V_j) = (\delta(V_i) \cup \delta(V_j)) \setminus \{e\}$. □

Analogously to the properness of an interdiction strategy, we call a solution $\mathcal{V}^* = \{V_1, ..., V_l\} \subseteq \mathcal{V}$ of Problem 4 *proper*, if no two sets $V_i, V_j \in \mathcal{V}^*, V_i \neq V_j$, in $\mathcal{V}^*$ can be merged without loosing feasibility or decreasing the objective value.

**Lemma 2** (Necessary Condition) *Let $\mathcal{V}^* = \{V_1, ..., V_l\} \subseteq \mathcal{V}$ be optimal for Problem 4, assume $\mathcal{V}^*$ to be proper, and choose $e = (u, v) \in \delta(V_i)$ for some arbitrary $i \in \{1, ..., l\}$. Let $u \in V_i$ and $v \notin V_i$. Then, either* $\deg(v) \geq 3$, *or $v$ is a facility.*

***Proof*** Assume that this is not the case, i.e., for some $i \in \{1, ..., l\}$, there is an edge $e = (u, v) \in \delta(V_i)$ with $u \in V_i$ and $v \notin V_i$ such that $\deg(v) \leq 2$. Assume that $v \notin S$. As $\mathcal{V}^*$ is proper, it follows $v \notin V_j$ for all $j \in \{1, ..., l\}$ by Lemma 1. This is immediately a contradiction to the optimality of $\mathcal{V}^*$, as replacing $V_i$ by $V_i \cup \{v\}$ strictly improves the objective value while $\sum_{V' \in \mathcal{V}'} |\delta(V')|$ does not increase. □

**Lemma 3** (*Necessary Condition*) *Let $\mathcal{V}^* \subseteq \mathcal{V}$ be optimal for Problem 4 and let $\mathcal{V}^*$ be proper. Then, any set $V_i \in \mathcal{V}^*$ is inclusion-wise maximal, i.e., there is no strict superset $V_i' \supset V_i$ with $|\delta(V_i')| \leq |\delta(V_i)|$.*

***Proof*** This follows as replacing $V_i$ by $V_i'$ in $\mathcal{V}^*$ strictly improves the objective value, provided that $V_i' \cap V_j = \emptyset$ for all $V_j \in \mathcal{V}^* \setminus \{V_i\}$, while the interdiction costs are not increased. So, suppose that $V_i' \cap V_j \neq \emptyset$ for some $V_j \in \mathcal{V}^* \setminus \{V_i\}$. Since $\mathcal{V}^*$ is proper, there must be at least one additional node $v \in V \setminus S$ such that $v \notin V_i$ for all $i \in \{1, ..., l\}$ on the (unique) path between the sets $V_i$ and $V_j$. Without loss of generality, choose $v$ adjacent to $V_i$. Then, replacing $V_i$ in solution $\mathcal{V}^*$ by $V_i' \setminus V_j$ strictly improves the objective value while maintaining feasibility, which is a contradiction. □

Lemma 3 motivates studying the following related Problem 5: given a node $v \in V \setminus S$ and a fixed interdiction budget $r'$, find an inclusion-wise maximal set $V' \in \mathcal{V}$ such that $v \in V'$ and $|\delta(V')| \leq r'$. Note that however, one can not conclude

that in an optimal solution $\mathcal{V}^* \subseteq \mathcal{V}$ of Problem 4, every $V_i$ is an optimal solution of Problem 5 for some node $x \in V \setminus S$ and some budget $r' \leq r$. In fact, this is not true, as we show in Example 5. Nonetheless, the analysis of Problem 5 prepares the solution algorithm of Problem 4.

**Problem 5**

Input: A cluster $T$ according to Definition 3, an integral interdiction budget $r' \leq r$ and a node $x \in V \setminus S$
Task: Find a set $V' \subseteq V \setminus S$ such that

(1) $x \in V'$,
(2) $\delta(V') \leq r'$, and
(3) $T[V']$ is connected

such that $|V'|$ is maximized.

In Problem 5, every facility in $S$ must be cut off from node $x$ in order to make $x$ non-reachable. We also refer to the set of non-reachable nodes as *non-reachable area* as the whole induced subgraph is non-reachable by property (3). To ensure connectedness of the induced subgraph of non-reachable nodes, every interdicted edge must lie on a path connecting $x$ and a facility $s \in S$ to maintain feasibility. In the sequel, our goal is to further reduce the set of candidate edges considered for interdiction.

**Definition 4** Define

$$\mathsf{CAND}(x) = \{e \in E \mid \exists u, v \in S : e = \arg\max_{(i,j) \in E(P_{ux}) \cap E(P_{vx})} \max\{\mathrm{dist}(x,i), \mathrm{dist}(x,j)\}\},$$

where $\mathrm{dist}(v_1, v_2)$ denotes the length of the shortest $v_1$-$v_2$-path with respect to the number of edges.

Intuitively speaking, the set $\mathsf{CAND}(x)$ contains all "last common" edges of paths from node $x$ to two facilities, leading to an increase of interdiction budget if additional nodes should be cut off. Note that in Definition 4, we do not require $u \neq v$; thus, every edge incident to a facility is also contained in $\mathsf{CAND}(x)$. In Fig. 3, the candidates $\mathsf{CAND}(x)$ with respect to node $x \in V$ are drawn dashed. We claim that it suffices to consider edges from $\mathsf{CAND}(x)$ for interdiction. In fact, $\delta(V') \subseteq \mathsf{CAND}(x)$ is even a necessary condition for the optimality of $V'$, if one presumes no redundant edges.

**Lemma 4** *Let $V^* \subseteq V \setminus S$ be optimal for Problem 5. Then $\delta(V^*) \subseteq \mathsf{CAND}(x)$.*

***Proof*** Consider any optimal set $V^*$ with $|\delta(V^*)| = r'$ such that there exists an edge $e \in \delta(V^*)$ with $e \notin \mathsf{CAND}(x)$. As stated before, $e$ must be contained in a path

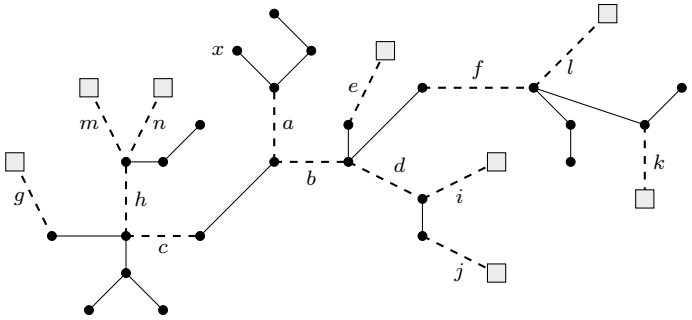


**Fig. 3** The candidate set $\mathsf{CAND}(x)$ is drawn dashed

$$P = ((x = v_0, v_1), (v_1, v_2), ..., (v_{p-1}, v_p = s))$$

connecting $x$ and a facility in $s \in S$ (otherwise the set of non-reachable nodes cannot be connected or $x$ is reachable). Let $e = (v_l, v_{l+1})$, i.e., $v_l \in V^*$ and $v_{l+1} \notin V^*$. Consider the edge set

$$\hat{E} := \{e \in E \mid e \text{ is incident to node } v_{l+1}\} \setminus \{e\},$$

and let $\Delta$ denote the number of edges in $\hat{E}$ that must be interdicted to keep the nodes in $V^* \cup \{v_{l+1}\}$ non-reachable. Suppose $\Delta \leq 1$. This is a contradiction to the optimality of $V^*$ as $|\delta(V^* \cup \{v_{l+1}\})| \leq |\delta(V^*)|$. Thus, it follows $\Delta \geq 2$. So, at least the edges $(v_{l+1}, v')$ and $(v_{l+1}, v'') \in \hat{E}$ with $v' \neq v''$ must be interdicted to keep $V^* \cup \{v_{l+1}\}$ non-reachable. This implies that both nodes $v'$ and $v''$ must lie on a path to different facilities. But then, by construction of $\mathsf{CAND}(x)$, the edge $(v_l, v_{l+1}) \in \mathsf{CAND}(x)$, which is a contradiction. □

In the sequel, we develop a solution algorithm for Problem 5 based on Lemma 4. It is easy to see that the (unique) minimum $x$-$S$-cut among the edges in $\mathsf{CAND}(x)$ corresponds to the interdiction strategy with the lowest cost that makes $x$ non-reachable and, at the same time, secondarily maximizes the number of non-reachable nodes. The idea is to iteratively increase the interdiction cost by dynamic programming shifting the set of interdicted edges step by step towards the facilities, enlarging the non-reachable area. For any edge of the current interdiction strategy, there is by Lemma 4 a distinct set of edges which may replace this edge. More precisely, one asks for the minimum cost extension of the non-reachable area in this direction. Formally, the *replacement set* of a candidate $e \in \mathsf{CAND}(x)$ is defined as follows:

$$\mathsf{rep}(e) := \{e' \in \mathsf{CAND}(x) \mid \text{the} \omega(e) - \alpha(e') - \text{path contains no edge of } \mathsf{CAND}(x)\},$$

where $\alpha(e)$ and $\omega(e)$ denote the endpoints of edge $e \in E$ closer to and farther away from node $x$, respectively. By construction of $\mathsf{CAND}(x)$, when the set of non-reachable nodes $V'$ should be extended by exchanging a current edge $e \in \delta(V')$, all edges

in $\mathsf{rep}(e)$ must be interdicted. Otherwise, $V'$ would be reachable through this path. Thus, replacing edge $e$ increases the current interdiction cost by

$$\mathsf{cost}(e) := |\mathsf{rep}(e)| - 1,$$

as one unit of cost can be saved for not anymore interdicting edge $e$. To simplify notation, we define the set of nodes enclosed by an edge set as follows:

**Definition 5** (*Enclosed Node Set*) For a proper interdiction strategy $R \subseteq E$ and a node $x \in V$, we define $\eta_x(R)$ to be the set of nodes containing $x$ *enclosed by* $R$, i.e., $\eta_x(R)$ is the node set of the connected component containing node $x$ in $T \setminus R$. If $R$ is not proper, we define $\eta_x(R) := \eta_x(R')$, where $R' \subseteq R$ denotes the strategy where all redundant edges of $R$ are removed.

With the last definition, we can describe the set of nodes additionally made non-reachable by the replacement of edge $e \subseteq R$ in any proper interdiction strategy:

$$\mu_x(e) := \eta_x(R \setminus \{e\} \cup \mathsf{rep}(e)) \setminus \eta_x(R)$$

Although the last definition suggests that $\mu_x(e)$ depends on the exact choice of $R$, the increment of replacing edge $e$ by $\mathsf{rep}(e)$ is the same for all such constructed interdiction strategies, as the set of additional non-reachable nodes is the same.

***Example 2*** Recall the network presented in Fig. 3. Replacing edge $c$ in a current interdiction strategy by edges $g$ and $h$ implies $|\mu_x(c)| = 5$, no matter which edges of $\{b, d, e, f, i, j, k, l\}$ are interdicted, see Fig. 4.

The profit of replacing edge $e$ in a current proper interdiction strategy $R$ by the set $\mathsf{rep}(e)$ is thus given by

$$\mathsf{profit}(e) := |\mu_x(e)|.$$

As mentioned, the algorithmic idea is to extend a current interdiction strategy iteratively by replacing one of its edges by its replacement set. Clearly, the current set of non-reachable nodes may be expanded into different directions, i.e., the edge of the current interdiction strategy that must be replaced by its replacement set is not (necessarily) unique. The goal is to find a feasible sequence of replacements such that

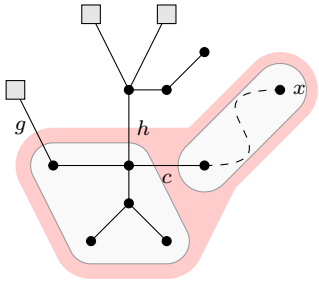


**Fig. 4** The increment of non-reachable nodes replacing edge $c$ by edges $g$ and $h$

the induced non-reachable area is maximal. To find this sequence of replacements, we change the perception of the replacements: the candidate edges of the original graph become nodes in a decision tree which is browsed by dynamic programming for finding the best replacement strategy. We point out that choosing a replacement does not mean that the corresponding edge is interdicted, but replaced by its replacement set in the current interdiction strategy.

Initially, we start with an artificial replacement denoted by $\emptyset^x$ and define $\mathsf{rep}(\emptyset^x)$ to be the set of edges contained in the minimum $x$-$S$-cut among the edges in $\mathsf{CAND}(x)$. Straightforward, we define $\mathsf{cost}(\emptyset^x) := |\mathsf{rep}(\emptyset^x)|$ and $\mathsf{profit}(\emptyset^x) = \eta_x(\mathsf{rep}(\emptyset^x))$.

The set of all possible replacements is thus given by $\mathcal{R} := \{e \in \mathsf{CAND}(x) \mid \mathsf{rep}(e) \neq \emptyset\} \cup \{\emptyset^x\}$. Choosing replacement $\emptyset^x$ means that an initially empty interdiction strategy is replaced by the minimum $x$-$S$-cut among the edges in $\mathsf{CAND}(x)$. For an element $\tau \in \mathcal{R} \setminus \{\emptyset^x\}$, we set its parent

$$\mathsf{parent}(\tau) := \tau', \text{ where } \tau' \text{ is the unique element such that } \tau \in \mathsf{rep}(\tau').$$

The provided parent–children relation between the replacements enables to illustrate the feasible replacement strategies in a decision tree with node set $\mathcal{R}$ rooted in node $\emptyset^x$ (see Fig. 5).

**Definition 6** A replacement strategy $I \subseteq \mathcal{R}$ is called *feasible*, if

1. $\emptyset^x \in I$ and
2. for $\tau \in I, \tau \neq \emptyset^x$, holds that $\mathsf{parent}(\tau) \in I$.

Consequently, every feasible replacement strategy is represented by a subtree of the decision tree that contains the root node $\emptyset^x$. Note that the replacement strategy does not explicitly specify the edges to be interdicted; rather, it tells which edges are being substituted, starting with $\emptyset^x$. Nevertheless, one can implicitly compute the resulting interdiction strategy: for a feasible replacement strategy $I \subseteq \mathcal{R}$, the corresponding interdiction strategy is given by $\{\tau \in \bigcup_{\tau' \in I} \mathsf{rep}(\tau') \mid \tau \notin I\}$. Costs and profit of $I$ are computed by $\mathsf{cost}(I) := \sum_{\tau \in I} \mathsf{cost}(\tau)$ and $\mathsf{profit}(I) := \sum_{\tau \in I} \mathsf{profit}(\tau)$, respectively.

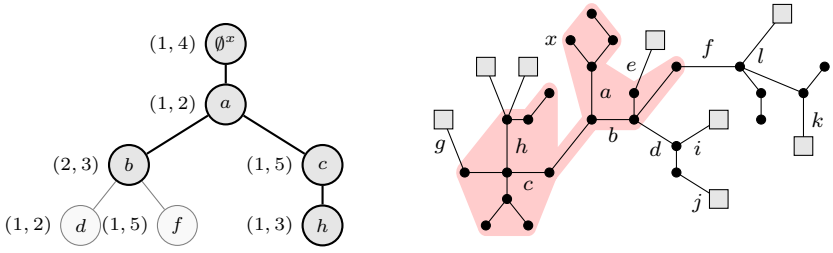


**Fig. 5** The decision tree with highlighted replacement strategy $\{\emptyset^x, a, b, c, h\}$ for the network in Fig. 3 with respect to node $x$ (left) and its corresponding non-reachable area (right)

Without loss of generality, we relabel the set of possible replacements $\mathcal{R} = \{\tau_0 = \emptyset^x, \tau_1, ..., \tau_p\}$ according to a breadth-first search starting from node $\emptyset^x$. That is, for two elements $\tau_i, \tau_j \in \mathcal{R}$ with $i < j$ it must hold $\text{dist}(\tau_i, \emptyset^x) \leq \text{dist}(\tau_j, \emptyset^x)$.

***Example 3*** The decision tree for the network given in Fig. 3 with respect to node $x$ is given in Fig. 5. The tuples next to the nodes depict costs and profits. The highlighted replacement strategy is $\{\emptyset^x, a, b, c, h\}$ with total cost 6 and total profit 17. The corresponding interdiction strategy is $\{d, e, f, g, m, n\}$. An ordering of $\mathcal{R}$ according to a breadth-first search is given by $\{\emptyset^x, a, b, c, d, f, h\}$.

The idea of the dynamic programming approach is to restrict the decision tree to the first $j + 1$ replacements $\{\tau_0, ..., \tau_j\}, j \leq |\mathcal{R}|$. By $W_{(i,j,k)}$, we denote the objective value of a corresponding optimal replacement strategy $I_{(i,j,k)}$ with respect to interdiction budget $i \leq r'$ provided that $\tau_k \in I_{(i,j,k)}$ and $I_{(i,j,k)} \subseteq \{\tau_0, ..., \tau_j\}$. Note that the last condition requires $k \leq j$. The third index $k \in \{1, ..., |\mathcal{R}|\}$ is needed to ensure feasibility of the replacement set, as outlined in the next lines.

Consider an arbitrary triple $(i, j, k)$ for suitable indices $i$, $j$ and $k$. As we require $\tau_k \in I_{(i,j,k)}$, there are three cases, and the best of these cases is chosen:

1. There is a former replacement strategy that already includes the required replacement $\tau_k$ for some lower interdiction budget $i' < i$ maximizing the total profit. In this case, we set
$$W_{(i,j,k)} = W_{(i-1,j,k)}.$$
Consequently, the related replacement strategy does not change, i.e., it follows $I_{(i,j,k)} = I_{(i-1,j,k)}$.
2. There is a former replacement strategy that already includes the required replacement $\tau_k$ for some smaller decision tree $j' < j$ maximizing the total profit. In this case, we set
$$W_{(i,j,k)} = W_{(i,j-1,k)}.$$
Again, the related replacement strategy does not change, i.e., it follows $I_{(i,j,k)} = I_{(i-1,j,k)}$.
3. There is a former replacement strategy not yet containing the replacement $\tau_k$ that can be extended by the replacement $\tau_k$ to maximize the total profit. For this more involved case, the computation of $W_{(i,j,k)}$ is explained below.

In the third case, we only may extend former replacement strategies $I_{(i',j',k')}$

- that contain only allowed replacements, i.e., it must hold $j' \leq j$,
- that satisfy the budget constraint after adding replacement $\tau_k$, i.e., it must hold $i' + \text{cost}(\tau_k) \leq i$, and
- whose corresponding interdiction strategy contains the edge $\tau_k$, (otherwise it cannot be replaced). This is the case if and only if $\text{parent}(\tau_k) \in I_{(i',j',k')}$ and $\tau_k \notin I_{(i',j',k')}$.

To shorten notation of the latter condition, we denote the candidate set of these feasible indices $k'$ by

$$\mathcal{K}(i,j,k) := \{k' \in \{1,...,j\} \mid \mathsf{parent}(\tau_k) \in I_{(i-\mathsf{cost}(\tau_k),j,k')} \text{ and } \tau_k \notin I_{(i-\mathsf{cost}(\tau_k),j,k')}\}$$

Intuitively speaking, the set $\mathcal{K}(i,j,k)$ contains all the indices $k'$ such that replacement $\tau_k$ can be added to $I_{(i-\mathsf{cost}(\tau_k),j,k')}$. This is illustrated in Fig. 6: the two left replacement strategies can be extended by replacement $c$, but not the right one, as this replacement is already made.

Thus, in the third case, the value $W_{(i,j,k)}$ can be computed by

$$W_{(i,j,k)} = \max_{k' \in \mathcal{K}(i,j,k)} W_{(i-\mathsf{cost}(\tau_k),j,k')} + \mathsf{profit}(\tau_k).$$

Combining the three cases, the recursion formula to compute $W_{(i,j,k)}$ is given by

$$W_{(i,j,k)} := \max\{W_{(i-1,j,k)}, \tag{3}$$

$$W_{(i,j-1,k)}, \tag{4}$$

$$\max_{k' \in \mathcal{K}(i,j,k)} W_{(i-\mathsf{cost}(\tau_k),j,k')} + \mathsf{profit}(\tau_k)\}, \tag{5}$$

provided that the requested index exists (if not assume the value to be $-\infty$). The recursion can be initialized with $I_{(\mathsf{cost}(\emptyset^x),j,0)} = \{\emptyset^x\}$ for all $j \in \{0,...,|\mathcal{R}|\}$. It follows:

**Theorem 2** *The optimal solution of Problem 5 is given by*

$$\max_{k \in \{1,...,|\mathcal{R}|\}} W_{(r',|\mathcal{R}|,k)},$$

*i.e., Algorithm 2 solves Problem 5 correctly in polynomial time.*

***Proof*** The correctness of the algorithm follows by the previous discussion. The candidate set $\mathsf{CAND}(x)$ can be determined in $\mathcal{O}(|V^2|)$, costs and profits can be obtained in linear time. In (5), the maximum over at most $j$ values is computed. As $r' \leq |E| \leq |V|$ and $|\mathcal{R}| \leq |E|$, the total runtime of Algorithm 2 is in $\mathcal{O}(|V|^4)$. □

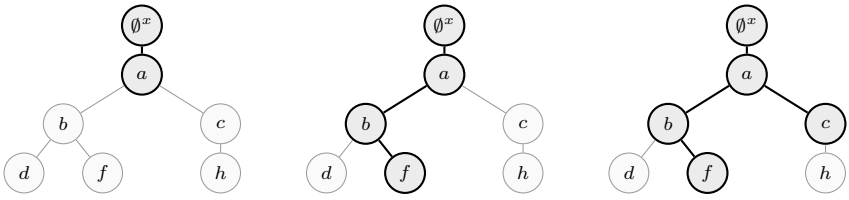


**Fig. 6** Three different replacement strategies. Only the left two can be extended by replacement $c$, as replacement $c$ is already contained in the right one

Note that Algorithm 2 is not optimized in terms of running time as our goal is to show polynomial solvability and to present an easy-to-understand reasoning.

**Algorithm 2:** Dynamic Program for Problem 5

**Input** : A cluster $T$ according to Definition 3, an integral interdiction budget $r' \leq r$ and a node $x \in V \setminus S$

**Output** : An optimal solution of Problem 5 and its objective value

$W_{(i,j,k)} \leftarrow -\infty$ for all $i \in \{-|S|, ..., 0, ..., r'\}, j \in \{0, ..., |\mathcal{R}|\}, k \in \{0, ..., j\}$

$W_{(\mathsf{cost}(\emptyset^x),j,0)} \leftarrow \mathsf{profit}(\emptyset)$ for all $j \in \{1, ..., |\mathcal{R}|\}$

**for** $i \in \{cost(\emptyset^x) + 1, ..., r'\}$ **do**
  **for** $j \in \{1, ..., |\mathcal{R}|\}$ **do**
    **for** $k \in \{1, ..., j\}$ **do**
      $W_{(i,j,k)} \leftarrow \max\{W_{(i-1,j,k)},$
        $W_{(i,j-1,k)},$
        $\max_{k' \in \mathcal{K}(i,j,k)} W_{(i-\mathsf{cost}(\tau_k),j,k')} + \mathsf{profit}(\tau_k)\}$
      **construct** $I_{(i,j,k)}, R_{(i,j,k)}$

$k^* \leftarrow \mathrm{argmax}_{k \in \{1,...,\mathcal{R}\}} W_{(r',|\mathcal{R}|,k)}$

**return** $I_{(r',|\mathcal{R}|,k^*)}, W_{(r',|\mathcal{R}|,k^*)}$

***Example 4*** Recall the network of Example 3 (Fig. 3) and its corresponding decision tree given in Fig. 5. The input of Algorithm 2 is annotated in Table 1, and the computation is outlined in Table 2 for interdiction budget 4. All missing values are $-\infty$. The optimal solution is given by $W_{(4,6,6)} = 14$, and the replacement strategy obtained by backtracking is $\{\tau_0, \tau_1, \tau_3, \tau_6\} = \{\emptyset^x, a, c, h\}$. The corresponding interdiction strategy is thus $\{b, g, m, n\}$.

As already mentioned, an optimal solution of Problem 4 may not be composed of optimal solutions of Problem 5. We justify this claim in the next example.

***Example 5*** Consider the graph depicted in Fig. 7 and interdiction budget 11. Then, the (unique) optimal solution of Problem 4 is given by $\mathcal{V}^* = \{\{v_1, v_2, v_3, v_4\}, \{v_6, v_7\}\}$ with $|\delta(\{v_1, v_2, v_3, v_4\})| = 10$ and $|\delta(\{v_6, v_7\})| = 1$. However, for interdiction budget 10, node set $\{v_1, v_2, v_3, v_4\}$ is not optimal (for Problem 5) for any fixed node $v_1, ..., v_4$, as it is always better to cut off the 5 upper facilities. For example, for fixed node $v_1$, the set $\{v_1, v_4, v_5, v_6, v_7\}$ with $|\delta(\{v_1, v_4, v_5, v_6, v_7\})| = 10$ is optimal for Problem 5.

It is easy to see that the size of the network (and thus the runtime of Algorithm 2) can be reduced by merging appropriate nodes, leading to a *weighted* formulation of Problem 1:

***Remark 1*** If it holds for two nodes $u, v \in V \setminus S$ that $\mathsf{CAND}(u) = \mathsf{CAND}(v)$, then, in an optimal solution, node $u$ is non-reachable if and only if node $v$ is non-reachable. Thus, nodes $u$ and $v$ may be merged into a single node $\{u, v\}$ with $\mathsf{profit}(\{u, v\}) = 2$.

**Table 1** Input of Algorithm 2

| | $j$ | Element | Replacement set | Cost | Profit |
|---|---|---|---|---|---|
| $\tau_0$ | 0 | $\emptyset^x$ | $a$ | 1 | 4 |
| $\tau_1$ | 1 | $a$ | $b, c$ | 1 | 2 |
| $\tau_2$ | 2 | $b$ | $d, f$ | 2 | 3 |
| $\tau_3$ | 3 | $c$ | $h$ | 1 | 5 |
| $\tau_4$ | 4 | $d$ | – | 1 | 2 |
| $\tau_5$ | 5 | $f$ | – | 1 | 5 |
| $\tau_6$ | 6 | $h$ | – | 1 | 3 |

**Table 2** Computation of the dynamic program

| $i = 1$ | $i = 2$ | $i = 3$ | $i = 4$ |
|---|---|---|---|
| $W_{(1,0,0)} = 4$ | $W_{(2,1,1)} = W_{(1,1,0)} + 2 = 6$ | $W_{(3,1,1)} = W_{(2,1,1)} = 6$ | $W_{(4,1,1)} = W_{(3,1,1)} = 6$ |
| $W_{(1,1,0)} = 4$ | $W_{(2,2,1)} = W_{(2,1,1)} = 6$ | $W_{(3,2,1)} = W_{(3,1,1)} = 6$ | $W_{(4,2,1)} = W_{(4,1,1)} = 6$ |
| $W_{(1,2,0)} = 4$ | $W_{(2,3,1)} = W_{(2,2,1)} = 6$ | $W_{(3,3,1)} = W_{(3,2,1)} = 6$ | $W_{(4,3,1)} = W_{(4,2,1)} = 6$ |
| $W_{(1,3,0)} = 4$ | $W_{(2,4,1)} = W_{(2,3,1)} = 6$ | $W_{(3,4,1)} = W_{(3,3,1)} = 6$ | $W_{(4,4,1)} = W_{(4,3,1)} = 6$ |
| $W_{(1,4,0)} = 4$ | $W_{(2,5,1)} = W_{(2,4,1)} = 6$ | $W_{(3,5,1)} = W_{(3,4,1)} = 6$ | $W_{(4,5,1)} = W_{(4,4,1)} = 6$ |
| $W_{(1,5,0)} = 4$ | $W_{(2,6,1)} = W_{(2,5,1)} = 6$ | $W_{(3,6,1)} = W_{(3,5,1)} = 6$ | $W_{(4,6,1)} = W_{(4,5,1)} = 6$ |
| $W_{(1,6,0)} = 4$ | | $W_{(3,3,3)} = W_{(2,3,1)} + 5 = 11$ | $W_{(4,2,2)} = W_{(2,2,1)} + 3 = 9$ |
| | | $W_{(3,4,3)} = W_{(3,3,3)} = 11$ | $W_{(4,3,2)} = W_{(4,2,2)} = 9$ |
| | | $W_{(3,5,3)} = W_{(3,4,3)} = 11$ | $W_{(4,4,2)} = W_{(4,3,2)} = 9$ |
| | | $W_{(3,6,3)} = W_{(3,5,3)} = 11$ | $W_{(4,5,2)} = W_{(4,4,2)} = 9$ |
| | | | $W_{(4,6,2)} = W_{(4,5,2)} = 9$ |
| | | | $W_{(4,3,3)} = W_{(3,3,3)} = 11$ |
| | | | $W_{(4,4,3)} = W_{(4,3,3)} = 11$ |
| | | | $W_{(4,5,3)} = W_{(4,4,3)} = 11$ |
| | | | $W_{(4,6,3)} = W_{(4,5,3)} = 11$ |
| | | | $W_{(4,6,6)} = W_{(3,6,3)} + 3 = 14$ |

Note that the equivalence of the minimum $u$-$S$-cut and the minimum $v$-$S$-cut is a (necessary and) sufficient condition for $\mathsf{CAND}(u) = \mathsf{CAND}(v)$.

### 3.2 Dynamic program for Problem 2

As a last step to show polynomial solvability of Problem 1, we show how to extend Algorithm 2 solving the restricted Problem 5 to solve the unrestricted Problem 2. By combining the solutions of all the clusters according to Algorithm 1, the desired result then follows.

Intuitively speaking, the idea of the overall solution algorithm is as follows: in contrast to Problem 5, where a single non-reachable area "grows" by iteratively increasing the budget, we now have several such disjoint areas. Thus, in addition to

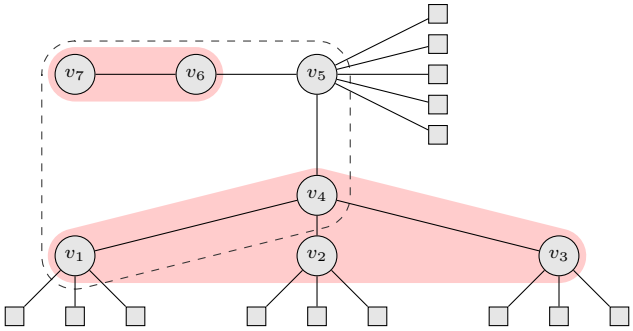


**Fig. 7** An instance with $r = 11$ and the optimal solution of Problem 4. However, the non-reachable area $\{v_1, v_2, v_3, v_4\}$ cannot be found by Problem 5

enlarging one of the current non-reachable areas with increasing interdiction budget, there is also the possibility to establish a new (disjoint) non-reachable area. It turns out that Algorithm 2 only needs to be slightly modified to cope with this.

To this end, we combine all the decision trees $DT_x$, $x \in V \setminus S$ as explained before in a single decision tree. Define the new decision tree $DT$ as follows: Create a root node $\emptyset$. For every $x \in V \setminus S$, denote by $DT_x$ the decision tree with respect to node $x$. All these decision trees $DT_x$ are linked by an edge from their root node $\emptyset^x$ to $\emptyset$. The resulting graph is sketched in Fig. 8. Again, we assume that the nodes of $DT$ are labeled according to a breadth-first search starting at $\emptyset$ by $\{\tau_0 = \emptyset, \tau_1, ..., \tau_p\}$. Further, we set $\mathsf{rep}(\emptyset) := \{\emptyset^x \mid x \in V \setminus S\}$ and $\mathsf{cost}(\emptyset) := \mathsf{profit}(\emptyset) := 0$.

If element $\emptyset^x$, $x \in V \setminus S$, is part of the current replacement strategy, there is a non-reachable area established with respect to node $x$ according to the previous section. Thus, every node of the cluster may be the origin of a non-reachable area, and every established non-reachable area may be extended analogously to Algorithm 2. However, we have to ensure that no two different non-reachable areas overlap, as this would cause non-proper interdiction strategies and re-count some non-reachable nodes when summing profits, requiring the following adaption of the recursion.

We keep the notation and interpretation of $W_{(i,j,k)}$, $I_{(i,j,k)}$, and $R_{(i,j,k)}$ analogous to Algorithm 2. Further, we introduce a set $V_{(i,j,k)} \subseteq V \setminus S$ containing all non-reachable nodes with respect to interdiction strategy $R_{(i,j,k)}$, which is easy to compute. Consider any arbitrary iteration $(i, j, k)$ and say that $\tau_k$ is part of the decision tree $DT_x$, i.e., $\tau_k$ is a replacement to enlarge the non-reachable area originating from node $x$. Then, the adapted recursion is given by

$$W_{(i,j,k)} := \max\{W_{(i-1,j,k)}, \tag{6}$$

$$W_{(i,j-1,k)}, \tag{7}$$

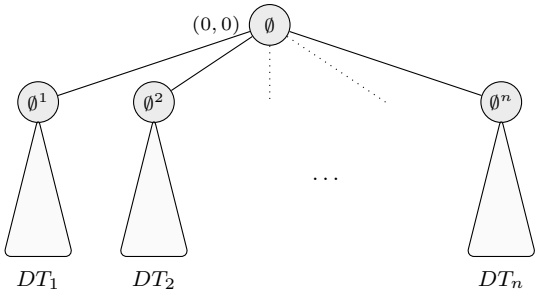


**Fig. 8** Construction of the decision tree

$$\max_{\substack{k' \in \mathcal{K}(i,j,k) : \\ V_{(i-\text{cost}(\tau_k),j,k')} \cap \mu_x(\tau_k) = \emptyset}} W_{(i-\text{cost}(\tau_k),j,k')} + \text{profit}(\tau_k)\}, \tag{8}$$

provided that the requested index exists (if not we again assume the value to be $-\infty$). The recursion can be initialized by $I_{(0,j,0)} = \{\emptyset\}$ for all $j \in \{0, ..., |\mathcal{R}|\}$.

In contrast to before, there is a slight difference in (8): additionally to the previous conditions, we have to ensure that no two different non-reachable areas merge, as the resulting interdiction strategy would not be proper and the computation of the profit would not be well-defined. Note that there is always a proper optimal interdiction strategy, such that this is no restriction. For all $k' \in \mathcal{K}(i,j,k)$, the condition $V_{(i-\text{cost}(\tau_k),j,k')} \cap \mu_x(\tau_k) = \emptyset$ can be easily checked, and the proof of the algorithm is analogous to before.

***Remark 2*** Obviously, a specific non-reachable area may be obtained by several replacement strategies with different origins. From that point on, both non-reachable areas could be augmented in the same way. Thus, the size of the decision tree can be reduced by deleting duplicate strategies that represent the same outcome, improving the runtime of the algorithm (cf. Example 6).

***Example 6*** Consider the upper left network given in Fig. 9. The encircled nodes can be merged due to Remark 1. The nodes are labeled by numbers, edges by letters. The resulting decision tree is sketched in Fig. 10. The grayed out parts can be omitted by Remark 2. The development of the current best non-reachable area(s) during the execution of the algorithm is illustrated in Fig. 9. Exemplarily, we show the computation of $W_{(4,9,8)}$, the optimum value with respect to interdiction budget 4 (note that we only demonstrate the path of the recursion where the maximum value is attained).

$$\begin{aligned}
W_{(4,9,8)} &= W_{(3,9,7)} + \text{profit}(\tau_8) = 8 + 3 = 11 \\
W_{(3,9,7)} &= W_{(2,9,1)} + \text{profit}(\tau_7) = 6 + 2 = 8 \\
W_{(2,9,1)} &= W_{(1,9,6)} + \text{profit}(\tau_1) = 3 + 3 = 6 \\
W_{(1,9,6)} &= W_{(0,9,0)} + \text{profit}(\tau_6) = 0 + 3 = 3
\end{aligned}$$

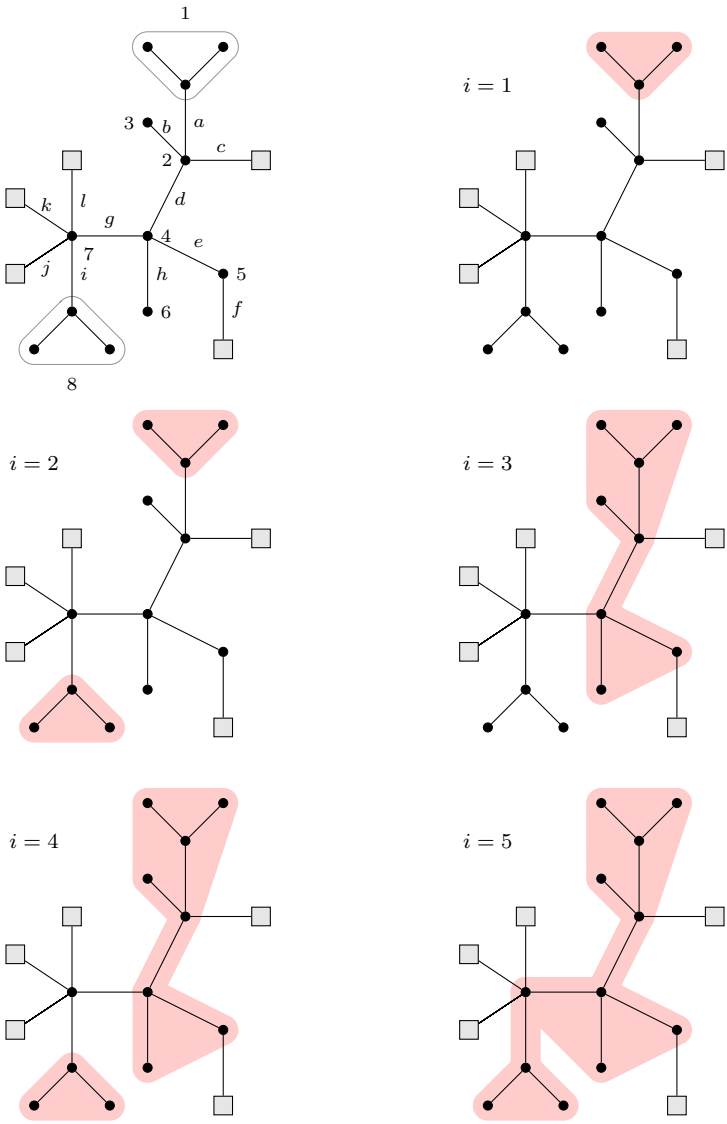


**Fig. 9** Optimal non-reachable areas for different interdiction budget

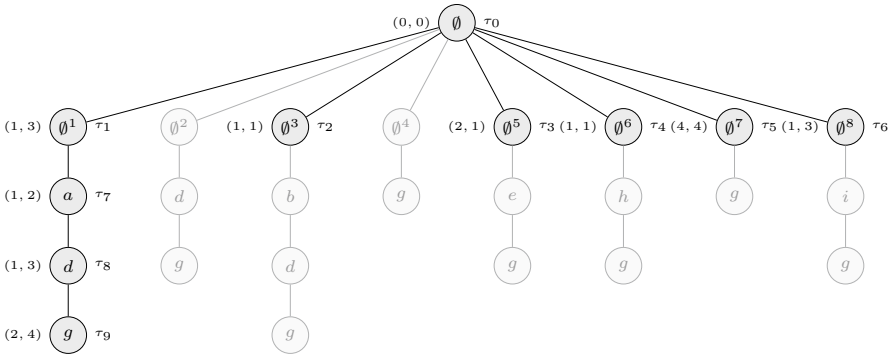


**Fig. 10** The decision tree for the network given in Fig. 9

In total we can conclude:

**Theorem 3** *Problem 2 is solvable in polynomial time.*

Note that the procedure discussed before overcomes the problem pointed out in Example 5, as every non-reachable area is extended in such a way that the total profit is optimized and not the profit of a single non-reachable area.

Finally, combining the solution algorithms of Problem 2 and Problem 3 as discussed at the beginning of this section, we can state the polynomial solvability of Problem 1:

**Theorem 4** *Problem 1 is solvable in polynomial time.*

## 4 Conclusion

In this article, we introduced an interdiction problem on tree networks, interdicting a subset of the edges in order to impair the facilities. Precisely, we asked for a size-constrained set of edges that maximizes the number of non-reachable nodes after interdiction. We formulated the problem as an integer program and presented a polynomial time algorithm to solve the problem. To the best of our knowledge, this is the first polynomial time algorithm for an interdiction problem in the context of facility location planning with edge interdiction. The developed algorithm is based on dynamic programming and reveals similarities to a knapsack problem. However, an additional dimension in the recurrence relation is needed, as some parts of the interdiction strategy are dependent from other parts.

Further research may concentrate on either optimizing running time to solve this problem or to study transferability to other, more general graphs. Further research may also address the locator's perspective: which nodes should be chosen as a facility such that the destruction of the network by removing a certain number of edges

affects the established facilities as less as possible. The latter problem is closely related to robust optimization assuming a worst-case analysis.

**Acknowledgements** This work was partially supported by the Bundesministerium für Bildung und Forschung (BMBF) under Grant FKZ 13N14561.

**Funding** Open Access funding enabled and organized by Projekt DEAL. This work was partially supported by the Bundesministerium für Bildung und Forschung (BMBF) under Grant FKZ 13N14561.

**Declarations**

**Conflict of interest** The authors declare that they have no conflict of interest.



## References

Adenso-Diaz B, Mar-Ortiz J, Lozano S (2018) Assessing supply chain robustness to links failure. Int J Prod Res 56(15):5104–5117

Ahmad W, Hasan O, Pervez U, Qadir J (2017) Reliability modeling and analysis of communication networks. J Network Comput Appl 78:191–215

Aksen D, Piyade N, Aras N (2010) The budget constrained r-interdiction median problem with capacity expansion. Central Eur J Oper Res 18(3):269–291

Alzorba S, Günther C, Popovici N (2015) A special class of extended multicriteria location problems. Optimization 64(5):1305–1320

Aringhieri R, Grosso A, Hosteins P, Scatamacchia R (2019) Polynomial and pseudo-polynomial time algorithms for different classes of the distance critical node problem. Discr Appl Math 253:103–121. https://doi.org/10.1016/j.dam.2017.12.035

Assimakopoulos N (1987) A network interdiction model for hospital infection control. Comput Biol Med 17(6):413–422

Baggio A, Carvalho M, Lodi A, Tramontani A (2021) Multilevel approaches for the critical node problem. Oper Res

Bar-Noy A, Khuller S, Schieber B (1995) The complexity of finding most vital arcs and nodes. Tech. Rep

Baron O, Milner J, Naseraldin H (2011) Facility location: a robust optimization approach. Prod Oper Manage 20(5):772–785

Bazgan C, Toubaline S, Vanderpooten D (2010) Complexity of determining the most vital elements for the 1-median and 1-center location problems. In: International conference on combinatorial optimization and applications. Springer, pp 237–251

Bazgan C, Toubaline S, Vanderpooten D (2013) Complexity of determining the most vital elements for the p-median and p-center location problems. J Comb Opt 25(2):191–207

Boros E, Borys K, Gurevich V, Rudolf G (2006) Inapproximability bounds for shortest-path network interdiction problems. Technical report, Rutgers University, Piscataway, NJ, USA

Burch C, Carr R, Krumke S, Marathe M, Phillips C, Sundberg E (2003) A decomposition-based pseudoapproximation algorithm for network flow inhibition. In: Network interdiction and stochastic integer programming. Springer, pp 51–68

Carrizosa E, Nickel S (2003) Robust facility location. Math Methods Oper Res 58(2):331–349

Chestnut SR, Zenklusen R (2016) Interdicting structured combinatorial optimization problems with {0, 1} -objectives. Math Oper Res 42(1):144–166
Chestnut SR, Zenklusen R (2017) Hardness and approximation for network flow interdiction. Networks 69(4):378–387
Church RL, Scaparra MP, Middleton RS (2004) Identifying critical infrastructure: the median and covering facility interdiction problems. Ann Assoc Am Geogr 94(3):491–502
Dinitz M, Gupta A (2013) Packing interdiction and partial covering problems. In: Goemans M, Correa J (eds) Integer programming and combinatorial optimization. Springer, Berlin, Heidelberg. pp 157–168
Drexl M, Schneider M (2015) A survey of variants and extensions of the location-routing problem. Eur J Oper Res 241(2):283–308
Fröhlich N, Ruzika S (2020) The complexity of median-location problems with edge interdiction. Tech. rep. Technische Universität Kaiserslautern
Fröhlich N, Ruzika S (2021) On the hardness of covering-interdiction problems. Theor. Comput. Sci. https://doi.org/10.1016/j.tcs.2021.04.007
Furini F, Ljubić I, Martin S, Segundo PS (2019) The maximum clique interdiction problem. Eur J Oper Res 277(1):112–127. https://doi.org/10.1016/j.ejor.2019.02.028
Ghaffarinasab N, Atayi R (2018) An implicit enumeration algorithm for the hub interdiction median problem with fortification. Eur J Oper Res 267(1):23–39. https://doi.org/10.1016/j.ejor.2017.11.035
Gilmore PC, Gomory RE (1966) The theory and computation of knapsack functions. Oper Res 14(6):1045–1074
Israeli E, Wood KR (2002) Shortest-path network interdiction. Networks Int J 40(2):97–111
Kalcsics J, Nickel S, Pozo MA, Puerto J, Rodríguez-Chía AM (2014) The multicriteria p-facility median location problem on networks. Eur J Oper Res 235(3):484–493
Khachiyan L, Boros E, Borys K, Elbassioni K, Gurvich V, Rudolf G, Zhao J (2008) On short paths interdiction problems: total and node-wise limited interdiction. Theory Comput Syst 43(2):204–233
Laporte G, Nickel S, Saldanha da Gama F (2015) Location science, vol 528. Springer
Lozano L, Smith JC (2017) A backward sampling framework for interdiction problems with fortification. INFORMS J Comput 29(1):123–139
Mahmoodjanloo M, Parvasi SP, Ramezanian R (2016) A tri-level covering fortification model for facility protection against disturbance in r-interdiction median problem. Comput Ind Eng 102:219–232
Morton DP, Pan F, Saeger KJ (2007) Models for nuclear smuggling interdiction. IIE Trans 39(1):3–14
Pan F, Schild A (2016) Interdiction problems on planar graphs. Discr Appl Math 198:215–231
Pferschy U, Schauer J (2009) The knapsack problem with conflict graphs. J Graph Algorithms Appl 13(2):233–249
Phillips CA (1993) The network inhibition problem. In: Proceedings of the twenty-fifth annual ACM symposium on theory of computing, pp 776–785
Ramamoorthy P, Jayaswal S, Sinha A, Vidyarthi N (2018) Multiple allocation hub interdiction and protection problems: model formulations and solution approaches. Eur J Oper Res 270(1):230–245. https://doi.org/10.1016/j.ejor.2018.03.031
Scaparra MP, Church RL (2008) An exact solution approach for the interdiction median problem with fortification. Eur J Oper Res 189(1):76–92. https://doi.org/10.1016/j.ejor.2007.05.027
Shen S, Smith JC (2011) Polynomial-time algorithms for solving a class of critical node problems on trees and series-parallel graphs. Networks 60(2):103–119. https://doi.org/10.1002/net.20464
Smith JC, Song Y (2020) A survey of network interdiction models and algorithms. Eur J Oper Res 283(3):797–811. https://doi.org/10.1016/j.ejor.2019.06.024
Soleimani-Alyar M, Ghaffari-Hadigheh A, Sadeghi F (2016) Controlling floods by optimization methods. Water Resour Manage 30(12):4053–4062
Streib L, Kattwinkel M, Heer H, Ruzika S, Schäfer RB (2020) How does habitat connectivity influence the colonization success of a hemimetabolous aquatic insect?—a modeling approach. Ecol Model 416:108909
Ullmert T, Ruzika S, Schöbel A (2020) On the p-hub interdiction problem. Comput Oper Res 124:105056
Wood RK (1993) Deterministic network interdiction. Math Comput Model 17(2):1–18
Zenklusen R (2010) Network flow interdiction on planar graphs. Discr Appl Math 158(13):1441–1455
Zenklusen R (2014) Connectivity interdiction. Oper Res Lett 42(6–7):450–454