RAIRO-Oper. Res. 56 (2022) 1283–1291
https://doi.org/10.1051/ro/2022044

RAIRO Operations Research
www.rairo-ro.org

AN IMPROVED HEURISTIC ALGORITHM FOR THE MAXIMUM BENEFIT CHINESE POSTMAN PROBLEM

Shiori Matsuura and Kenjiro Takazawa*

Abstract. The maximum benefit Chinese postman problem (MBCPP) is a practical generalization of the Chinese postman problem. A distinctive feature of MBCPP is that the postman can traverse each edge arbitrary times and obtains a benefit for each traversal of an edge, which depends on the number of times that the edge has been traversed. (Pearn and Wang, *Omega* **31** (2003) 269–273) discussed MBCPP under the assumption that the benefits of each edge is a non-increasing function of the number of traversals. They showed that MBCPP under this assumption is NP-hard, and proposed a heuristic algorithm which applies the minimum spanning tree and the minimal-cost T-join algorithms. (Corberán, Plana, Rodríguez-Chía and Sanchis, *Math. Program.* **141** (2013) 21–48) presented an integer programming formulation and a branch-and-cut algorithm for MBCPP without the assumption on the benefits. This is based on the idea of integrating the benefits of each edge into two benefits, each representing that the edge is traversed an odd or even number of times. In this paper, by applying the idea of Corberán *et al.*, we improve the heuristic algorithm of Pearn and Wang. Our algorithm applies to the general case with no assumption on the benefits, and can perform better even if the benefits are non-increasing. We then analyze the efficiency of our heuristic algorithm in theory and in practice, and prove that it finds the optimal solution when the benefits satisfy a certain property.

Mathematics Subject Classification. 90C59, 90C27, 05C85.

Received March 15, 2021. Accepted March 18, 2022.

1. Introduction

The *Chinese postman problem* [6, 9] is a problem well known to be solved in polynomial time, and many generalizations are proposed. One of such generalizations is the *maximum benefit Chinese postman problem* (*MBCPP*), which was proposed by Malandraki and Daskin [13] in directed graphs. In this paper, we deal with MBCPP in undirected graphs, which was first studied by Pearn and Wang [14].

1.1. Problem definition

An intuitive description of MBCPP in undirected graphs is as follows. A postman traverses edges while offering service to the edges. Each edge is endowed with a set of *benefits* that are obtained at each traversal with service, as well as a *service cost* for a traversal with service and a *deadhead cost* for a traversal without

Keywords. Combinatorial optimization, maximum benefit Chinese postman problem, heuristic algorithm, spanning tree, T-join.

Department of Industrial and Systems Engineering, Faculty of Science and Engineering, Hosei University, Koganei-shi, Tokyo, Japan.
*Corresponding author: takazawa@hosei.ac.jp

service. Namely, the postman obtains a *net benefit*, which is the benefit minus the service cost at a traversal with service, or the negative of the deadhead cost at a traversal without service. The objective of MBCPP is to find a postman tour starting from a depot vertex and returning to the depot that maximizes the total net benefit. A distinctive feature of MBCPP is that there is no limit to the number of times each edge can be traversed. In other words, an edge may be traversed multiple times, or not at all. Furthermore, the benefit obtained at each traversal of an edge depends on the number of services that have been offered to the edge. The MBCPP finds its applications in the design of street cleaning, snow-plowing, snow-salting, and so on.

Formally, MBCPP is defined in the following way. Let $G = (V, E)$ be a connected undirected graph with vertex set V and edge set E, and let $v_0 \in V$ be a specified vertex, representing the depot. Each edge $e \in E$ is associated with an integer $k_e \in \mathbb{Z}_+$ representing the number of services offered to e. Services are offered for the first k_e traversals, and no service is offered afterwards. Thus, for the tth traversal on e, we obtain a benefit $b_e(t) \geq 0$ for $t = 1, 2, \ldots, k_e$, and no benefit for $t \geq k_e + 1$.

Costs are also associated with each traversal of an edge: service cost associated with a traversal with service; and deadhead cost associated with a traversal without service. The service cost of an edge e varies with the number of traversals with service on e. For $t = 1, 2, \ldots, k_e$, let $s_e(t)$ denote the cost of the tth traversal of e. The deadhead cost of e does not depend on the number of traversals and is denoted by d_e. We assume that $s_e(t) \geq 0$ $(t = 1, 2, \ldots, k_e)$ and $d_e \geq 0$.

In summary, the net benefit $\bar{b}_e(t)$ of the tth traversal of an edge e is described as

$$\bar{b}_e(t) = \begin{cases} b_e(t) - s_e(t) & (t = 1, 2, \ldots, k_e), \\ -d_e & (t \geq k_e + 1). \end{cases} \tag{1.1}$$

The objective of MBCPP is to find a spanning Eulerian multi-subgraph (U, F) of G with depot $v_0 \in U$ maximizing the total net benefit

$$\sum_{e \in E} \sum_{t=1}^{m_F(e)} \bar{b}_e(t),$$

where $m_F(e) \in \mathbb{Z}_+$ is the multiplicity of an edge e in the multiset F.

1.2. Related work

The two studies most related to this paper are due to Pearn and Wang [14] and Corberán, Plana, Rodríguez-Chía, and Sanchis [4].

Pearn and Wang [14] proved that MBCPP is NP-hard even when the benefit $b_e(t)$ is a monotone non-increasing function of the number t of traversals and $s_e(t)$ is constant for each $e \in E$, by reducing the *rural postman problem*, which had been proved to be NP-hard [12]. Pearn and Wang [14] further proposed a heuristic algorithm for MBCPP with the above benefits, while its complexity is not analyzed and only a small computational example is provided.

This heuristic algorithm uses a minimum spanning tree and a minimum-weight T-join algorithms (see, *e.g.*, [10, 16]), and has some similarity to the algorithm for the Chinese postman problem [7] and Christofides' approximation algorithm for the traveling salesman problem [3]. In a graph $G = (V, E)$ and its vertex subset $T \subseteq V$, an edge subset $J \subseteq E$ is a *T-join* if the set of vertices of odd degree in the subgraph (V, J) is T. Note that, if T is equal to the set of vertices of odd degree in the original graph G, then adding a T-join J to G results in a Eulerian (multi)graph.

Corberán *et al.* [4] presented an integer programming formulation and a branch-and-cut algorithm for MBCPP without any assumption on the benefits. A key idea is to integrate the net benefits of each edge into only two benefits, each representing that the edge is traversed in an odd or even number of times.

Theorem 1.1 (Corberán *et al.* [4])**.** *Solving an arbitrary instance I of MBCPP is equivalent to solving an instance I', where each edge e has only two net benefits b_e^{odd} and b_e^{even} for the first and the second traversals of*

each edge e, respectively, where

– *If $k_e \geq 1$,*

$$b_e^{odd} = \max\left\{\sum_{t=1}^{\ell} \bar{b}_e(t) \colon \ell \leq k_e + 1,\ \ell \text{ is odd}\right\},$$

$$b_e^{even} = \max\left\{\sum_{t=1}^{\ell} \bar{b}_e(t) \colon \ell \leq k_e + 1,\ \ell \text{ is even}\right\} - b_e^{odd},$$

– *If $k_e = 0$, $b_e^{odd} = b_e^{even} = -d_e$.*

Theorem 1.1 implies that, by integrating the net benefits on each edge, any instance of MBCPP can be transformed into an instance with only zero, one or two traversals on each edge. Namely, we do not need to define the net benefit of the tth traversal with $t \geq 3$.

While this integer programming approach does not have an effective bound on the complexity, Corberán *et al.* [4] empirically demonstrated its practical efficiency, by showing that it can solve instances with up to 1,000 vertices and 3,000 edges within one hour.

Other related work includes the following. As mentioned before, Malandraki and Daskin [13] dealt with MBCPP in directed graphs. They modeled it as a minimum-cost flow problem with subtour elimination constraints, and proposed a branch-and-bound procedure. Pearn and Chiu [15] also proposed a heuristic algorithm for MBCPP in directed graphs. The *prize-collecting arc routing problem* [1,2] is a special case of MBCPP, where $k_e = 1$ for each edge e in some edge subset $D \subseteq E$ and $k_e = 0$ for each $e \in E \setminus D$. Shafahi and Haghani [18] proposed a common generalization of MBCPP and the vehicle routing problem, and presented a mixed integer programming formulation.

1.3. Our contribution

In this paper, by applying the idea of Corberán *et al.* [4] (Thm. 1.1), we extend the heuristic algorithm of Pearn and Wang [14] to the general case where the net benefit $\bar{b}_e(t)$ is not necessarily a monotone non-increasing function of the number t of traversals. We show that, even when $\bar{b}_e(t)$ is monotone non-increasing, in some cases our algorithm performs better than the previous algorithm; see Section 3.1.2.

While there already exists a computational method [4], our heuristic algorithm is of interest in a sense that it builds upon fundamental concepts in combinatorial optimization such as minimum spanning trees and minimum-weight T-joins. This fact provides the ease of implementation and estimation of the efficiency in theory and in practice: given the integrated benefits b_e^{odd} and b_e^{even} of each edge e, our algorithm runs in $O(n^3)$ time, where n is the number of vertices. This is the first algorithm for MBCPP with effective complexity bound.

Furthermore, we prove that the extended heuristic algorithm finds the optimal solution when the integrated net benefit for the first traversal is larger than or equal to that for the second traversal, and the sum of these two integrated net benefits is non-negative (Assumption 3.3 in Sect. 3.3). This property is satisfied, for instance, when $\bar{b}_e(t)$ is monotone non-increasing for each edge e.

We summarize the aforementioned features of the algorithms in Table 1.

TABLE 1. Summary of the algorithms for MBCPP.

	Solution	Assumption	Complexity	Method
Pearn and Wang [14]	Heuristic	$\bar{b}_e(t)$: non-increasing	Not Analyzed	Combinatorial
Corberán *et al.* [4]	Exact	None	Not Analyzed	Branch-and-Cut
This paper	Heuristic	None	$O(n^3)$	Combinatorial

1.4. Organization of the paper

The organization of this paper is as follows. We describe our extended heuristic algorithm in Section 2. Section 3 provides the detail of its improvements: improvements in the heuristics in Section 3.1; theoretical and practical efficiency in Section 3.2; and the optimality when the benefits obey the above assumption in Section 3.3. Section 4 concludes this paper by a summary and future work.

2. Improved heuristic algorithm

In this section, we describe our heuristic algorithm, which extends the algorithm by Pearn and Wang [14], by using the idea of the integration of the net benefits by Corberán *et al.* [4] (Thm. 1.1). We begin with a brief sketch to provide an intuition of the algorithm in Section 2.1, followed by its detailed description in Section 2.2.

2.1. Algorithm sketch

For each edge $e \in E$, make two copies e' and e'', and associate the benefits $b_{e'}$ and $b_{e''}$ by $b_{e'} = b_e^{\mathrm{odd}}$ and $b_{e''} = b_e^{\mathrm{even}}$ defined in Theorem 1.1, each representing the first and second traversal of e. Denote $E' = \{e' \mid e \in E\}$ and $E'' = \{e'' \mid e \in E\}$.

Now the objective is to find a subset $F \subseteq E' \cup E''$ which forms a connected Eulerian subgraph including v_0 and maximizes the total benefit $\sum_{f \in F} b_f$. A remarkable feature is that F must satisfy

$$e'' \in F \Longrightarrow e' \in F \tag{2.1}$$

for every $e \in E$.

We initialize F to be the set of the edges $f \in E' \cup E''$ with non-negative benefit b_f, except for edges e'' with $b_{e'} < 0$. That is,

$$F = \{e' \in E' \mid b_{e'} \geq 0\} \cup \{e'' \in E'' \mid b_{e''} \geq 0,\ b_{e'} \geq 0\}.$$

Next we modify the edge set F so that it induces a connected and Eulerian subgraph. If the subgraph induced by F is not connected, we connect the components by adding some edges with negative benefit, with the aid of a minimum spanning tree algorithm. We then add a minimum-weight T-join to F, where T is the set of vertices of odd degree in (V, F), to obtain an Eulerian graph. We remark that, the weights of the edges in finding the minimum spanning tree and the T-join are carefully defined, based on b_e^{odd} and b_e^{even}.

Finally, we remove cycles of negative benefit from F as many as possible, while maintaining the condition (2.1) and the connectivity of F.

2.2. Algorithm description

We exhibit a pseudocode of our algorithm in Algorithm 1, followed by its detailed description.

Line 1 (Collecting non-negative benefits). For each edge $e \in E$, define $b_e^{\mathrm{odd}}, b_e^{\mathrm{even}} \in \mathbb{R}$ in the same way as Theorem 1.1. According to the values of b_e^{odd} and b_e^{even}, partition the edge set E into four sets E_1, E_2, E_3, and E_4 by

$$\begin{aligned}
E_1 &= \{e \in E \mid b_e^{\mathrm{odd}} \geq 0,\ b_e^{\mathrm{even}} \geq 0\},\\
E_2 &= \{e \in E \mid b_e^{\mathrm{odd}} \geq 0 > b_e^{\mathrm{even}}\},\\
E_3 &= \{e \in E \mid b_e^{\mathrm{odd}} < 0 \leq b_e^{\mathrm{even}}\},\\
E_4 &= \{e \in E \mid b_e^{\mathrm{odd}} < 0,\ b_e^{\mathrm{even}} < 0\}.
\end{aligned}$$

Let E' and E'' be two copies of E. Denote the copy of $e \in E$ in E' by e', and its benefit $b_{e'} = b_e^{\mathrm{odd}}$. Similarly, let e'' be the copy of $e \in E$ in E'' and let $b_{e''} = b_e^{\mathrm{even}}$. Then, define $F \subseteq E' \cup E''$ by

$$F = \bigcup_{e \in E_1} \{e', e''\} \cup \bigcup_{e \in E_2} \{e'\}.$$

Algorithm 1 Extended heuristic algorithm.

1: $F \leftarrow \bigcup_{e\in E_1}\{e', e''\} \cup \bigcup_{e\in E_2}\{e'\}$
2: **if** (V, F) is not connected **then**
3: $\quad \tilde{S} \leftarrow$ Minimum spanning tree in $\tilde{G}$ w.r.t. length $-\tilde{c}$
4: $\quad F \leftarrow F \cup S' \cup S''$
5: **if** (V, F) is not Eulerian **then**
6: $\quad T \leftarrow$ Set of the vertices of odd degree in (V, F)
7: $\quad J \leftarrow$ Maximum-weight T-join w.r.t. weight w
8: $\quad$ Update F by (2.5)
9: $\mathcal{C} \leftarrow$ Maximal family of edge-disjoint cycles in (V, F) with negative benefit
10: **for** $C \in \mathcal{C}$ **do**
11: $\quad$ **if** $(V, F \setminus C)$ is connected and $e'' \in F \setminus C$ implies $e' \in F \setminus C$ **then**
12: $\quad\quad F \leftarrow F \setminus C$
13: Output F

Lines 2–4 (Connecting the components). Suppose that (V, F) is not connected. Let $\mathcal{K}$ denote the set of connected components in (V, F), and let $\tilde{G} = (\mathcal{K}, \tilde{E})$ be a graph with vertex set $\mathcal{K}$, whose edge set $\tilde{E}$ and edge-length function $\tilde{c}$ are defined as follows. The edge set $\tilde{E}$ is defined by

$$\tilde{E} = \{(K_1, K_2) \mid K_1, K_2 \in \mathcal{K},\ G \text{ has an edge } e \in E_3 \cup E_4 \text{ connecting } K_1 \text{ and } K_2\}.$$

For every edge $e \in E$ connecting two components in $\mathcal{K}$, define its weight $c_e \in \mathbb{R}$ by

$$c_e = \begin{cases} b_e^{\mathrm{odd}} + b_e^{\mathrm{even}} & (e \in E_3), \\ b_e^{\mathrm{odd}} & (e \in E_4), \end{cases} \tag{2.2}$$

representing that $e \in E_3$ is traversed twice, while $e \in E_4$ once. Now the edge-length function $\tilde{c}$ on $\tilde{E}$ is defined by

$$\tilde{c}(K_1, K_2) = \max\{c_e \colon e \in E \text{ is an edge between } K_1, K_2 \in \mathcal{K}\}. \tag{2.3}$$

for each $(K_1, K_2) \in \tilde{E}$.

Then, find a minimum spanning tree $\tilde{S}$ in $\tilde{G}$ with respect to $-\tilde{c}$, and let $S \subseteq E$ be the set of edges corresponding to $\tilde{S}$. Finally, define $S' \subseteq E'$ and $S'' \subseteq E''$ by

$$S' = \{e' \mid e \in S \cap (E_3 \cup E_4)\}, \qquad S'' = \{e'' \mid e \in S \cap E_3\},$$

and update F by $F := F \cup S' \cup S''$.

Lines 5–8 (Correcting the parity). Suppose that the graph (V, F) is not Eulerian. Let $T \subseteq V$ be the set of vertices with odd degree in (V, F). For each $e \in E$, define a new weight $w_e \in \mathbb{R}$ by

$$w_e = \begin{cases} -b_e^{\mathrm{even}} & (e', e'' \in F), \\ b_e^{\mathrm{even}} & (e' \in F,\ e'' \notin F), \\ b_e^{\mathrm{odd}} & (e', e'' \notin F). \end{cases} \tag{2.4}$$

Then, find a maximum-weight T-join $J \subseteq E$ with respect to w, and update F by

$$F := \left(F \setminus \bigcup_{e\in J\colon e', e''\in F} \{e''\}\right) \cup \left(\bigcup_{e\in J\colon e'\in F, e''\notin F} \{e''\}\right) \cup \left(\bigcup_{e\in J\colon e', e''\notin F} \{e'\}\right). \tag{2.5}$$

Lines 9–12 (Removing negative cycles). Find a maximal family $\mathcal{C}$ of edge-disjoint cycles in F with negative benefit, *i.e.*, $\sum_{f\in C} b_f < 0$ for each $C \in \mathcal{C}$. Then, delete as many cycles in $\mathcal{C}$ as possible, provided that the remaining graph (V, F) is connected and each edge $e \in E$ with $e'' \in F$ satisfies $e' \in F$.

3. Detail of improvements

In this section, we describe specific features of Algorithm 1. Section 3.1 explains the improvements upon the algorithm by Pearn and Wang [14]. In Section 3.2, we analyze its efficiency in theory and in practice, by giving the details of implementation. Finally, in Section 3.3, we prove that Algorithm 1 finds an optimal solution under a certain assumption described by the integrated net benefits b_e^{odd} and b_e^{even} (Assumption 3.3 below).

3.1. Improvements in the heuristics

3.1.1. Non-monotone benefits

The most remarkable improvement upon the algorithm by Pearn and Wang [14] is that we have removed the assumption that the net benefit $\overline{b}_e(t)$ is a monotone non-increasing function of the number t of traversals. This assumption is indeed necessary in their algorithm for the following two reasons.

The first reason lies in the initialization of F. Their algorithm makes a copy of e for each traversal of e, and collects the edges with non-negative net benefit $\overline{b}_e(t)$. The MBCPP demands that, if an edge representing the tth traversal is collected, then the edges representing the t'th traversal for every $t' \le t$ must also be collected. This is why their algorithm requires the net benefit $\overline{b}_e(t)$ to be monotone non-increasing.

The second reason comes from the spanning tree, whose counterpart in Algorithm 1 appears in Line 3. In the algorithm of Pearn and Wang [14], for every edge e in $\mathcal{K}$ its weight is defined as $\overline{b}_e(1)$, which is negative. Thus, if we apply the algorithm of Pearn and Wang for the case where the net benefits $\overline{b}_e(t)$ is not monotone, we cannot collect the non-negative benefits of the later traversals. In other words, the algorithm of Pearn and Wang cannot reasonably take the edges of type E_3 into account: indeed, $E_3 = \emptyset$ in the monotone case.

We have resolved these two issues by applying the idea of Corberán *et al.* [4]. For the first issue, Algorithm 1 just makes two copies e' and e'' of each edge $e \in E$ and associates the two integrated net benefits b_e^{odd} and b_e^{even}, respectively, as in Theorem 1.1. Then, the algorithm collects the edges with non-negative benefit except for the edges e'' of $e \in E_3$, obeying that e' must be collected if e'' is collected. This means that we can initialize F with the best multiplicities of each edge, regardless of whether $\overline{b}_e(t)$ is monotone or not.

The integrated benefits also resolved the second issue. In Lines 2–4 of Algorithm 1, the weights of the edges in $E_3 \cup E_4$ are defined by (2.2). This definition again means that we are choosing the best multiplicity for the edges which have not been used in the initialization of F.

3.1.2. Edge deletion in correcting the parity

The second improvement appears in Lines 5–8. Here Algorithm 1 updates the solution F to be Eulerian by a T-join J according to (2.5). This update (2.5) means that, in order to attain the desired parity, we can either add or delete an edge, and choose the better one. In contrast, in the algorithm of Pearn and Wang [14], J consists of edges corresponding to the next traversal. This means we can only add edges but cannot delete them.

Again, this idea of [14] is reasonable only when the net benefit function $\overline{b}_e(t)$ is monotone. Furthermore, even if $\overline{b}_e(t)$ is monotone, in some cases Algorithm 1 performs better, as shown in the next example.

Example 3.1. Let $\overline{b}_{e_1}(1) = 5$, $\overline{b}_{e_1}(2) = 4$, $\overline{b}_{e_1}(3) = 1$, and $\overline{b}_{e_1}(4) = -3$ for an edge $e_1 \in E$. We have that $b_{e_1}^{\mathrm{odd}} = 10$ and $b_{e_1}^{\mathrm{even}} = -1$, meaning that e_1 shall be traversed three times or twice. Suppose that $e_1' \in F$ and $e_1'' \notin F$ in Line 5, which means that at this point e_1 is traversed three times. Suppose further that $e_1 \in J$. Then, Algorithm 1 adds e_1'' to F in Line 8, which means that we come to traverse e_1 twice by canceling the third traversal of e_1, to obtain total benefit 9 from e_1. In contrast, the algorithm of Pearn and Wang [14] adds the fourth traversal, resulting to the total net benefit 7 from e_1.

Consider another edge $e_2 \in E$ with $\overline{b}_{e_2}(1) = 4$, $\overline{b}_{e_2}(2) = 3$, $\overline{b}_{e_2}(3) = 2$, $\overline{b}_{e_2}(4) = 1$, and $\overline{b}_{e_2}(5) = -3$. We have that $b_{e_2}^{\mathrm{odd}} = 9$ and $b_{e_2}^{\mathrm{even}} = 1$, meaning that e_2 shall be traversed three times or four times. Suppose that

$e_2', e_2'' \in F$ in Line 5, which means that at this point e_2 is traversed four times. Suppose further that $e_2 \in J$. Then Algorithm 1 cancels the fourth traversal to obtain total net benefit 9 from e_2, while the algorithm in [14] adds the fifth traversal, resulting in total net benefit 7.

Let us mention one more difference. In the algorithm of Pearn and Wang [14], the addition of the last traversal may create a negative cycle consisting of two parallel edges. Indeed, in Example 3.1, the third and fourth traversals of e_1 and the fourth and fifth traversals of e_2 result in negative cycles of two edges. Such negative cycles may be removed in the next step if it is detected, or may not if some other negative cycle including one of these two edges is detected. In other words, Algorithm 1 removes such negative cycles of two edges in advance.

3.2. Theoretical and practical efficiency

This section provides how to implement Algorithm 1 by combining basic graph algorithms. This enables us to give a theoretical bound on the time complexity, and discuss practical efficiency compared to the integer programming approach by Corberán *et al.* [4].

Recall that the number of the vertices in the original graph $G = (V, E)$ is denoted by n. Algorithm 1 mainly consists of the following three procedures:

Lines 2–4 finding a minimum spanning tree;
Lines 5–8 finding a maximum-weight T-join; and
Lines 9–12 finding negative cycles.

As is well known, all of these three procedures can be performed in polynomial time. First, a standard implementation of Prim's minimum spanning tree algorithm runs in $O(n^2)$ time, and many algorithms with better theoretical complexity bound are in the literature; see [10, 16, 17].

To find a maximum-weight T-join, it suffices to have implementations of an all-pairs shortest paths algorithm with nonnegative length and a maximum-weight matching algorithm [10,16]. For the former, the Warshall-Floyd method is standard and runs in in $O(n^3)$ time [10, 16, 17]. For the latter, an $O(n^3)$-time implementation [8, 11] of Edmonds' algorithm [5] is famous.

Finally, a family of negative cycles can also be found by one execution of a minimum-weight T-join algorithm in the following way. For two sets X and Y, denote their symmetric difference $(X \setminus Y) \cup (Y \setminus X)$ by $X \triangle Y$. Let $F^- \subseteq F$ be a set of the edges in F with negative benefit, *i.e.*, $F^- = \{f \in F \mid b_f < 0\}$. Let $T^- \subseteq V$ be the set of vertices of odd degree in the subgraph (V, F^-). Then, find a minimum-weight T^--join J^- with respect to weights $|b_f|$. It is straightforward to see that $F^- \triangle J^-$ is a collection $\mathcal{C}$ of cycles with minimum total benefit.

Therefore, we conclude that Algorithm 1 can be implemented to run in $O(n^3)$ time.

Theorem 3.2. *Algorithm 1 runs in $O(n^3)$ time.*

For the details of the implementations and the practical performance of the above algorithms, we refer the readers to, *e.g.*, [10, 17]. We remark that the above complexity bounds do not involve large hidden constants and are often pessimistic: the algorithms are much faster for practical instances. For example, for instances on dense graphs with 1,000 vertices, all of the procedures can be done within one minute.

3.3. The case where the optimal solution is found

We now prove that Algorithm 1 finds an optimal solution if the integrated benefits obey the following assumption.

Assumption 3.3. *For all edges $e \in E$, it holds that*

$$b_e^{odd} \geq b_e^{even}, \tag{3.1}$$

$$b_e^{odd} + b_e^{even} \geq 0. \tag{3.2}$$

Observe that Assumption 3.3 is satisfied if $\bar{b}_e(t)$ is monotone non-increasing for each $e \in E$, as stated in Section 1.3.

Theorem 3.4. *Under Assumption 3.3, Algorithm 1 finds an optimal solution, i.e., an Eulerian edge set $F \subseteq E' \cup E''$ maximizing the benefit $\sum_{f \in F} b_f$.*

Proof. We first show that, under Assumption 3.3, there exists an optimal solution containing at least one of e' or e'' for each $e \in E$. Suppose to the contrary that an optimal solution $F^* \subseteq E' \cup E''$ neither contains e' nor e'' for an edge $e \in E$. Then, for all of such an edge e, add both e' and e'' to F^*. It follows from (3.2) in Assumption 3.3 that this transformation does not decrease the objective function value $\sum_{f \in F^*} b_f$, while maintaining feasibility. Therefore, the transformed F^* is also an optimal solution.

Without loss of generality, we can assume that this F^* contains e' for every edge $e \in E$. If not, it follows that $e' \notin F^*$ and $e'' \in F^*$ for some edge $e \in E$, and then we can just remove e'' from F^* and add e' to F^*. From (3.1) in Assumption 3.3, this operation does not decrease the objective function value, and thus F^* is still an optimal solution.

We now complete the proof by showing that Algorithm 1 finds an Eulerian edge set $F \subseteq E' \cup E''$ maximizing the benefit $\sum_{f \in F} b_f$ among those edge sets containing e' for every edge $e \in E$.

For edge sets $F \subseteq E' \cup E''$ and $J_0 \subseteq E$, let $F \oplus J_0$ denote the subset of $E' \cup E''$ obtained from F and J_0 by (2.5), where J is replaced by J_0. For $F \subseteq E' \cup E''$, let $b(F)$ denote $\sum_{f \in F} b_f$. Similarly, for $J_0 \subseteq E$, let $w(J_0) = \sum_{e \in J_0} w_e$.

Denote the intermediate solution in Line 5 by $F^{(5)} \subseteq E' \cup E''$. Let $\tilde{J} \subseteq E$ an edge subset satisfying that

$$F^{(5)} \oplus \tilde{J} = F^*,$$

or explicitly,

$$\tilde{J} = \left\{ e \in E \mid e'' \in F^{(5)} \triangle F^* \right\}.$$

Recall that $T \subseteq V$ is the set of vertices with odd degree in $(V, F^{(5)})$ and F^* is an Eulerian edge set. It then follows that $\tilde{J}$ is a T-join. Since J is a maximum-weight T-join with respect to w, it holds that

$$w(\tilde{J}) \leq w(J). \tag{3.3}$$

Also, since F^* is an Eulerian edge subset of $E' \cup E''$ maximizing the total net benefit, it follows that

$$b(F^*) = b(F^{(5)} \oplus \tilde{J}) \geq b(F^{(5)} \oplus J). \tag{3.4}$$

Here, it is derived from the definition (2.4) of w that

$$b(F^{(5)} \oplus \tilde{J}) = b(F^{(5)}) + w(\tilde{J}), \quad b(F^{(5)} \oplus J) = b(F^{(5)}) + w(J),$$

and hence

$$b(F^{(5)} \oplus \tilde{J}) \leq b(F^{(5)} \oplus J) \tag{3.5}$$

by (3.3). Therefore, we conclude from (3.4) and (3.5) that

$$b(F^{(5)} \oplus J) = b(F^{(5)} \oplus \tilde{J}) = b(F^*),$$

and thus the output $F^{(5)} \oplus J$ of Algorithm 1 is an optimal solution. □

4. Conclusion

We have presented an improved heuristic algorithm for MBCPP (Algorithm 1), putting emphasis on the improvements upon the previous heuristic algorithm [14] and the theoretical and practical efficiency compared to the exact branch-and-cut method [4]. We have also proved that our heuristic algorithm finds an optimal solution under a certain assumption (Assumption 3.3).

Directions of future research would include designing approximation algorithms with guaranteed approximation ratio for MBCPP, and extending our approach to MBCPP in digraphs [15] and the common generalization of MBCPP and the vehicle routing problem [18].

Acknowledgements. The authors are thankful to Yutaro Yamaguchi for insightful comments. They are also indebted to anonymous referees for helpful comments. The second author is partially supported by JSPS KAKENHI Grant Numbers JP16K16012, JP20K11699, Japan.

References

[1] J. Aráoz, E. Fernández and C. Zoltan, Privatized rural postman problems. *Comput. Oper. Res.* **33** (2006) 3432–3449.
[2] J. Aráoz, E. Fernández and O. Meza, Solving the prize-collecting rural postman problem. *Eur. J. Oper. Res.* **196** (2009) 886–896.
[3] N. Christofides, Worst-case analysis of a new heuristic for the travelling salesman problem, Management Sciences Research Report 388, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA (1976).
[4] A. Corberán, I. Plana, A.M. Rodríguez-Chía and J.M. Sanchis, A branch-and-cut algorithm for the maximum benefit Chinese postman problem. *Math. Program.* **141** (2013) 21–48.
[5] J. Edmonds, Maximum matching and a polyhedron with 0,1-vertices. *J. Res. Natl. Bur. Stand. Sect. B* **69** (1965) 125–130.
[6] J. Edmonds, The Chinese postman's problem. *Bull. Oper. Res. Soc. Am.* **13** (1965) 73.
[7] J. Edmonds and E.L. Johnson, Matching, Euler tours and the Chinese postman. *Math. Program.* **5** (1973) 88–124.
[8] H.N. Gabow, *Implementation of algorithms for maximum matching on nonbipartite graphs.* Ph.D. thesis, Department of Computer Science, Stanford University, Stanford, California (1973).
[9] M.-g. Guan, Graphic programming using odd or even points. *Acta Math. Sin.* **10** (1960) 263–266.
[10] B. Korte and J. Vygen, Combinatorial Optimization–Theory and Algorithms, 6th ed. Springer, Heidelberg (2018).
[11] E.L. Lawler, Combinatorial Optimization: Networks and Matroids, edited by Holt, Rinehart and Winston. New York (1976).
[12] J.K. Lenstra and A.H.G. Rinnooy Kan, On general routing problems. *Networks* **6** (1976) 593–597.
[13] C. Malandraki and M.S. Daskin, The maximum benefit Chinese postman problem and the maximum benefit traveling salesman problem. *Eur. J. Oper. Res.* **65** (1993) 218–234.
[14] W.L. Pearn and K.L. Wang, On the maximum benefit Chinese postman problem. *Omega* **31** (2003) 269–273.
[15] W.L. Pearn and W.C. Chiu, Approximate solutions for the maximum benefit Chinese postman problem. *Int. J. Syst. Sci.* **36** (2005) 815–822.
[16] A. Schrijver, Combinatorial Optimization–Polyhedra and Efficiency. Springer, Heidelberg (2003).
[17] R. Sedgewick, Algorithms in C, Part 5: Graph Algorithms, 3rd ed. Addison-Wesley (2002).
[18] A. Shafahi and A. Haghani, Generalized maximum benefit multiple Chinese postman problem. *Transp. Res. Part C Emerg. Technol.* **55** (2015) 261–272.