Journal of the Operations Research Society of China
https://doi.org/10.1007/s40305-023-00488-y

Approximation Algorithms for Solving the *k*-Chinese Postman Problem Under Interdiction Budget Constraints

Peng-Xiang Pan[1] · Jun-Ran Lichen[2] · Wen-Cheng Wang[1] · Li-Jian Cai[1] · Jian-Ping Li[1]

Received: 29 November 2022 / Revised: 29 March 2023 / Accepted: 17 April 2023

Abstract

In this paper, we address the k-Chinese postman problem under interdiction budget constraints (the k-CPIBC problem, for short), which is a further generalization of the k-Chinese postman problem and has many practical applications in real life. Specifically, given a weighted graph $G = (V, E; w, c; v_1)$ equipped with a weight function $w : E \rightarrow \mathbb{R}^+$ that satisfies the triangle inequality, an interdiction cost function $c : E \rightarrow \mathbb{Z}^+$, a fixed depot $v_1 \in V$, an integer $k \in \mathbb{Z}^+$ and a budget $B \in \mathbb{N}$, we are asked to find a subset $S_k \subseteq E$ such that $c(S_k) = \sum_{e \in S_k} c(e) \leqslant B$ and that the subgraph $G \backslash S_k$ is connected, the objective is to minimize the value $\min_{\mathcal{C}_{E \backslash S_k}} \max\{w(C_i) \mid C_i \in \mathcal{C}_{E \backslash S_k}\}$ among such all aforementioned subsets S_k, where

These authors are supported by the National Natural Science Foundation of China (Nos. 11861075, 12101593) and Project for Innovation Team (Cultivation) of Yunnan Province (No. 202005AE-160006). Peng-Xiang Pan is also supported by Project of Yunnan Provincial Department of Education Science Research Fund (No. 2020Y0040), Jun-Ran Lichen is also supported by Fundamental Research Funds for the Central Universities (No. buctrc202219), and Jian-Ping Li is also supported by Project of Yunling Scholars Training of Yunnan Province (No. K264202011820).

✉ Jian-Ping Li
jianping@ynu.edu.cn

Peng-Xiang Pan
pengxiang@ynu.edu.cn

Jun-Ran Lichen
J.R.Lichen@buct.edu.cn

Wen-Cheng Wang
wencheng2018a@163.com

Li-Jian Cai
1915618969@qq.com

1 School of Mathematics and Statistics, Yunnan University, Kunming 650504, Yunnan, China

2 School of Mathematics and Physics, Beijing University of Chemical Technology, Beijing 100190, China

$\mathcal{C}_{E\setminus S_k}$ is a set of k-tours (of $G\setminus S_k$) starting and ending at the depot v_1, jointly traversing each edge in $G\setminus S_k$ at least once, and $w(C_i) = \sum_{e\in C_i} w(e)$ for each tour $C_i \in \mathcal{C}_{E\setminus S_k}$. We obtain the following main results: (1) Given an α-approximation algorithm to solve the minimization knapsack problem, we design an $(\alpha+\beta)$-approximation algorithm to solve the k-CPIBC problem, where $\beta = \frac{7}{2} - \frac{1}{k} - \lfloor \frac{1}{k} \rfloor$. (2) We present a β-approximation algorithm to solve the special version of the k-CPIBC problem, where $c(e) \equiv 1$ for each edge e in G and β is defined in (1).

Keywords Combinatorial optimization · Arc routing · k-Chinese postman problem · Interdiction · Approximation algorithms

Mathematics Subject Classification 90C27 · 68W25

1 Introduction

The problems to be discussed in this paper are all defined on undirected graphs. Arc routing problems are very active and interesting topics in operations research fields, due to their many applications in our reality, such as routings of street sweepers, mail delivery, snow plows and the inspection of electric power lines. In most arc routing problems, the objective is to determine a least-weight traversal of a set of specified edges in a graph. Guan [1] in 1960 considered the Chinese postman problem (the CP problem, for short), which is one of the arc routing problems specialized to weighted graphs, and it is defined as follows. Given a weighted graph $G = (V, E; w)$ with a weight function $w : E \rightarrow \mathbb{R}^+$, it is asked to find a tour C traversing each edge $e \in E$ at least once, the objective is to minimize the total weight $w(C) = \sum_{e\in E} t(e)w(e)$ among all possible tours mentioned above, where $t(e)$ denotes the number of times that the edge e in G is traversed by the tour C. For the CP problem, using an algorithm to solve the maximum weighted matching problem [2], Edmonds and Johnson [3] designed an exact polynomial-time algorithm.

Frederickson et al. [4] in 1978 introduced the k-Chinese postman problem (the k-CP problem, for short), which is a generalization of the CP problem, and it is defined as follows. Given a weighted graph $G = (V, E; w; v_1)$ with a weight function $w : E \rightarrow \mathbb{R}^+$, a fixed depot $v_1 \in V$ and an integer $k \in \mathbb{Z}^+$, it is asked to find a set $\mathcal{C}_k = \{C_1, C_2, \cdots, C_k\}$ of k tours starting and ending at the same depot v_1, jointly traversing each edge in G at least once. The objective is to minimize the weight $w_{\max}(\mathcal{C}_k) = \max\{w(C_i) \mid C_i \in \mathcal{C}_k\}$ among all k tours, where $w(C_i) = \sum_{e\in C_i} c(e)$ for each $i = 1, 2, \cdots, k$. Although the CP problem is solvable by the exact algorithm due to Edmonds and Johnson [3], Frederickson et al. [4] showed that the k-CP problem is NP-complete for the case $k \geqslant 2$, and using a splitting technique, they designed a $(2 - \frac{1}{k})$-approximation algorithm to solve the k-CP problem. For other different arc routings and related problems, we can find considerable number of approximation algorithms in these references [5–11].

In network design, it is often of interest to know how sensitive a particular property of a network is with respect to changes in the graph structure, like the removal or failure of edges or vertices. One natural measure of sensitivity is to compute the maximum change of the property when some limited set of edges or vertices is being removed, which leads to a class of problems referred to as the interdiction problems. The interdiction problems have been employed in a wide variety of settings, these problems involve many practical applications in real life, such as military planning [12], hospital infection control [13], drug interdiction [14], infrastructure protection [15] and electric power grids protection [16].

Many edge interdiction problems may be formulated as follows. Given a weighted graph $G = (V, E; w, c)$ with a weight function $w : E \to \mathbb{R}^+$, an interdiction cost function $c : E \to \mathbb{Z}^+$, and a budget $B \in \mathbb{N}$, it is asked to determine a subset $E' \subseteq E$ satisfying $c(E') = \sum_{e \in E'} c(e) \leqslant B$. The objective is to maximize the change of some existing property (i.e., the optimal value of some optimization problem) between the original graph G and the residual subgraph $G \backslash E'$, where $G \backslash E'$ is the subgraph of G only by deleting the edges in E' from G. Similarly, the interdiction problems of vertices in undirected graphs may be formulated in the view to use a certain set of vertices to substitute for a subset of edges in the edge interdiction problems. Given many different existing properties, these interdiction problems have been studied extensively, including the minimum spanning tree interdiction problem [17–21], the maximum matching interdiction problem [22–24], the shortest path interdiction problem [25–27], the maximum s-t flow interdiction problem [14, 28, 29] and the connectivity interdiction (minimum cut interdiction) problem [30]. It is well known that most of these problems are *NP*-complete [18, 22, 26, 29, 30], even in some special versions.

In recent years, many approximation algorithms have been designed to solve some aforementioned interdiction problems. Using linear programming and rounding techniques, Dinitz and Gupta [23] in 2013 presented an $O(1)$-approximation algorithm to solve the maximum matching interdiction problem. Employing an algorithm for solving the budgeted minimum cut problem [31], Zenklusen [30] in 2014 gave a polynomial-time approximation scheme (PTAS) to solve the connectivity interdiction problem. Using a greedy algorithm for extracting a good interdiction set from one that exceeds the interdiction budget, Zenklusen [20] in 2015 designed a 14-approximation algorithm to solve the minimum spanning tree interdiction problem. Considering the tree knapsack problem [32] to extract a good interdiction set, Linhares and Swamy [21] in 2017 designed a 4-approximation algorithm to resolve the minimum spanning tree interdiction problem. In particular, Smith and Song [33] in 2020 presented a survey of network interdiction models and algorithms. However, as far as we know, there are no approximation algorithms with constant performance ratios to solve the interdiction versions of the arc routing problems.

In real life, it is necessary to rescue persons on some roads of a network who are about to suffer from natural disasters. Now, we may consider the question as follows. Given k vehicles that are used to rescue persons on some roads of a network, it is necessary to arrange these k vehicles starting and ending at the same depot such that they jointly traverse each road of this network at least once. In order to minimize the rescuing-time span as short as possible, we would choose a limited set of roads from this network, whose removal causes to decrease maximum of rescuing-time span, equivalently, these k vehicles do not need to jointly traverse these limited roads removed. And then, we determine k routings of these k vehicles to jointly traverse each road in this residual network, transfer persons in need to the fixed depot, avoid traversing these limited roads removed. The question must be raised in the following way: which limited roads of this network are chosen for removal?

Motivated by this interesting question, the aforementioned interdiction problems and the k-CP problem, we want to find a limited set of edges for removal so that the optimal value of the k-CP problem on the remaining graph is as small as possible, then we address the k-Chinese postman problem under interdiction budget constraints (the k-CPIBC problem, for short). Specifically, given a weighted connected graph $G = (V, E; w, c; v_1)$ equipped with a weight function $w : E \rightarrow \mathbb{R}^+$ that satisfies the triangle inequality, an interdiction cost function $c : E \rightarrow \mathbb{Z}^+$, a fixed depot $v_1 \in V$, an integer $k \in \mathbb{Z}^+$ and a budget $B \in \mathbb{N}$, we are asked to find a subset $S_k \subseteq E$ such that $c(S_k) = \sum_{e \in S_k} c(e) \leqslant B$ and that the subgraph $G \backslash S_k$ is connected, where $G \backslash S_k$ is the subgraph of G only by deleting all edges in S_k from G. The objective is to minimize the value $\min_{\mathcal{C}_{E \backslash S_k}} \max\{w(C_i) \mid C_i \in \mathcal{C}_{E \backslash S_k}\}$ among all aforementioned subsets S_k, i.e., $\min_{S_k} \min_{\mathcal{C}_{E \backslash S_k}} \max\{w(C_i) \mid C_i \in \mathcal{C}_{E \backslash S_k}\}$, where $\mathcal{C}_{E \backslash S_k}$ is a set of k-tours starting and ending at the same depot v_1, jointly traversing each edge in $G \backslash S_k$ at least once, and $w(C_i) = \sum_{e \in C_i} c(e)$ for each tour $C_i \in \mathcal{C}_{E \backslash S_k}$. For convenience, such a set of these k-tours, denoted by $\mathcal{C}_{E \backslash S_k}$, is called as a k-tours covering of the subgraph $G \backslash S_k$.

Now, there are three comments to be presented: (1) Whenever the budget B is less than the value $\min\{c(e) \mid e \in E\}$, the k-CPIBC problem reduces to the k-CP problem, implying that the k-CPIBC problem is *NP*-complete for the case $k \geqslant 2$. (2) We can prove that the 1-CPIBC problem is *NP*-complete (see Theorem 1). (3) Whenever a cost function $c(\cdot)$ satisfies $c(e) \equiv 1$ for each edge e in the connected graph G, we refer to this version of the k-CPIBC problem as the k-Chinese postman problem under interdiction cardinality constraints (the k-CPICC problem, for short).

As far as we know, the k-CPIBC problem and the related problems have not been considered in the literature works. We shall design some approximation algorithms to solve this new problem and its special version in the sequel. We hope that these problems will have many further practical applications in real life.

The remainder of this paper is organized into the following sections. In Sect. 2, we present some terminologies and notations to state descriptions of approximation algorithms and some fundamental lemmas to ensure the correctness of our algorithms. In Sect. 3, given an α-approximation algorithm to solve the minimization knapsack problem, we design an $(\alpha + \beta)$-approximation algorithm to solve the k-CPIBC problem, where $\lfloor \frac{1}{k} \rfloor$ is the greatest integer less than or equal to $\frac{1}{k}$ and $\beta = \frac{7}{2} - \frac{1}{k} - \lfloor \frac{1}{k} \rfloor$. In Sect. 4, we present a β-approximation algorithm to solve the k-CPICC problem, where β is defined as mentioned above. In Sect. 5, we provide our conclusion and further research.

2 Preliminaries

In order to clearly present our approximation algorithms to solve the k-CPIBC problem, we provide some terminologies, notations and fundamental lemmas in the sequel, most of which are standard.

If no confusion, given a graph $G = (V, E)$, we may denote $n = |V|$ and $m = |E|$ to be the numbers of vertices and edges in G, respectively. For a real number r, denote $\lfloor r \rfloor$ to be the greatest integer less than or equal to r.

Given a graph $G = (V, E)$, a graph $G' = (V', E')$ is called as a subgraph of G if $V' \subseteq V$ and $E' \subseteq E$. In addition, if $V' = V$, then $G' = (V', E')$ is called a spanning subgraph of G. If $E' = \{uv \in E \mid u, v \in V'\}$, the subgraph $G[V'] = (V', E')$ is called as the subgraph induced by V'. Similarly, if $V' = \{u, v \in V \mid uv \in E'\}$, we call $G[E'] = (V', E')$ as the subgraph induced by E'. For a subset $E' \subseteq E$ in G, denote $G \backslash E' = (V, E \setminus E')$.

Given two vertices $v_{i_1}, v_{i_{k+1}}$ in the graph $G = (V, E)$, a walk P connecting v_{i_1} and $v_{i_{k+1}}$ is an alternating sequence $\pi = v_{i_1} e_{i_1} v_{i_2} e_{i_2} v_{i_3} \cdots v_{i_k} e_{i_k} v_{i_{k+1}}$ such that, for each integer $1 \leqslant j \leqslant k$, two end-vertices of edge e_{i_j} are v_{i_j} and $v_{i_{j+1}}$. A walk P is called as a tour with k edges if $v_{i_1} = v_{i_{k+1}}$. A walk P is said to be a path if the vertices in P are all distinct. Similarly, we call a tour C a cycle if the vertices in C are all distinct. In addition, a Hamiltonian cycle C is a cycle that traverses each vertex $v \in V$ exactly once, and an Euler tour C is a tour that traverses each edge $e \in E$ exactly once. We call a graph G to be an Eulerian graph if G admits an Euler tour.

A graph $G = (V, E)$ is called to be connected if, for each pair of distinct vertices $x, y \in V$, there exists a path P_{xy} connecting x and y. For a vertex $v \in V$, the degree $d_G(v)$ of vertex v is the number of edges which are incident to v. We call v an even-degree vertex if $d_G(v)$ is even, otherwise v is said to be an odd-degree vertex. In addition, a matching M is a set of vertex-disjoint edges, and M is called a perfect matching if the vertices in V are all covered by M. Moreover, for a weighted graph $G = (V, E; w)$, M is called a minimum-weight perfect matching if M is a perfect matching and $w(M) = \sum_{e \in M} w(e)$ is minimized among all perfect matchings in G.

Given a weighted graph $G = (V, E; w; v_1)$ with a fixed depot $v_1 \in V$ and an integer $k \in \mathbb{Z}^+$, a k-tours covering in G is a set of k tours starting and ending at the same depot v_1, jointly traversing each edge in G at least once. An optimal k-tours covering in G is a k-tours covering such that the maximum value of the weights of such k tours is minimized among all k-tours coverings in G. Given a set $S \subseteq E$, we then denote $\mathcal{C}_S^k$ as an optimal k-tours covering in the induced subgraph $G[S]$ and $w_{\max}(\mathcal{C}_S^k) = \max\{w(C_i) \mid C_i \in \mathcal{C}_S^k\}$ as the weight of such a k-tours covering $\mathcal{C}_S^k$ for $G[S]$. In particular, we have $w(\mathcal{C}_S^1) = w_{\max}(\mathcal{C}_S^1)$.

For convenience, we denote an instance of the k-CPIBC problem as a given weighted graph $G = (V, E; w, c; v_1)$ with an integer $k \in \mathbb{Z}^+$ and a budget $B \in \mathbb{N}$, where $w : E \to \mathbb{R}^+$ is a weight function that satisfies the triangle inequality, $c : E \to \mathbb{Z}^+$ is an interdiction cost function and $v_1 \in V$ is a fixed depot. If there exists a subset $S_k \subseteq E$ such that $c(S_k) \leqslant B$ and that the subgraph $G \setminus S_k$ is connected, implying that there exists a k-tours covering in $G \setminus S_k$, then we call the subset $S_k \subseteq E$ to be a feasible solution to the graph G for the k-CPIBC problem.

When we design approximation algorithms to solve the k-CPIBC problem, we shall use the following lemmas.

Lemma 1 *Given a weighted connected graph $G = (V, E; w, c; v_1)$ as an instance of the k-CPIBC problem and the m-CPIBC problem, respectively, if $k \geqslant m$, then the k-CPIBC problem is equivalent to the m-CPIBC problem, where $m = |E|$.*

Proof Given an instance $G = (V, E; w, c; v_1)$ of the k-CPIBC problem and the m-CPIBC problem, respectively, we may first prove that S' is a feasible solution to G for the k-CPIBC problem with the value $w_{\max}(\mathcal{C}^k_{E\backslash S'})$ if and only if S' is a feasible solution to G for the m-CPIBC problem with the value $w_{\max}(\mathcal{C}^m_{E\backslash S'})$. In fact, if S' is a feasible solution to G for the k-CPIBC problem, we have $c(S') \leqslant B$ and that the subgraph $G\backslash S'$ is connected, implying that there exists an m-tours covering in $G \setminus S'$ by the fact that there is an optimal tour to $G \setminus S'$ for the Chinese postman problem [3], then S' is a feasible solution to the graph G for the m-CPIBC problem. And vice versa.

Secondly, we may prove that $w_{\max}(\mathcal{C}^k_{E\backslash S'}) = w_{\max}(\mathcal{C}^m_{E\backslash S'})$. For each integer $k' \geqslant |E \setminus S'|$, since $\mathcal{C}^{k'}_{E\backslash S'}$ is an optimal k'-tours covering in the subgraph $G \setminus S'$, it is easy to conclude that $w_{\max}(\mathcal{C}^{k'}_{E\backslash S'}) = \max\{w(P'_{v_1,v_i}) + w(v_i v_j) + w(P'_{v_j,v_1}) \mid v_i v_j \in E\backslash S'\}$, where $P'_{x,y}$ is a shortest path from x to y in $G \setminus S'$. By the assumption $k \geqslant m = |E| \geqslant |E\backslash S'|$, we obtain $w_{\max}(\mathcal{C}^k_{E\backslash S'}) = w_{\max}(\mathcal{C}^m_{E\backslash S'}) = \max\{w(P'_{v_1,v_i}) + w(v_i v_j) + w(P'_{v_j,v_1}) \mid v_i v_j \in E\backslash S'\}$. Thus, the k-CPIBC problem is equivalent to the m-CPIBC problem.

This completes a proof of the lemma.

For convenience, using Lemma 1, we may assume that $k \leqslant m$ in each instance of the k-CPIBC problem.

For a weighted connected graph $G = (V, E; w)$ with two distinct vertices $v_i, v_j \in V$, using the Dijkstra algorithm [34] to solve the shortest path problem, we can construct a shortest v_i-v_j path in G.

Lemma 2 [34] *Given a weighted graph $G = (V, E; w)$ with two vertices $v_i, v_j \in V$, the Dijkstra algorithm can either produce a shortest v_i-v_j path in G or show no such a path, and this algorithm runs in time $O(n^2)$.*

Given a weighted connected graph $G = (V, E; w)$ and a subset $F_1 \subseteq E$, modifying the MST algorithm in [34, 35] to solve the minimum spanning tree problem, we can determine a minimum-weight subset $F_2 \subseteq E$ such that the induced subgraph $G[F_1 \cup F_2]$ is a connected spanning subgraph of G.

Lemma 3 [34, 35] *Given a weighted connected graph $G = (V, E; w)$ and a subset $F_1 \subseteq E$, the MST algorithm can determine a minimum-weight subset $F_2 \subseteq E$ such that $G[F_1 \cup F_2]$ is a connected spanning subgraph of G, and this algorithm runs in time $O(n^2)$.*

Given a weighted connected graph $G = (V, E; w)$, we can use the Floyd algorithm [36] to construct all shortest paths among all vertices in G.

Lemma 4 [36] *The Floyd algorithm can produce all shortest paths among all vertices in G, and this algorithm runs in time $O(n^3)$.*

Given a weighted connected graph $G = (V, E; w)$, using a simple shrinking technique, Edmonds [2] in 1965 designed an exact algorithm in time $O(n^3)$ to produce a minimum-weight perfect matching in G. In addition, using a complicated data structure for blossom creation, Gabow [37] in 2018 presented an exact algorithm in lower time $O(n(m + n \log n))$ to determine a minimum-weight perfect matching in G. It is sufficient for us to use the Edmonds algorithm [2] as a subroutine to solve the k-CPIBC problem.

Lemma 5 [2] *The Edmonds algorithm can produce a minimum-weight perfect matching in G, and this algorithm runs in time $O(n^3)$.*

Given an Eulerian graph $G = (V, E)$, we can use the Euler algorithm [3] to determine an Euler tour in G.

Lemma 6 [3] *Given an Eulerian graph $G = (V, E)$, the Euler algorithm can produce an Euler tour in G, and this algorithm runs in time $O(m)$.*

We need the definition of K-tree in a connected graph $G = (V, E)$ [38]. For a nonnegative integer K, a spanning K-tree in G is a connected spanning subgraph $T_K = (V, E_K)$ with $|E_K| = n + K - 1$ edges in G. In fact, one 0-tree is exactly a spanning tree of G. Given a weighted connected graph $G = (V, E; w)$ with a weight function $w : E \to \mathbb{R}^+$ and an integer $B \in \mathbb{N}$, using the Fisher algorithm [38] to construct a minimum-weight spanning K-tree with $K = \max\{m - B - (n - 1), 0\}$, we can determine a minimum-weight connected spanning subgraph $T_K = (V, E_K)$ with $|E_K| \geqslant m - B$.

Lemma 7 [38] *Given a weighted connected graph $G = (V, E; w)$ and an integer $B \in \mathbb{N}$, the Fisher algorithm can produce a minimum-weight connected spanning subgraph $T_K = (V, E_K)$ with $|E_K| \geqslant m - B$ in G, and this algorithm runs in time $O(n^2)$.*

3 The *k*-Chinese Postman Problem Under Interdiction Budget Constraints

In this section, we consider the k-CPIBC problem. By using the definition of the k-CPIBC problem, whenever the budget B is less than the value $\min\{c(e) \mid e \in E\}$, the k-CPIBC problem reduces to the k-CP problem, implying that the k-CPIBC problem is *NP*-complete for the case $k \geqslant 2$.

However, unlike the 1-CP problem (i.e., Chinese postman problem), which is solvable in polynomial time [3], we have the following result for the 1-CPIBC problem.

Theorem 1 *The 1-CPIBC problem remains NP-complete, even if an interdiction cost function $c : E \to \mathbb{Z}^+$ satisfies $c(e) \equiv 1$ for each edge e in a weighted connected graph $G = (V, E; w, \mathbf{1}; v_1)$, where v_1 is a fixed vertex in G.*

Proof The reduction is transformed from the Hamiltonian cycle (HC) problem, where the HC problem is to determine whether a given connected graph $G = (V, E)$ would contain a cycle traversing each vertex in G exactly once. However, the HC problem is *NP*-complete [39].

Without loss of generality, we may assume that $m \geqslant n$, otherwise G is a tree by our assumption, implying that G is a "NO"-instance. Given an instance $G = (V, E)$ of the HC problem, we construct an instance $H = (V, E; w, c; v_1)$ of the 1-CPIBC problem by adding a weight function $w : E \rightarrow \{1\}$, an interdiction cost function $c : E \rightarrow \{1\}$ and a budget $B = m - n$, where $v_1 \in V$ is a fixed vertex. It is straightforward to obtain the fact that G is a "YES"-instance if and only if the optimal value OPT to the instance H for the 1-CPIBC problem is exactly n, i.e., OPT$= n$. Thus, the 1-CPIBC problem is *NP*-complete.

This completes a proof of the theorem.

In order to efficiently design an approximation algorithm to solve the k-CPIBC problem, we intend to consider the minimization knapsack problem (the MinKP problem, for short) [40, 41], which is defined as follows. Given a set $X = \{x_1, x_2, \cdots, x_q\}$ of q items with a weight function $u : X \rightarrow \mathbb{R}^+$, a cost function $s : X \rightarrow \mathbb{R}^+$, and a constant D, it is asked to find a subset $X' \subseteq X$ to have $u(X') = \sum_{x \in X'} u(x) \geqslant D$, the objective is to minimize $s(X') = \sum_{x \in X'} s(x)$. For convenience, we always use $I = (X; u, s)$ to denote an instance of the MinKP problem.

Given a weighted connected graph $G = (V, E; w, c)$ with a weight function $w : E \rightarrow \mathbb{R}^+$, an interdiction cost function $c : E \rightarrow \mathbb{Z}^+$, and an integer $B \in \mathbb{N}$, we construct an instance $I = (X; u, s)$ of the MinKP problem in the following ways. Denote $X = \{x_e \mid e \in E\}$ and a constant $D = c(E) - B$. For each item $x_e \in X$, denote $u(x_e) = c(e)$ and $s(x_e) = w(e)$, respectively. Using an α-approximation algorithm $\mathcal{A}_\alpha$ in [40–43] to solve the MinKP problem, we can determine a subset $X' \subseteq X$ with $u(X') \geqslant c(E) - B$ and $s(X') \leqslant \alpha \cdot s(X^*)$, where X^* is an optimal solution to the instance I for the MinKP problem.

Given a subset X' of the aforementioned instance $I = (X; u, s)$ for the MinKP problem, we can construct an edge-subset $F_1 = \{e \mid x_e \in X'\}$ of the weighted connected graph $G = (V, E; w, c)$ to have $c(F_1) \geqslant c(E) - B$ and $w(F_1) \leqslant \alpha \cdot w(F_1^*)$, where F_1^* is a minimum-weight edge-subset of G such that $c(F_1^*) \geqslant c(E) - B$.

Lemma 8 *Let $G = (V, E; w, c)$ be a connected graph with a weight function $w : E \rightarrow \mathbb{R}^+$, a cost function $c : E \rightarrow \mathbb{Z}^+$ and an integer $B \in \mathbb{N}$. Given an α-approximation algorithm $\mathcal{A}_\alpha$ to solve the MinKP problem, we can determine a subset $F_1 \subseteq E$ with $c(F_1) \geqslant c(E) - B$ and $w(F_1) \leqslant \alpha \cdot w(F_1^*)$, where F_1^* is a minimum-weight edge-subset such that $c(F_1^*) \geqslant c(E) - B$.*

Proof For a weighted graph $G = (V, E; w, c)$, we construct an instance $I = (X; u, s)$ for the MinKP problem as mentioned above. Suppose that X^* is an optimal solution to the instance I for the MinKP problem and that F_1^* is a minimum-weight edge-subset of G such that $c(F_1^*) \geqslant c(E) - B$. Using the algorithm $\mathcal{A}_\alpha$, we can determine a subset $X' \subseteq X$ with $u(X') \geqslant c(E) - B$ and $s(X') \leqslant \alpha \cdot s(X^*)$. Denoting $F_1 = \{e \mid x_e \in X'\}$, we obtain $c(F_1) = u(X') \geqslant c(E) - B$.

Since F_1^* is a minimum-weight edge-subset such that $c(F_1^*) \geqslant c(E) - B$, we obtain the fact that $X_1^* = \{x_e \mid e \in F_1^*\}$ is an optimal solution to the instance I for the MinKP problem. Otherwise, we may suppose, to the contrary, that we obtain $s(X_1^*) > s(X^*)$. Then, there exists an edge-subset $F^* = \{e \mid x_e \in X^*\}$ to have the properties $c(F^*) = u(X^*) \geqslant c(E) - B$; however, $w(F^*) = s(X^*) < s(X_1^*) = w(F_1^*)$, which contradicts that F_1^* is a minimum-weight edge-subset such that $c(F_1^*) \geqslant c(E) - B$. Indeed, X_1^* is an optimal solution to the instance I for the MinKP problem, implying $s(X_1^*) = s(X^*)$.

Since $s(X') \leqslant \alpha \cdot s(X^*)$ by the algorithm $\mathcal{A}_\alpha$ to solve the MinKP problem, we obtain the following:

$$w(F_1) = s(X') \leqslant \alpha \cdot s(X^*) = \alpha \cdot s(X_1^*) = \alpha \cdot w(F_1^*).$$

This completes a proof of the lemma.

We now consider the tour augmentation (TA, for short) problem, which is defined as follows. Given a weighted connected graph $G = (V, E; w; v_1)$ with a depot $v_1 \in V$, a weight function $w : E \to \mathbb{R}^+$, an integer $k \in \mathbb{Z}^+$ and a tour $C = v_1 v_{i_1} v_{i_2} \cdots v_{i_t} v_1$ that traverses each vertex v in G at least once, it is asked to find a subset $F_4 \subseteq E$ such that the weight of the optimal k-tours covering in the induced subgraph $G[E(C) \cup F_4]$ is minimized, i.e., $\min_{F_4} w_{\max}(\mathcal{C}^k_{E(C)\cup F_4})$.

Using a splitting technique in [4] to solve the k-CP problem, we design an algorithm, denoted by the TA algorithm, to solve the TA problem, which is described as follows.

Algorithm 1 : TA

Input: A connected graph $G = (V, E; w; v_1)$ with a depot $v_1 \in V$, a function $w : E \to \mathbb{R}^+$, an integer $k \in \mathbb{Z}^+$, and a tour $C = v_1 v_{i_1} v_{i_2} \cdots v_{i_t} v_1$ that traverses each vertex $v \in V$ at least once;
Output: A subset $F_4 \subseteq E$ such that $w_{\max}(\mathcal{C}^k_{E(C)\cup F_4})$ is as small as possible.

```
Begin
Step 1 For each vertex $u \in V$, use the Dijkstra algorithm [34] to find a shortest path $P_{v_1,u}$ connecting $v_1$
  and $u$ in $G$, depending on the function $w(\cdot)$. Denote $\mathcal{P} = \{P_{v_1,u} \mid u \in V\}$, $w_0 = \max\{w(P_{v_1,u}) \mid u \in V\}$
  and $F_4 = \varnothing$;
Step 2 For $j = 1$ to $k - 1$ do:
    (2.1) Denote $L_j = \frac{j}{k}(w(C) - 2w_0) + w_0$;
    (2.2) Find the last vertex $v_{i_{q(j)}}$ on $C$ such that $w\big(C[v_1, v_{i_{q(j)}}]\big) \leqslant L_j$,
              where $C[v_1, v_{i_{q(j)}}] := v_1 v_{i_1} v_{i_2} \cdots v_{i_{q(j)}}$;
    (2.3) Denote $R_j = L_j - w\big(C[v_1, v_{i_{q(j)}}]\big)$;
    (2.4) If $R_j + w\big(P_{v_{i_{q(j)}},v_1}\big) \leqslant w\big(v_{i_{q(j)}} v_{i_{q(j)+1}}\big) - R_j + w\big(P_{v_{i_{q(j)+1}},v_1}\big)$ then
              denote $v_{i_{p(j)}} = v_{i_{q(j)}}$;
          Else
              denote $v_{i_{p(j)}} = v_{i_{q(j)+1}}$;
    (2.5) Set $F_4 := F_4 \cup E(P_{v_1,v_{i_{p(j)}}})$;
Step 3 Output "the subset $F_4$."
End
```

Executing the TA algorithm, we obtain the following.

Lemma 9 *Given an instance of the tour augmentation problem, the TA algorithm can produce an edge-subset F_4 such that there exists a k-tours covering $\{C_j \mid j = 1, 2, \cdots, k\}$ in $G[E(C) \cup F_4]$ satisfying $w(C_j) \leqslant \frac{1}{k} \cdot (w(C) - 2w_0) + 4w_0$ for each integer $j \in \{1, 2, \cdots, k\}$, where w_0 is defined at Step 1 in the TA algorithm, and this algorithm runs in time $O(n^3)$.*

Proof When $k = 1$, using the TA algorithm, we have $F_4 = \varnothing$ and that C is still a 1-tour covering in $G[E(C) \cup F_4]$, then we obtain $w(C) \leqslant w(C) + 2w_0 = \frac{1}{1} \cdot (w(C) - 2w_0) + 4w_0$.

When $k \geqslant 2$, using the TA algorithm, we construct a k-tours covering $\{C_j \mid j = 1, 2, \cdots, k\}$ in the subgraph $G[E(C) \cup F_4]$ induced by $E(C) \cup F_4$ as follows.

(1) We construct k walks in the following ways: $Q_1 = C[v_1, v_{i_{p(1)}}] = v_1 v_{i_1} \cdots v_{i_{p(1)}}$, $Q_2 = C[v_{i_{p(1)}}, v_{i_{p(2)}}] = v_{i_{p(1)}} \cdots v_{i_{p(2)}}, \cdots, Q_k = C[v_{i_{p(k-1)}}, v_1] = v_{i_{p(k-1)}} \cdots v_{i_t} v_1$;
(2) Denote $C_1 = Q_1 \circ P_{v_{i_{p(1)}}, v_1}$, $C_k = P_{v_1, v_{i_{p(k-1)}}} \circ Q_k$, and for each integer $j \in \{2, \cdots, k-1\}$, we can augment Q_j, adding two shortest paths $P_{v_1, v_{i_{p(j-1)}}}$ and $P_{v_1, v_{i_{p(j)}}}$ from $\mathcal{P} = \{P_{v_1,u} \mid u \in V\}$ to connect v_1 to the initial vertex $v_{i_{p(j-1)}}$ and the terminal vertex $v_{i_{p(j)}}$ of the walk Q_j, to become a tour C_j, i.e., $C_j = P_{v_1, v_{i_{p(j-1)}}} \circ Q_j \circ P_{v_{i_{p(j)}}, v_1}$. It is easy to verify that $\{C_j \mid j = 1, 2, \cdots, k\}$ is a k-tours covering in $G[E(C) \cup F_4]$.

Now, we first consider the tour C_j for each integer $j \in \{2, \cdots, k-1\}$. Since the weight function $w(\cdot)$ on set of edges in G satisfies the triangle inequality, depending on the definition of w_0, which is maximum weight of paths in $\mathcal{P}$ defined at Step 1 of the TA algorithm, we obtain the following:

$$w(v_{i_{q(j)}} v_{i_{q(j)+1}}) \leqslant w\big(P_{v_{i_{q(j)}}, v_1}\big) + w\big(P_{v_1, v_{i_{q(j)+1}}}\big) \leqslant 2w_0,$$

implying

$$w(P_{v_1, v_{i_{q(j)}}}) + w(v_{i_{q(j)}} v_{i_{q(j)+1}}) + w(P_{v_{i_{q(j)+1}}, v_1}) \leqslant 4w_0.$$

Then, we obtain the following:

$$\min\Big\{w(P_{v_1, v_{i_{q(j)}}}) + R_j,\ w(v_{i_{q(j)}} v_{i_{q(j)+1}}) - R_j + w(P_{v_{i_{q(j)+1}}, v_1})\Big\} \leqslant 2w_0.$$

Similarly, we obtain the following:

$$\min\Big\{w(P_{v_1, v_{i_{q(j-1)}}}) + R_{j-1},\ w(v_{i_{q(j-1)}} v_{i_{q(j-1)+1}}) - R_{j-1} + w(P_{v_{i_{q(j-1)+1}}, v_1})\Big\} \leqslant 2w_0.$$

Using the TA algorithm, we obtain the worst case where the weight of C_j is maximized only when this tour C_j starts at the depot v_1, reaching the vertex $v_{i_{q(j-1)}}$ along the shortest path $P_{v_1, v_{i_{q(j-1)}}}$, going to $v_{i_{q(j)+1}}$ along C, and finally ends at the depot v_1 along the shortest path $P_{v_{i_{q(j)+1}}, v_1}$.

In the worst case, it is easy to see that $w(P_{v_1, v_{i_{q(j-1)}}}) + R_{j-1} \leqslant 2w_0$ and $w(v_{i_{q(j)}} v_{i_{q(j)+1}}) - R_j + w(P_{v_{i_{q(j)+1}}, v_1}) \leqslant 2w_0$, then we obtain the following:

$$\begin{aligned} w(C_j) &\leqslant 2w_0 + \frac{1}{k} \cdot (w(C) - 2w_0) + 2w_0 \\ &= \frac{1}{k} \cdot (w(C) - 2w_0) + 4w_0. \end{aligned}$$

Similarly, we consider C_j for each $j \in \{1, k\}$, and we obtain the following:

$$\begin{aligned} w(C_j) &\leqslant w_0 + \frac{1}{k} \cdot (w(C) - 2w_0) + 2w_0 \\ &\leqslant \frac{1}{k} \cdot (w(C) - 2w_0) + 4w_0. \end{aligned}$$

Combining the aforementioned arguments, for each $j \in \{1, 2, \cdots, k\}$, we obtain the following:

$$w(C_j) \leqslant \frac{1}{k} \cdot (w(C) - 2w_0) + 4w_0.$$

The running time of the TA algorithm can be determined as follows. (1) By Lemma 2, since it needs time $O(n^2)$ to execute the Dijkstra algorithm [34] per circulation, Step 1 constructs the set $\mathcal{P}$ and the maximum weight w_0 in time $O(n^3)$. (2) Since $k \leqslant m$ by Lemma 1, Steps 2–3 execute in time $O(mn)$. Hence, the TA algorithm needs whole time $O(n^3)$.

This establishes the lemma.

Using the TA algorithm, we design an approximation algorithm to solve the k-CPIBC problem using the following strategies:

(1) Find a subset $F_1 \subseteq E$, having $c(F_1) \geqslant c(E) - B$, such that $w(F_1)$ is minimized;
(2) Find a minimum-weight subset $F_2 \subseteq E$, such that the induced subgraph $G[F_1 \cup F_2]$ is a connected spanning subgraph in G;
(3) Determine a minimum-weight subset $F_3 \subseteq E$ such that the induced subgraph $G[F_1 \cup F_2 \cup F_3]$ is an Eulerian graph;
(4) Find a subset $F_4 \subseteq E$ such that the weight of the optimal k-tours covering in the induced subgraph $G[F_1 \cup F_2 \cup F_3 \cup F_4]$ is minimized;
(5) Output the subset $S_k = E \backslash (F_1 \cup F_2 \cup F_3 \cup F_4)$.

Our approximation algorithm, denoted by the k-CPIBC algorithm, to solve the k-CPIBC problem is described as follows.

Algorithm 2 : k-CPIBC

```
Input: A connected graph $G = (V, E; w, c; v_1)$ with a depot $v_1 \in V$, a function $w : E \to \mathbb{R}^+$, an integer $k \in \mathbb{Z}^+$ and a budget $B \in \mathbb{N}$;
Output: A subset $S_k \subset E$ such that $\min_{\mathcal{C}_{E\setminus S_k}} \max\{w(C_i) \mid C_i \in \mathcal{C}_{E\setminus S_k}\}$ is as small as possible.

Begin
Step 1 Given a weighted connected graph $G = (V, E; w, c)$ as an instance of the $k$-CPIBC problem, we construct an instance $I = (X; u, s)$ of the MinKP problem, where $X = \{x_e \mid e \in E\}$, a constant $D = c(E) - B$, $u(x_e) = c(e)$ and $s(x_e) = w(e)$ for each item $x_e \in X$;
Step 2 Using an $\alpha$-approximation algorithm $\mathcal{A}_\alpha$ on this instance $I = (X; u, s)$ of the MinKP problem, determine a subset $X' \subseteq X$ with $u(X') \geqslant c(E) - B$ and $s(X') \leqslant \alpha \cdot s(X^*)$, where $X^*$ is an optimal solution to the instance $I$ for the MinKP problem;
Step 3 For the subset $X' \subseteq X$ produced at Step 2, construct a subset $F_1 = \{e \mid x_e \in X'\}$ $(\subseteq E)$, having $c(F_1) = u(X') \geqslant c(E) - B$ and $w(F_1) = s(X') \leqslant \alpha \cdot s(X^*)$;
Step 4 For the subset $F_1$, use the MST algorithm [34, 35] to find a minimum-weight subset $F_2 \subseteq E$ such that $G[F_1 \cup F_2]$ is a connected spanning subgraph in $G$; and denote $F_3 = \varnothing$;
Step 5 If (there exists an odd-degree vertex in $G[F_1 \cup F_2]$) then
    (5.1) Denote by $V_o$ the set of all odd-degree vertices in $G[F_1 \cup F_2]$. Construct a complete subgraph $H = (V_o, E_H; w_H)$ on $V_o$, where weight $w_H(v_i v_j)$ of edge $v_i v_j \in E_H$ is the weight of a shortest path $P_{v_i,v_j}$ connecting $v_i$ and $v_j$ in $G$;
    (5.2) Using the Edmonds algorithm [2], find a minimum-weight perfect matching $M$ in $H$; Denote $F_3 = \bigcup_{v_i v_j \in M} E(P_{v_i,v_j})$, where $E(P_{v_i,v_j})$ denotes the set of all edges locating on the shortest path $P_{v_i,v_j}$;
Step 6 If $(k = 1)$ then
        Output the subset $S_1 = E \setminus (F_1 \cup F_2 \cup F_3)$, and STOP.
Step 7 Using the Euler algorithm [3], determine an Euler tour $C$ in $G[F_1 \cup F_2 \cup F_3]$ that starts and ends at the depot $v_1$, where we may assume $C = v_1 v_{i_1} v_{i_2} \cdots v_{i_t} v_1$;
Step 8 Using the TA algorithm, determine a subset $F_4 \subseteq E$ such that the weight of the optimal $k$-tours covering in $G[E(C) \cup F_4]$ is as small as possible;
Step 9 Output the subset $S_k = E \setminus (F_1 \cup F_2 \cup F_3 \cup F_4)$.
End
```

Theorem 2 *Given an α-approximation algorithm $\mathcal{A}_\alpha$ for solving the MinKP problem, the k-CPIBC algorithm is an $(\alpha + \beta)$-approximation algorithm to solve the k-CPIBC problem, and this algorithm runs in time $O(n^3 + f(n, m))$, where $f(n, m)$ is the running time of the algorithm $\mathcal{A}_\alpha$ and $\beta = \frac{7}{2} - \frac{1}{k} - \lfloor \frac{1}{k} \rfloor$.*

Proof Given an instance $G = (V, E; w, c; v_1)$ of the k-CPIBC problem, we may assume that S_k^* is an optimal solution with an optimal k-tours covering $\mathcal{C}^k_{E\setminus S_k^*}$ in $G \setminus S_k^*$ and the optimal value $\text{OPT}_k = w_{\max}(\mathcal{C}^k_{E\setminus S_k^*})$. And let S_k $(= E\setminus(F_1 \cup F_2 \cup F_3 \cup F_4))$ denote the edge-subset produced by the k-CPIBC algorithm with a k-tours covering $\mathcal{C}^k_{E\setminus S_k}$ in the subgraph $G\setminus S_k$ and the output value $\text{OUT}_k = w_{\max}(\mathcal{C}^k_{E\setminus S_k})$.

For each integer $k \in \mathbb{Z}^+$, by Steps 1–4 of the k-CPIBC algorithm, we obtain the facts $c(F_1) \geqslant c(E) - B$ and that $G[F_1 \cup F_2]$ is a connected spanning subgraph in G, then we have $c(S_k) = c(E\setminus(F_1 \cup F_2 \cup F_3 \cup F_4)) \leqslant c(E\setminus F_1) \leqslant c(E) - (c(E) - B) = B$ and $G\setminus S_k \supseteq G[F_1 \cup F_2]$, which implies that $G \setminus S_k$ is connected. This shows that S_k produced by the k-CPIBC algorithm is a feasible solution to the instance $G = (V, E; w, c; v_1)$ of the k-CPIBC problem. We shall prove that $\text{OUT}_1 \leqslant (\alpha + \frac{3}{2}) \cdot \text{OPT}_1$ for the case $k = 1$ and $\text{OUT}_k \leqslant (\alpha + \frac{7}{2} - \frac{1}{k}) \cdot \text{OPT}_k$ for the case $k \geqslant 2$.

We consider the following two cases.

Case 1 $k = 1$

For each optimal solution S_1^* to the graph G as an instance of the 1-CPIBC problem, we have the facts $c(E\backslash S_1^*) \geqslant c(E) - B$ and that $G\backslash S_1^*$ is connected. Suppose that F_1^* is a minimum-weight edge-subset with $c(F_1^*) \geqslant c(E) - B$ in G. Since $E\backslash S_1^*$ is an edge-subset with $c(E\backslash S_1^*) \geqslant c(E) - B$ in G, i.e., $c(S_1^*) \leqslant B$, we obtain $w(F_1^*) \leqslant w(E\backslash S_1^*)$. Since the k-CPIBC algorithm executes the algorithm $\mathcal{A}_\alpha$ for the MinKP problem to find a subset with $F_1 \subseteq E$, using Lemma 8, we obtain $c(F_1) \geqslant c(E) - B$ and $w(F_1) \leqslant \alpha \cdot w(F_1^*)$, respectively. Then, we obtain the following:

$$w(F_1) \leqslant \alpha \cdot w(F_1^*) \leqslant \alpha \cdot w(E \setminus S_1^*) \leqslant \alpha \cdot w(\mathcal{C}^1_{E\backslash S_1^*}).$$

For the subgraph $G[F_1]$, the MST algorithm [34, 35] at Step 4 produces a minimum-weight edge-subset F_2 such that $G[F_1 \cup F_2]$ is a connected spanning subgraph in G. Since $G \setminus S_1^*$ is connected, we obtain the fact $w(F_2) \leqslant w(E\backslash S_1^*) \leqslant w(\mathcal{C}^1_{E\backslash S_1^*})$.

Using the similar arguments as in [44], we can use the tour $\mathcal{C}^1_{E\backslash S_1^*}$ to construct a Hamiltonian cycle $C_o = v_{j_1} v_{j_2} \cdots v_{j_{|V_o|}} v_{j_1}$ in $H = (V_o, E_H;\, w_H)$, i.e., the Hamiltonian cycle C_o may be constructed by visiting each vertex in the orders of their occurrences first in the tour $\mathcal{C}^1_{E\backslash S_1^*}$, satisfying $w_H(C_o) \leqslant w(\mathcal{C}^1_{E\backslash S_1^*})$. Knowing the fact that the Hamiltonian cycle C_o has even vertices, we can choose alternating edges from the even cycle C_o to construct two perfect matchings in H, denoted by $M_1 = \{v_{j_i} v_{j_{i+1}} \mid i = 1, 3, \cdots, |V_o| - 1\}$ and $M_2 = \{v_{j_i} v_{j_{i+1}} \mid i = 2, 4, \cdots, |V_o| - 2\} \cup \{v_{j_{|V_o|}} v_{j_1}\}$. Since Step 5 produces a minimum-weight perfect matching M in H, we obtain the fact $w_H(M) \leqslant \min\{w_H(M_1), w_H(M_2)\} \leqslant \frac{1}{2} w_H(C_o)$, implying that $w(F_3) = w_H(M) \leqslant \frac{1}{2} w_H(C_o) \leqslant \frac{1}{2} w(\mathcal{C}^1_{E\backslash S_1^*})$.

By Steps 3–5, we see that $G[F_1 \cup F_2 \cup F_3]$ is an Eulerian graph. Since $S_1 = E\backslash(F_1 \cup F_2 \cup F_3)$, it follows that $\mathrm{OUT}_1 = w(\mathcal{C}^1_{E\backslash S_1}) \leqslant w(F_1) + w(F_2) + w(F_3)$. Thus, we obtain the following:

$$\begin{aligned}
\mathrm{OUT}_1 &\leqslant w(F_1) + w(F_2) + w(F_3) \\
&\leqslant \alpha \cdot w(\mathcal{C}^1_{E\backslash S_1^*}) + w(\mathcal{C}^1_{E\backslash S_1^*}) + \frac{1}{2} w(\mathcal{C}^1_{E\backslash S_1^*}) \\
&= (\alpha + \frac{3}{2}) \cdot w(\mathcal{C}^1_{E\backslash S_1^*}) \\
&= (\alpha + \frac{3}{2}) \cdot \mathrm{OPT}_1.
\end{aligned}$$

Case 2 $k \geqslant 2$

Using the similar arguments in Case 1, we have $F_1 \cup F_2 \cup F_3 = E(C)$ and $w(C) = w(F_1) + w(F_2) + w(F_3) \leqslant (\alpha + \frac{3}{2}) \cdot w(\mathcal{C}^1_{E\backslash S_1^*})$. We shall prove $\mathrm{OUT}_k \leqslant (\alpha + \frac{7}{2} - \frac{1}{k}) \cdot \mathrm{OPT}_k$ in the sequel.

Since $F_1 \cup F_2 \cup F_3 = E(C)$ by the k-CPIBC algorithm, we obtain $G[F_1 \cup F_2 \cup F_3 \cup F_4] = G[E(C) \cup F_4]$. Using the proof of Lemma 9, we can construct a k-tours covering $\{C_j \mid j = 1, 2, \cdots, k\}$ in $G[F_1 \cup F_2 \cup F_3 \cup F_4]$, i.e., this k-tours covering is in $G \setminus S_k$, and for each integer $j \in \{1, 2, \cdots, k\}$, this k-tours covering satisfies $w(C_j) \leqslant \frac{1}{k} \cdot (w(C) - 2w_0) + 4w_0$.

Since S_1^* is an optimal solution to the graph G as an instance of the 1-CPIBC problem and S_k^* is an optimal solution to the graph G as an instance for the k-CPIBC problem, implying that S_k^* is a feasible solution to the graph G as an instance for the 1-CPIBC problem, we obtain $w(\mathcal{C}^1_{E\setminus S_1^*}) \leqslant w(\mathcal{C}^1_{E\setminus S_k^*})$. Since all tours in the k-tours covering $\mathcal{C}^k_{E\setminus S_k^*}$ start and end at the depot v_1, we can easily transform them into 1-tour covering in $G\setminus S_k^*$. Since $\mathcal{C}^1_{E\setminus S_k^*}$ is an optimal 1-tour covering in $G \setminus S_k^*$, we obtain $w(\mathcal{C}^1_{E\setminus S_k^*}) \leqslant w(\mathcal{C}^k_{E\setminus S_k^*}) \leqslant k \cdot w_{\max}(\mathcal{C}^k_{E\setminus S_k^*})$, which shows that $w(\mathcal{C}^1_{E\setminus S_1^*}) \leqslant w(\mathcal{C}^1_{E\setminus S_k^*}) \leqslant k \cdot w_{\max}(\mathcal{C}^k_{E\setminus S_k^*})$.

Since S_k^* is an optimal solution to the graph G as an instance of the k-CPIBC problem, we have the fact that $G\setminus S_k^*$ is a connected spanning subgraph in G, which implies that the union of all tours in $\mathcal{C}^k_{E\setminus S_k^*}$ must traverse each vertex $v \in V$ at least once, then we have $w_0 \leqslant \frac{1}{2} w_{\max}(\mathcal{C}^k_{E\setminus S_k^*})$. Thus, for each $j \in \{1, 2, \cdots, k\}$, we obtain the following:

$$\begin{aligned}
w(C_j) &\leqslant \frac{1}{k} \cdot (w(C) - 2w_0) + 4w_0 \\
&= 2 \cdot \left(2 - \frac{1}{k}\right) \cdot w_0 + \frac{1}{k} \cdot w(C) \\
&\leqslant 2 \cdot \left(2 - \frac{1}{k}\right) \cdot w_0 + \frac{1}{k} \cdot \left(\alpha + \frac{3}{2}\right) \cdot w(\mathcal{C}^1_{E\setminus S_1^*}) \\
&\leqslant \left(2 - \frac{1}{k}\right) \cdot w_{\max}(\mathcal{C}^k_{E\setminus S_k^*}) + \left(\alpha + \frac{3}{2}\right) \cdot w_{\max}(\mathcal{C}^k_{E\setminus S_k^*}) \\
&= \left(\alpha + \frac{7}{2} - \frac{1}{k}\right) \cdot w_{\max}(\mathcal{C}^k_{E\setminus S_k^*}) \\
&= \left(\alpha + \frac{7}{2} - \frac{1}{k}\right) \cdot \mathrm{OPT}_k,
\end{aligned}$$

implying $\mathrm{OUT}_k = w_{\max}(\mathcal{C}^k_{E\setminus S_k}) \leqslant \max\{w(C_j) \mid j = 1, 2, \cdots, k\} \leqslant (\alpha + \frac{7}{2} - \frac{1}{k}) \cdot \mathrm{OPT}_k$.

Combining the aforementioned arguments in Cases 1 and 2, we obtain the following:

$$\mathrm{OUT}_k \leqslant (\alpha + \beta) \cdot \mathrm{OPT}_k,$$

where $\beta = \frac{7}{2} - \frac{1}{k} - \lfloor \frac{1}{k} \rfloor$ for each integer $k \in \mathbb{Z}^+$. This shows that the k-CPIBC algorithm is an $(\alpha + \beta)$-approximation algorithm to solve the k-CPIBC problem.

The complexity of the k-CPIBC algorithm can be determined as follows: (1) Step 1 needs time $O(m + n)$ to construct an instance $I = (X; u, s)$ of the MinKP problem as mentioned above. (2) Step 2 executes the algorithm $\mathcal{A}_\alpha$ in time $f(n, m)$ to find a subset $X' \subseteq X$. (3) Step 3 constructs a subset $F_1 = \{e \mid x_e \in X'\}$ $(\subseteq E)$ in time

$O(m)$. (4) By Lemma 3, the MST algorithm [34, 35] at Step 4 needs time $O(n^2)$ to compute a minimum-weight subset $F_2 \subseteq E$ such that $G[F_1 \cup F_2]$ is a connected spanning subgraph in G. (5) For the case where there exists an odd-degree vertex in $G[F_1 \cup F_2]$, using the Floyd algorithm [36], we can construct the graph H at Step 5 in time $O(n^3)$ due to Lemma 4, and the Edmonds algorithm [2] at Step 5 needs time $O(n^3)$ to produce a minimum-weight perfect matching M in H due to Lemma 5. (6) By Lemma 6, the Euler algorithm [3] at Step 7 needs time $O(m)$ to construct an Euler tour C in $G[F_1 \cup F_2 \cup F_3]$. (7) The TA algorithm at Step 8 needs at most time $O(n^3)$. Hence, the k-CPIBC algorithm needs whole time $O(n^3 + f(n, m))$.

This establishes the theorem.

4 The k-Chinese Postman Problem Under Interdiction Cardinality Constraints

In this section, we consider the k-CPIBC problem specialized to connected graphs with unit costs, i.e., the special version of the k-CPIBC problem where an interdiction cost function $c(\cdot) \equiv 1$ holds. For convenience, we refer this version as the k-Chinese postman problem under interdiction cardinality constraints (the k-CPICC problem).

Given a graph $G = (V, E; w, \mathbf{1}; v_1)$ as an instance of the k-CPICC problem, we can choose $m - B$ least-weight edges F_1' from E in time $O(m \log m)$. And it is easy to verify that F_1' is a minimum-weight edge-subset with $|F_1'| \geqslant m - B$ in E, implying that we may choose $\alpha = 1$ in the k-CPIBC algorithm for an interdiction cost function $c(\cdot) \equiv 1$. Using Theorem 2, we obtain the fact that there exists a $(\beta + 1)$-approximation algorithm in running time $O(n^3)$ to solve the k-CPICC problem, where $\beta = \frac{7}{2} - \frac{1}{k} - \lfloor \frac{1}{k} \rfloor$.

In addition, we use the Fisher algorithm [38] to solve the minimum-weight spanning K-tree problem, then we design a better approximation algorithm to solve the k-CPICC problem using the following strategies: (1) Determine a minimum-weight subset $F_{1,2} \subseteq E$, such that $|F_{1,2}| \geqslant m - B$ and that $G[F_{1,2}]$ is a connected spanning subgraph in G; (2) find a minimum-weight subset $F_3 \subseteq E$, such that $G[F_{1,2} \cup F_3]$ is an Eulerian graph; (3) compute a subset $F_4 \subseteq E$, such that the weight of the optimal k-tours covering in $G[F_{1,2} \cup F_3 \cup F_4]$ is as small as possible; and (4) output the subset $S_k = E \backslash (F_{1,2} \cup F_3 \cup F_4)$.

Our approximation algorithm, denoted by the k-CPICC algorithm, for the k-CPICC problem is described as follows.

Theorem 3 *The k-CPICC algorithm is a β-approximation algorithm to solve the k-CPICC problem, and this algorithm runs in time $O(n^3)$, where $\beta = \frac{7}{2} - \frac{1}{k} - \lfloor \frac{1}{k} \rfloor$.*

Proof Given an instance $G = (V, E; w, \mathbf{1}; v_1)$ of the k-CPICC problem, we may assume that S_k^* is an optimal solution with an optimal k-tours covering $\mathcal{C}^k_{E \backslash S_k^*}$ in $G \setminus S_k^*$ and the optimal value $\mathrm{OPT}_k = w_{\max}(\mathcal{C}^k_{E \backslash S_k^*})$. And let S_k $(= E \setminus (F_{1,2} \cup F_3 \cup F_4))$ denote the edge-subset produced by the k-CPICC algorithm with a k-tours covering $\mathcal{C}^k_{E \backslash S_k}$ in the subgraph $G \setminus S_k$ and the output value $\mathrm{OUT}_k = w_{\max}(\mathcal{C}^k_{E \backslash S_k})$.

Algorithm 3 : k-CPICC

Input: A connected graph $G = (V, E; w, \mathbf{1}; v_1)$ with a depot $v_1 \in V$, an integer $k \in \mathbb{Z}^+$ and a budget $B \in \mathbb{N}$;
Output: A subset $S_k \subset E$ such that $\min_{\mathcal{C}_{E\setminus S_k}} \max\{w(C_i) \mid C_i \in \mathcal{C}_{E\setminus S_k}\}$ is as small as possible.

```
Begin
Step 1 Use the Fisher algorithm [38] to find a connected spanning subgraph $G[F_{1,2}]$ in $G$, having $|F_{1,2}| \geqslant m - B$, such that $w(F_{1,2})$ is minimized;
Step 2 If (there exists an odd-degree vertex in $G[F_{1,2}]$) then
    (2.1) Denote by $V_o$ the set of all odd-degree vertices in $G[F_{1,2}]$; Construct a complete subgraph $H = (V_o, E_H; w_H)$ on $V_o$, where weight $w_H(v_i v_j)$ of edge $v_i v_j \in E_H$ is the weight of a shortest path $P_{v_i,v_j}$ connecting $v_i$ and $v_j$ in $G$;
    (2.2) Use the Edmonds algorithm [2] to find a minimum-weight perfect matching $M$ in $H$;
    (2.3) Denote $F_3 = \bigcup_{v_i v_j \in M} E(P_{v_i,v_j})$, where $E(P_{v_i,v_j})$ denotes the set of all edges that lie on the shortest path $P_{v_i,v_j}$;
Step 3 If ($k = 1$) then
        Output the subset $S_1 = E \setminus (F_{1,2} \cup F_3)$, and STOP.
Step 4 Use the Euler algorithm [3] to determine an Euler tour $C$ in $G[F_{1,2} \cup F_3]$ that starts and ends at the same depot $v_1$, where we may assume $C = v_1 v_{i_1} v_{i_2} \cdots v_{i_t} v_1$;
Step 5 Use the TA algorithm to compute a subset $F_4 \subseteq E$, such that the weight of the optimal $k$-tours covering in $G[E(C) \cup F_4]$ is as small as possible;
Step 6 Output the subset $S_k = E \setminus (F_{1,2} \cup F_3 \cup F_4)$.
End
```

For each integer $k \in \mathbb{Z}^+$, executing Step 1, we have that $|S_k| = |E\setminus(F_{1,2} \cup F_3 \cup F_4)| \leqslant |E\setminus F_{1,2}| \leqslant m - (m - B) = B$ and $G\setminus S_k \supseteq G[F_{1,2}]$, implying that $G \setminus S_k$ is connected. This shows that S_k is a feasible solution to the instance G of the k-CPICC problem. We shall prove $\text{OUT}_1 \leqslant \frac{3}{2} \cdot \text{OPT}_1$ for the case $k = 1$ and $\text{OUT}_k \leqslant (\frac{7}{2} - \frac{1}{k}) \cdot \text{OPT}_k$ for the case $k \geqslant 2$, respectively.

Case 1 $k = 1$

For each optimal solution S_1^* to the graph G as an instance of the 1-CPICC problem, we have that $G \setminus S_1^*$ is a connected spanning subgraph in G with $|E\setminus S_1^*| \geqslant m - B$. Since $G[F_{1,2}]$ produced at Step 1 is a minimum-weight connected spanning subgraph with $|F_{1,2}| \geqslant m - B$ in G, we obtain the following:

$$w(F_{1,2}) \leqslant w(E \setminus S_1^*) \leqslant w(\mathcal{C}^1_{E\setminus S_1^*}).$$

Using the similar arguments as in [44] and in the proof of Theorem 2, we can use $\mathcal{C}^1_{E\setminus S_1^*}$ to construct a Hamiltonian cycle $C_o = v_{j_1} v_{j_2} \cdots v_{j_{|V_o|}} v_{j_1}$ in $H = (V_o, E_H; w_H)$ satisfying $w_H(C_o) \leqslant w(\mathcal{C}^1_{E\setminus S_1^*})$. In addition, we can construct two perfect matchings in H, denoted by $M_1 = \{v_{j_i} v_{j_{i+1}} \mid i = 1, 3, \cdots, |V_o| - 1\}$ and $M_2 = \{v_{j_{|V_o|}} v_{j_1}\} \cup \{v_{j_i} v_{j_{i+1}} \mid i = 2, 4, \cdots, |V_o| - 2\}$. Since M produced at Step 2 is a minimum-weight perfect matching in H, we have $w_H(M) \leqslant \min\{w_H(M_1), w_H(M_2)\} \leqslant \frac{1}{2} w_H(C_o)$. This implies $w(F_3) = w_H(M) \leqslant \frac{1}{2} w_H(C_o) \leqslant \frac{1}{2} w(\mathcal{C}^1_{E\setminus S_1^*})$.

Using Steps 1–2, we have that $G[F_{1,2} \cup F_3]$ is an Eulerian graph. And by the fact $S_1 = E \setminus (F_{1,2} \cup F_3)$ produced by the k-CPICC algorithm, we have that $\text{OUT}_1 =$

$w(\mathcal{C}^1_{E\backslash S_1}) \leqslant w(F_{1,2}) + w(F_3)$. Thus, we obtain the following:

$$\mathrm{OUT}_1 \leqslant w(F_{1,2}) + w(F_3) \leqslant w(\mathcal{C}^1_{E\backslash S_1^*}) + \frac{1}{2} w(\mathcal{C}^1_{E\backslash S_1^*}) = \frac{3}{2} \cdot w(\mathcal{C}^1_{E\backslash S_1^*}) = \frac{3}{2} \cdot \mathrm{OPT}_1.$$

Case 2 $k \geqslant 2$

Using the similar arguments in Case 1, we obtain $F_{1,2} \cup F_3 = E(C)$, and $w(C) = w(F_{1,2}) + w(F_3) \leqslant \frac{3}{2} \cdot w(\mathcal{C}^1_{E\backslash S_1^*})$. We shall prove $\mathrm{OUT}_k \leqslant (\frac{7}{2} - \frac{1}{k}) \cdot \mathrm{OPT}_k$.

Since $F_{1,2} \cup F_3 = E(C)$ produced by the k-CPICC algorithm, we have $G[F_{1,2} \cup F_3 \cup F_4] = G[E(C) \cup F_4]$. Using Lemma 9, we can construct a k-tours covering $\{C_j \mid j = 1, 2, \cdots, k\}$ in $G[F_{1,2} \cup F_3 \cup F_4]$, i.e., this k-tours covering is in $G\backslash S_k$, and for each integer $j \in \{1, 2, \cdots, k\}$, this k-tours covering satisfies $w(C_j) \leqslant \frac{1}{k} \cdot (w(C) - 2w_0) + 4w_0$.

Using the similar arguments in the proof of Case 2 of Theorem 2, we have that $w(\mathcal{C}^1_{E\backslash S_1^*}) \leqslant w(\mathcal{C}^1_{E\backslash S_k^*}) \leqslant k \cdot w_{\max}(\mathcal{C}^k_{E\backslash S_k^*})$ and $w_0 \leqslant \frac{1}{2} w_{\max}(\mathcal{C}^k_{E\backslash S_k^*})$. Thus, for each $j \in \{1, 2, \cdots, k\}$, we obtain the following:

$$\begin{aligned} w(C_j) &\leqslant \frac{1}{k} \cdot (w(C) - 2w_0) + 4w_0 \\ &= 2 \cdot \left(2 - \frac{1}{k}\right) \cdot w_0 + \frac{1}{k} \cdot w(C) \\ &\leqslant 2 \cdot \left(2 - \frac{1}{k}\right) \cdot w_0 + \frac{1}{k} \cdot \frac{3}{2} \cdot w(\mathcal{C}^1_{E\backslash S_1^*}) \\ &\leqslant \left(2 - \frac{1}{k}\right) \cdot w_{\max}(\mathcal{C}^k_{E\backslash S_k^*}) + \frac{3}{2} \cdot w_{\max}(\mathcal{C}^k_{E\backslash S_k^*}) \\ &= \left(\frac{7}{2} - \frac{1}{k}\right) \cdot w_{\max}(\mathcal{C}^k_{E\backslash S_k^*}) \\ &= \left(\frac{7}{2} - \frac{1}{k}\right) \cdot \mathrm{OPT}_k, \end{aligned}$$

implying $\mathrm{OUT}_k = w_{\max}(\mathcal{C}^k_{E\backslash S_k}) \leqslant \max\{w(C_j) \mid j = 1, 2, \cdots, k\} \leqslant (\frac{7}{2} - \frac{1}{k}) \cdot \mathrm{OPT}_k$.

Combining the aforementioned arguments in Cases 1 and 2, we obtain the following:

$$\mathrm{OUT}_k \leqslant \beta \cdot \mathrm{OPT}_k,$$

where $\beta = \frac{7}{2} - \frac{1}{k} - \lfloor \frac{1}{k} \rfloor$ for each integer $k \in \mathbb{Z}^+$.

The complexity of the k-CPICC algorithm can be determined as follows: (1) By Lemma 7, the Fisher algorithm [38] at Step 1 needs time $O(n^2)$ to compute a minimum-weight connected spanning subgraph $G[F_{1,2}]$ with $|F_{1,2}| \geqslant m - B$. (2) By the similar arguments of the complexity of the k-CPIBC algorithm in Theorem 2, Steps 2–5 need at most time $O(n^3)$ to obtain F_3 and F_4, respectively. Hence, the k-CPICC algorithm needs the whole time $O(n^3)$.

This establishes the theorem.

5 Conclusion and Further Research

In this paper, we consider the k-Chinese postman problem under interdiction budget constraints (the k-CPIBC problem) and its special version (the k-CPICC problem) and obtain the following two main results:

(1) Given an α-approximation algorithm $\mathcal{A}_\alpha$ for solving the minimization knapsack problem, we design an $(\alpha + \beta)$-approximation algorithm to solve the k-CPIBC problem, where $\beta = \frac{7}{2} - \frac{1}{k} - \lfloor \frac{1}{k} \rfloor$;
(2) We present a β-approximation algorithm to solve the k-CPICC problem, where $c(e) \equiv 1$ for each edge e in $G = (V, E; w, \mathbf{1}; v_1)$ and β is defined in (1).

In further research, we shall consider some interdiction versions of other arc routing problems.

Acknowledgements We are indeed grateful to the two anonymous reviewers for their insightful comments and for their suggested changes that improve the presentation greatly.

Author Contributions J.-P. Li has proposed this problem and contributed to providing some ideas, methods, discussion and writing the final manuscript as the corresponding author. P.-X. Pan has finished some design of algorithms, theoretical proofs and the original manuscript. J.-R. Lichen, W.-C. Wang and L.-J. Cai have contributed to providing some ideas, analyses, discussions and revisions. All the authors have discussed the theoretical results, reviewed and approved the final manuscript.

Conflict of interest The authors declare no conflict of interest.

References

[1] Guan, M.G.: Graphic programming using odd or even points. Acta Math. Sin. **10**, 263–266 (1960). (**in Chinese**)
[2] Edmonds, J.: Maximum matching and a polyhedron with (0,1)-vertices. J. Res. Natl. Bureau Stand. B **69**, 125–130 (1965)
[3] Edmonds, J., Johnson, E.L.: Matching, Euler tours and the Chinese postman. Math. Program. **5**(1), 88–124 (1973)
[4] Frederickson, G.N., Hecht, M.S., Kim, C.E.: Approximation algorithms for some routing problems. SIAM J. Comput. **7**(2), 178–193 (1978)
[5] Beullens, P., Muyldermans, L., Cattrysse, D., Oudheusden, D.V.: A guided local search heuristic for the capacitated arc routing problem. Eur. J. Oper. Res. **147**(3), 629–643 (2003)
[6] Corberán, Á., Martí, R., Sanchis, J.M.: A GRASP heuristic for the mixed Chinese postman problem. Eur. J. Oper. Res. **142**(1), 70–80 (2002)
[7] Ding, H.L., Li, J.P., Lih, K.W.: Approximation algorithms for solving the constrained arc routing problem in mixed graphs. Eur. J. Oper. Res. **239**(1), 80–88 (2014)
[8] Jozefowiez, N., Semet, F., Talbi, E.G.: Multi-objective vehicle routing problems. Eur. J. Oper. Res. **189**, 293–309 (2008)
[9] Lysgaard, J., Wøhlk, S.: A branch-and-cut-and-price algorithm for the cumulative capacitated vehicle routing problem. Eur. J. Oper. Res. **236**, 800–810 (2014)
[10] Mourão, M.C., Amado, L.: Heuristic method for a mixed capacitated arc routing problem: a refuse collection application. Eur. J. Oper. Res. **160**, 139–153 (2005)
[11] Zachariadis, E.E., Kiranoudis, C.T.: Local search for the undirected capacitated arc routing problem with profits. Eur. J. Oper. Res. **210**(2), 358–367 (2011)
[12] Ghare, P.M., Montgomery, D.C., Turner, W.C.: Optimal interdiction policy for a flow network. Naval Res. Logist. Q. **18**, 37–45 (1971)
[13] Assimakopoulos, N.: A network interdiction model for hospital infection control. Comput. Biol. Med. **17**(6), 413–422 (1987)

[14] Wood, R.K.: Deterministic network interdiction. Math. Comput. Model. **17**(2), 1–18 (1993)
[15] Church, R.L., Scaparra, M.P., Middleton, R.S.: Identifying critical infrastructure: the median and covering facility interdiction problems. Ann. Assoc. Am. Geogr. **94**(3), 491–502 (2004)
[16] Salmeron, J., Wood, K., Baldick, R.: Analysis of electric grid security under terrorist threat. IEEE Trans. Power Syst. **19**(2), 905–912 (2004)
[17] Lin, K.C., Chern, M.S.: The most vital edges in the minimum spanning tree problem. Inform. Process. Lett. **45**(1), 25–31 (1993)
[18] Frederickson, G.N., Solis-Oba, R.: Increasing the weight of minimum spanning trees. J. Algorithms **33**(2), 244–266 (1999)
[19] Bazgan, C., Toubaline, S., Vanderpooten, D.: Critical edges/nodes for the minimum spanning tree problem: complexity and approximation. J. Comb. Optim. **26**(1), 178–189 (2013)
[20] Zenklusen, R.: An $O(1)$-approximation for minimum spanning tree interdiction. In: Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 709–728 (2015)
[21] Linhares, A., Swamy, C.: Improved algorithms for MST and metric-TSP interdiction. In: Proceedings of 44th International Colloquium on Automata, Languages, and Programming (ICALP) Art.No 32, 14pp (2017) https://doi.org/10.48550/arXiv.1706.00034
[22] Zenklusen, R.: Matching interdiction. Discrete Appl. Math. **158**(15), 1676–1690 (2010)
[23] Dinitz, M., Gupta, A.: Packing interdiction and partial covering problems. In: Proceedings of International Conference on Integer Programming and Combinatorial Optimization (IPCO'13), pp. 157–168 (2013)
[24] Pan, F., Schild, A.: Interdiction problems on planar graphs. Discrete Appl. Math. **198**, 215–231 (2016)
[25] Golden, B.: A problem in network interdiction. Naval Res. Logist. Q. **25**(4), 711–713 (1978)
[26] Ball, M.O., Golden, B.L., Vohra, R.V.: Finding the most vital arcs in a network. Oper. Res. Lett. **8**(2), 73–76 (1989)
[27] Israeli, E., Wood, R.K.: Shortest-path network interdiction. Networks **40**(2), 97–111 (2002)
[28] Phillips, C.A.: The network inhibition problem. In: Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing (STOC'93), pp. 776–785 (1993)
[29] Zenklusen, R.: Network flow interdiction on planar graphs. Discrete Appl. Math. **158**(13), 1441–1455 (2010)
[30] Zenklusen, R.: Connectivity interdiction. Oper. Res. Lett. **42**(6–7), 450–454 (2014)
[31] Armon, A., Zwick, U.: Multicriteria global minimum cuts. Algorithmica **46**, 15–16 (2006)
[32] Johnson, D.S., Niemi, K.A.: On knapsacks, partitions, and a new dynamic programming technique for trees. Math. Oper. Res. **8**, 1–14 (1983)
[33] Smith, J.C., Song, Y.J.: A survey of network interdiction models and algorithms. Eur. J. Oper. Res. **283**(3), 797–811 (2020)
[34] Dijkstra, E.W.: A note on two problems in connexion with graphs. Numer. Math. **1**, 269–271 (1959)
[35] Prim, R.C.: Shortest connection networks and some generalizations. Bell Syst. Tech. J. **36**(6), 1389–1401 (1957)
[36] Floyd, R.W.: Algorithm 97: shortest path. Commun. ACM **5**(6), 345 (1962)
[37] Gabow, H.N.: Data structures for weighted matching and extensions to b-matching and f-factors. ACM Trans. Algorithms **14**(3), Art. 39, 80pp (2018)
[38] Fisher, M.L.: A polynomial algorithm for the degree-constrained minimum K-tree problem. Oper. Res. **42**(4), 775–779 (1994)
[39] Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman and Co., New York (1979)
[40] Güntzer, M.M., Jungnickel, D.: Approximate minimization algorithms for the 0/1 knapsack and subset-sum problem. Oper. Res. Lett. **26**(2), 55–66 (2000)
[41] Kellerer, H., Pferschy, U., Pisinger, D.: Knapsack Problems. Springer, Berlin (2004)
[42] Csirik, J., Frenk, J.B.G., Labbé, M., Zhang, S.: Heuristics for the 0–1 min-knapsack problem. Acta Cybern. **10**(1–2), 15–20 (1991)
[43] Carnes, T., Shmoys, D.B.: Primal-dual schema for capacitated covering problems. Math. Program. **153**(2), 289–308 (2015)
[44] Christofides, N.: Worst-case analysis of a new heuristic for the travelling salesman problem. Report 388. Graduate School of Industrial Administration, Carnegie Mellon University (1976)