A NOTE ON K BEST SOLUTIONS TO THE CHINESE POSTMAN PROBLEM

YASUFUMI SARUWATARI * AND TOMOMI MATSUI †

Abstract. The K-*best problems* on combinatorial optimization problems, in which K best solutions are considered instead of an optimal solution under the same conditions, have widely been studied. In this paper, we consider the K-best problem on the famous Chinese postman problem and develop an algorithm that finds K best solutions. The time complexity of our algorithm is $O(S(n,m) + K(n+m+\log K + nT(n+m,m)))$ where $S(s,t)$ denotes the time complexity of an algorithm for ordinary Chinese postman problems and $T(s,t)$ denotes the time complexity of a post optimal algorithm for non-bipartite matching problems defined on a graph with s vertices and t edges.

Key words. Combinatorial Optimization, Chinese Postman Problem, K-best Problem, T-join Problem, Matching Theory, Graph Theory

AMS(MOS) subject classifications. 05C38, 05C45

1. Introduction. The Chinese postman problem is proposed by Mei-ko Kwan in [8] for the first time. The problem is interpreted as follows [10]. The postman delivers mail along a set of streets and he must traverse each street at least once, in either direction. He starts at the post office and must return to this starting point. The Chinese postman problem finds a tour which enables the postman to walk the shortest possible distance.

It is well-known that the above problem is reformulated as follows. Let G be a graph whose edges correspond to the streets in the city. For each edge, a (non-negative) length (we call a weight) of the street is associated. If an Eulerian graph arises from G by parallelizing some edges, then an Eulerian cycle of this graph yields a postman's tour of the original. Thus, the problem finds an Eulerian graph of minimum total weight which is obtained by replacing some edges in the graph G by a set of parallel ones.

Now we give a formal description of the problem. Let $G = (V,E)$ be an undirected connected graph without loops and parallel edges. Denote by $\boldsymbol{w} \in \mathcal{Q}_+^E$ a non-negative weight function, where $\mathcal{Q}_+$ is the set of non-negative rational numbers. Then the Chinese postman problem is formulated as:

$$\begin{array}{lll} \text{minimize} & \boldsymbol{wx} = \sum_{e \in E} \boldsymbol{w}(e)\boldsymbol{x}(e), & \\ \text{subject to} & \boldsymbol{x}(e) \geq 1, & \forall e \in E, \\ & \sum_{e \in \delta(v)} \boldsymbol{x}(e) \ \text{ is even,} & \forall v \in V, \\ & \boldsymbol{x} \in \mathcal{Z}_+^E, & \end{array}$$

where $\mathcal{Z}_+$ denotes the set of non-negative integer numbers and $\delta(v)$ denotes the set of edges incident with the vertex v. The variable $\boldsymbol{x}(e)$ denotes the number of times the edge e is traversed in the postman's tour. The Chinese postman problem is a well-solved problem [8] and actually Edmonds presented a polynomial time algorithm by transforming the problem into a non-bipartite matching problem [5, 6].

* Department of Management Science, Science University of Tokyo, 1-3 Kagurazaka, Shinjuku-ku, Tokyo 162, Japan. saru@ms.kagu.sut.ac.jp or saruwata@gssm.otsuka.tsukuba.ac.jp

† Department of Industrial Administration, Science University of Tokyo, 2641 Yamazaki, Noda-shi, Chiba 278, Japan. tomomi@misojiro.t.u-tokyo.ac.jp

Here we consider the K-best Chinese postman problem which finds K distinct feasible solutions $\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^K$ such that $\boldsymbol{w}\boldsymbol{x}^1 \leq \boldsymbol{w}\boldsymbol{x}^2 \leq \cdots \leq \boldsymbol{w}\boldsymbol{x}^K \leq \boldsymbol{w}\boldsymbol{x}'$ for any feasible solution $\boldsymbol{x}' \neq \boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^K$. In this paper, the solutions $\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^K$ are called K *best solutions*. The K-best problem was first introduced by Murty and he developed an algorithm for finding K best solutions of the assignment problem [12]. In 1972, Lawler generalized the Murty's algorithm for finding K best solutions of general 0-1 integer problems [9]. However, it is hard to extend the Lawler's algorithm for the Chinese postman problem, since the variables are not 0-1 valued in this problem.

In Section 2, we introduce some properties of a solution for the 2-best Chinese postman problem instead of considering K best solutions directly. In Section 3, we develop a polynomial time algorithm for the 2-best Chinese postman problem. In Section 4, we construct an algorithm for K-best Chinese postman problems as an extension, which finds K best solutions by solving 2-best Chinese postman problems iteratively.

2. Properties of a solution of the 2-best Chinese postman problem. In this section, we show a property of K best solutions of the Chinese postman problem at first. The property induces an alternative formulation of the Chinese postman problem which is comfortable for solving the K-best Chinese postman problem.

Since the weight of each edge is non-negative, there exists an optimal postman's tour for the Chinese postman problem such that each edge is traversed at most twice. The following lemma is an extension of this property.

LEMMA 2.1. *The Chinese postman problem has* K *distinct feasible solutions* $\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^K$ *which satisfy the following two conditions:*
(1) $\boldsymbol{w}\boldsymbol{x}^1 \leq \boldsymbol{w}\boldsymbol{x}^2 \leq \cdots \leq \boldsymbol{w}\boldsymbol{x}^K \leq \boldsymbol{w}\boldsymbol{x}'$ *for any feasible solution* $\boldsymbol{x}' \neq \boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^K$,
(2) $k = 1, 2, \ldots, K, \quad \boldsymbol{x}^k(e) \leq 2k$ *for all* $e \in E$.

Proof. Let $(\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^{k-1})$ be a sequence of $k-1$ feasible solutions satisfying the conditions (1) and (2). Since the edge weights $\boldsymbol{w}(e)$ are non-negative, there exists a feasible solution $\boldsymbol{x}^k$ such that the sequence $(\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^k)$ satisfies the condition (1). Now consider the case that the sequence $(\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^k)$ violates the condition (2). Let $e' \in E$ be an edge with $\boldsymbol{x}^k(e') > 2k$. Since $\boldsymbol{x}^k$ is feasible to the Chinese postman problem, it is clear that the solution:

$$\boldsymbol{x}'(e) = \begin{cases} \boldsymbol{x}^k(e) & \text{if } e \neq e', \\ \boldsymbol{x}^k(e) - 2 & \text{if } e = e', \end{cases}$$

is also feasible and $\boldsymbol{x}'(e') \geq 2k - 1 > 2(k-1)$. Thus the solution $\boldsymbol{x}'$ is distinct from the solutions $\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^{k-1}$. From the assumption that $\boldsymbol{w}$ is non-negative, the solutions $\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^{k-1}, \boldsymbol{x}'$ are k best solutions. By applying this procedure iteratively, we can construct a sequence of K feasible solutions satisfying the conditions (1) and (2). □

The above lemma shows that when we solve K-best Chinese postman problems, it is sufficient to consider the set of finite number of feasible solutions satisfying $\boldsymbol{x}(e) \leq 2K$ for all $e \in E$.

In the rest of this paper, we consider the following problem instead of the problem described in the previous section, for simplicity of the notations. We call the following problem **CPP** in this paper:

$$\begin{array}{llll} (\mathbf{CPP}): & \text{minimize} & \boldsymbol{wx} = \sum_{e\in E} \boldsymbol{w}(e)\boldsymbol{x}(e), & \\ & \text{subject to} & \boldsymbol{b} \geq \boldsymbol{x} \geq \boldsymbol{a}, & \\ & & \sum_{e\in\delta(v)} \boldsymbol{x}(e) \ \text{ is even}, & \forall v \in V_1, \\ & & \sum_{e\in\delta(v)} \boldsymbol{x}(e) \ \text{ is odd}, & \forall v \in V \setminus V_1, \\ & & \boldsymbol{x} \in \mathcal{Z}_+^E, & \end{array}$$

where $\boldsymbol{w} \in \mathcal{Q}_+^E$ is a non-negative weight function, $\boldsymbol{a}$ and $\boldsymbol{b}$ are non-negative integer vectors in $\mathcal{Z}_+^E$ and V_1 denotes a subset of vertices. Clearly, an optimal solution of the ordinary Chinese postman problem is obtained by setting $\boldsymbol{a} = (1, 1, \ldots, 1)^T$, $\boldsymbol{b} = (2, 2, \ldots, 2)^T$, and $V_1 = V$. When we need K best solutions of the ordinary Chinese postman problem, it is sufficient to replace $\boldsymbol{b}$ by $(2K, 2K, \ldots, 2K)^T$. As easily seen that in the case $\boldsymbol{a} \leq \boldsymbol{b} \leq \boldsymbol{a}+\mathbf{1}$, this problem becomes the T-join problem [6, 11].

Similary to the ordinary Chinese postman problem, an optimal solution of **CPP** has the following property.

Claim 2.2. *If* **CPP** *has a feasible solution, then it has an optimal solution* $\boldsymbol{x}^*$ *such that, for all edge* $e \in E$, $\boldsymbol{x}^*(e)$ *is either* $\boldsymbol{a}(e)$ *or* $\boldsymbol{a}(e) + 1$.

Proof. From the assumption that $\boldsymbol{w}$ is non-negative, it is clear. □

When an integer vector $\boldsymbol{x} \in \mathcal{Z}_+^E$ is feasible to **CPP** and it satisfies the conditions that $\boldsymbol{x}(e)$ is either $\boldsymbol{a}(e)$ or $\boldsymbol{a}(e)+1$ for each $e \in E$, we say $\boldsymbol{x}$ is a *matching type solution of* **CPP**. Edmonds and Johnson [6] developed a polynomial time algorithm for the T-join problem and an optimal solution obtained by the algorithm is a matching type solution.

Given an optimal **CPP** solution $\boldsymbol{x}^*$, a feasible solution $\boldsymbol{x}^{\text{2nd}}$ of **CPP** is called a *second best solution* of **CPP** with respect to $\boldsymbol{x}^*$ if $\boldsymbol{wx}^* \leq \boldsymbol{wx}^{\text{2nd}} \leq \boldsymbol{wx}'$ holds for all feasible solutions $\boldsymbol{x}' \neq \boldsymbol{x}^*$. In the rest of this section, we show some properties of a second best solution of **CPP**.

For a given edge $e \in E$, let $\boldsymbol{u}^e$ be the 0-1 valued vector indexed by E such that

$$\boldsymbol{u}^e(e') = \begin{cases} 0 & \text{if } \ e' \neq e, \\ 1 & \text{if } \ e' = e. \end{cases}$$

Here we have two lemmas below.

Lemma 2.3. *Let* $\boldsymbol{x}^*$ *be a matching type optimal solution of* **CPP**. *If a second best solution* $\boldsymbol{x}^{\text{2nd}}$ *of* **CPP** *w.r.t.* $\boldsymbol{x}^*$ *exists and satisfies* $\boldsymbol{x}^{\text{2nd}}(e) \geq \boldsymbol{x}^*(e) + 2$ *for some edge* $e \in E$, *then* $\boldsymbol{x}' = \boldsymbol{x}^* + 2\boldsymbol{u}^e$ *is also a second best solution of* **CPP** *(w.r.t.* $\boldsymbol{x}^*$ *).*

Proof. Since $\boldsymbol{b}(e') \geq \max\{\boldsymbol{x}^{\text{2nd}}(e'), \boldsymbol{x}^*(e')\} \geq \boldsymbol{x}'(e') \geq \boldsymbol{x}^*(e') \geq \boldsymbol{a}(e')$ for every $e' \in E$, $\boldsymbol{x}'$ is a feasible solution of **CPP**.

Since $\boldsymbol{wx}' \geq \boldsymbol{wx}^{\text{2nd}}$ is obvious, it is sufficient to show the reverse inequality. By the definition of $\boldsymbol{x}^{\text{2nd}}$, it is clear that $\boldsymbol{x}^{\text{2nd}} - 2\boldsymbol{u}^e$ is feasible to **CPP**. Then, it follows that $\boldsymbol{wx}^{\text{2nd}} = \boldsymbol{w}(\boldsymbol{x}^{\text{2nd}} - 2\boldsymbol{u}^e) + 2\boldsymbol{wu}^e \geq \boldsymbol{wx}^* + 2\boldsymbol{wu}^e = \boldsymbol{w}(\boldsymbol{x}^* + 2\boldsymbol{u}^e) = \boldsymbol{wx}'$. □

Lemma 2.4. *Let* $\boldsymbol{x}^*$ *be a matching type optimal solution of* **CPP**. *Assume that there is a second best solution* $\boldsymbol{x}^{\text{2nd}}$ *w.r.t.* $\boldsymbol{x}^*$ *satisfying for any* $e \in E$,

$\boldsymbol{x}^{2\mathrm{nd}}(e) < \boldsymbol{x}^*(e)+2$. *Then there also exists a second best solution* $\boldsymbol{x}'$ *that is matching type.*

Proof. If $\boldsymbol{x}^{2\mathrm{nd}}$ is matching type, there is nothing to prove. Suppose that there exists an edge e with $\boldsymbol{b}(e) \geq \boldsymbol{x}^{2\mathrm{nd}}(e) > \boldsymbol{a}(e)+1$. Since $\boldsymbol{x}^{2\mathrm{nd}}(e) < \boldsymbol{x}^*(e)+2$ and $\boldsymbol{x}^*(e) \leq \boldsymbol{a}(e)+1$, $\boldsymbol{x}^{2\mathrm{nd}}(e) \leq \boldsymbol{a}(e)+2$ holds, thus it implies $\boldsymbol{x}^{2\mathrm{nd}}(e) = \boldsymbol{a}(e)+2$.

Here we consider the case $\boldsymbol{x}^*(e) = \boldsymbol{a}(e)$. Then we have $\boldsymbol{x}^{2\mathrm{nd}}(e) = \boldsymbol{a}(e)+2 = \boldsymbol{x}^*(e)+2$ and it contradicts to $\boldsymbol{x}^{2\mathrm{nd}}(e) < \boldsymbol{x}^*(e)+2$.

Now we just have the case $\boldsymbol{x}^*(e) = \boldsymbol{a}(e)+1$. Let $\boldsymbol{x}' = \boldsymbol{x}^{2\mathrm{nd}} - 2\boldsymbol{u}^e$. With respect to $\boldsymbol{x}'$, the parity of the degree of each vertex is the same as $\boldsymbol{x}^{2\mathrm{nd}}$ and $\boldsymbol{b} \geq \boldsymbol{x}' \geq \boldsymbol{a}$ holds by $\boldsymbol{x}'(e) = \boldsymbol{a}(e)$. It implies that $\boldsymbol{x}'$ is feasible to **CPP**. Clearly, $\boldsymbol{x}^* \neq \boldsymbol{x}'$ since $\boldsymbol{x}^*(e) = \boldsymbol{a}(e)+1 \neq \boldsymbol{a}(e) = \boldsymbol{x}'(e)$. By the definitions of $\boldsymbol{x}'$ and $\boldsymbol{x}^{2\mathrm{nd}}$, $\boldsymbol{w}\boldsymbol{x}' = \boldsymbol{w}\boldsymbol{x}^{2\mathrm{nd}}$. In this way, we can decrease the number of edges satisfying $\boldsymbol{x}^{2\mathrm{nd}}(e) > \boldsymbol{a}(e)+1$ from $\boldsymbol{x}^{2\mathrm{nd}}$ and in the sequel, a matching type second best solution is obtained. □

Summarizing Lemmas above, we have the following theorem, which shows the existence of a second best solution possessing the properties in Lemma 3 or 4.

THEOREM 2.5. *Assume that there exist an optimal and a second best solutions of* **CPP**. *Let* $\boldsymbol{x}^*$ *be a matching type optimal solution of* **CPP**. *Then, there always exists a second best solution* $\boldsymbol{x}^{2\mathrm{nd}}$ *w.r.t.* $\boldsymbol{x}^*$ *such that either* $\boldsymbol{x}^{2\mathrm{nd}}$ *is matching type or* $\boldsymbol{x}^{2\mathrm{nd}} = \boldsymbol{x}^* + 2\boldsymbol{u}^e$ *for an edge* $e \in E$.

Given a matching type optimal solution $\boldsymbol{x}^*$ of **CPP**, we call a second best solution $\boldsymbol{x}^{2\mathrm{nd}} = \boldsymbol{x}^* + 2\boldsymbol{u}^e$ for an edge $e \in E$ a *non-matching type second best solution* (w.r.t. $\boldsymbol{x}^*$).

Now we show a simple property of a matching type second best solution of **CPP**. It plays an important role in our algorithm when a matching type second best solution of **CPP** exists. Let $\boldsymbol{x}^*$ be a matching type optimal solution of **CPP** and $\boldsymbol{x}^{2\mathrm{nd}}$ a matching type second best solution w.r.t. $\boldsymbol{x}^*$. Then it is clear that $-1 \leq \boldsymbol{x}^{2\mathrm{nd}}(e) - \boldsymbol{x}^*(e) \leq 1$ for any $e \in E$. Denote by $G(\boldsymbol{x}^*, \boldsymbol{x}^{2\mathrm{nd}})$ a graph induced by the edge subset $\{e \in E \mid \boldsymbol{x}^{2\mathrm{nd}}(e) - \boldsymbol{x}^*(e) \neq 0\}$. For each vertex $v \in V$, $|\sum_{e\in\delta(v)} (\boldsymbol{x}^{2\mathrm{nd}}(e) - \boldsymbol{x}^*(e))|$ is even, and it implies $\sum_{e\in\delta(v)} |\boldsymbol{x}^{2\mathrm{nd}}(e) - \boldsymbol{x}^*(e)|$ is also even. Therefore the graph $G(\boldsymbol{x}^*, \boldsymbol{x}^{2\mathrm{nd}})$ satisfies that the number of edges incident with each vertex is even, *i.e.,* it is Eulerian.

Here we have the following lemma.

LEMMA 2.6. *Let* $\boldsymbol{x}^*$ *be a matching type optimal solution of* **CPP**. *Assume that there exists a matching type second best solution w.r.t.* $\boldsymbol{x}^*$. *Then we can construct a matching type second best solution* $\boldsymbol{x}'$ *such that the graph* $G(\boldsymbol{x}^*, \boldsymbol{x}')$ *consists of a single elementary cycle.*

Proof. From the assumption, there exists a matching type second best solution $\boldsymbol{x}^{2\mathrm{nd}}$ w.r.t. $\boldsymbol{x}^*$. The case that $G(\boldsymbol{x}^*, \boldsymbol{x}^{2\mathrm{nd}})$ consists of exactly one elementary cycle is trivial. If not, the graph $G(\boldsymbol{x}^*, \boldsymbol{x}^{2\mathrm{nd}})$ contains at least two elementary cycles since $G(\boldsymbol{x}^*, \boldsymbol{x}^{2\mathrm{nd}})$ is Eulerian. Let $C \subseteq E$ be one of such cycles. Let

$$\boldsymbol{d}(e) = \begin{cases} 1 & \text{if } e \in C \text{ and } \boldsymbol{x}^*(e) = \boldsymbol{a}(e), \\ -1 & \text{if } e \in C \text{ and } \boldsymbol{x}^*(e) = \boldsymbol{a}(e)+1, \\ 0 & \text{if } e \notin C, \end{cases}$$

and $\boldsymbol{x}' = \boldsymbol{x}^* + \boldsymbol{d}$. From the definitions of $G(\boldsymbol{x}^*, \boldsymbol{x}^{2\mathrm{nd}})$ and $\boldsymbol{d}$, $\boldsymbol{x}'$ satisfies $\boldsymbol{a}(e) \leq \min\{\boldsymbol{x}^*(e), \boldsymbol{x}^{2\mathrm{nd}}(e)\} \leq \boldsymbol{x}'(e) \leq \max\{\boldsymbol{x}^*(e), \boldsymbol{x}^{2\mathrm{nd}}(e)\} \leq \boldsymbol{b}(e)$ for any $e \in E$. Then, it is clear that $\boldsymbol{x}'$ is also feasible to **CPP**. Obviously $\boldsymbol{w}\boldsymbol{d} \geq 0$ since if $\boldsymbol{w}\boldsymbol{d} < 0$ then $\boldsymbol{w}\boldsymbol{x}' = \boldsymbol{w}\boldsymbol{x}^* + \boldsymbol{w}\boldsymbol{d} < \boldsymbol{w}\boldsymbol{x}^*$ and it leads to a contradiction. If $\boldsymbol{w}\boldsymbol{d} = 0$, then $\boldsymbol{w}\boldsymbol{x}' = \boldsymbol{w}\boldsymbol{x}^*$ and we can choose $\boldsymbol{x}'$ as a second best solution. By the definition of $\boldsymbol{x}'$, it is clear that $G(\boldsymbol{x}^*, \boldsymbol{x}')$ consists of one elementary cycle. Now consider the case $\boldsymbol{w}\boldsymbol{d} > 0$. Let $\boldsymbol{x}'' = \boldsymbol{x}^{\text{2nd}} - \boldsymbol{d}$. Then $\boldsymbol{x}''$ is feasible to **CPP** and $\boldsymbol{w}\boldsymbol{x}'' = \boldsymbol{w}\boldsymbol{x}^{\text{2nd}} - \boldsymbol{w}\boldsymbol{d} < \boldsymbol{w}\boldsymbol{x}^{\text{2nd}}$. Since $\boldsymbol{x}'' \neq \boldsymbol{x}^*$, it is a contradiction. □

The above lemma leads an algorithm for finding a matching type second best solution, if it exists.

3. An algorithm for finding a second best solution of CPP. In this section, we describe an algorithm for finding a second best solution of **CPP**.

Given a matching type optimal solution $\boldsymbol{x}^*$, we define a weight function $\widetilde{\boldsymbol{w}}$ on E as:

$$\widetilde{\boldsymbol{w}}(e) = \begin{cases} \boldsymbol{w}(e) & \text{if} \quad \boldsymbol{x}^*(e) = \boldsymbol{a}(e), \\ -\boldsymbol{w}(e) & \text{if} \quad \boldsymbol{x}^*(e) = \boldsymbol{a}(e) + 1. \end{cases}$$

Denote by C any elementary cycle in G. For a matching type solution $\boldsymbol{x}$, let $\boldsymbol{x} \bigtriangleup C$ be the integer vector in $\mathcal{Z}_+^E$ such that:

$$\boldsymbol{x} \bigtriangleup C\ (e) = \begin{cases} \boldsymbol{x}(e) + 1 & \text{if} \quad e \in C \quad \text{and} \quad \boldsymbol{x}(e) = \boldsymbol{a}(e), \\ \boldsymbol{x}(e) - 1 & \text{if} \quad e \in C \quad \text{and} \quad \boldsymbol{x}(e) = \boldsymbol{a}(e) + 1, \\ \boldsymbol{x}(e) & \text{if} \quad e \notin C. \end{cases}$$

It is clear that $\boldsymbol{x} \bigtriangleup C$ is also a matching type. If a matching type second best solution exists, then according to Lemma 2.6, we can construct one by finding an elementary cycle $C \subseteq E' = \{e \in E \mid \boldsymbol{a}(e) + 1 \leq \boldsymbol{b}(e)\}$ that minimizes $\sum_{e \in C} \widetilde{\boldsymbol{w}}(e)$.
Let $\mathcal{C}$ be the set of elementary cycles in the graph $G' = (V, E')$. Now we denote by $P(G, \boldsymbol{x}^*)$ the problem:

$$\begin{aligned} P(G, \boldsymbol{x}^*): \quad & \text{minimize} \quad \sum_{e \in C} \widetilde{\boldsymbol{w}}(e), \\ & \text{subject to} \quad C \in \mathcal{C}. \end{aligned}$$

An elementary cycle C^* obtained by solving $P(G, \boldsymbol{x}^*)$ is called a *minimum elementary cycle,* then $\boldsymbol{x}^* \bigtriangleup C^*$ is a matching type second best solution of **CPP**. Here from Theorem 2.5, we can develop the following algorithm that finds a second best solution w.r.t. a matching type optimal solution $\boldsymbol{x}^*$.

The algorithm *2-best*

Inputs: Graph $G = (V, E)$, weight function $\boldsymbol{w}$, lower bound $\boldsymbol{a}$, upper bound $\boldsymbol{b}$, vertex subset V_1 and a matching type optimal **CPP** solution $\boldsymbol{x}^*$.

Output: A second best solution $\boldsymbol{x}^{\text{2nd}}$, if it exists; and else say "none exist".

Step 1. Define a weight function $\widetilde{\boldsymbol{w}}$ on E as:

$$\widetilde{\boldsymbol{w}}(e) = \begin{cases} \boldsymbol{w}(e) & \text{if} \quad \boldsymbol{x}^*(e) = \boldsymbol{a}(e), \\ -\boldsymbol{w}(e) & \text{if} \quad \boldsymbol{x}^*(e) = \boldsymbol{a}(e) + 1. \end{cases}$$

Step 2.1 Solve $P(G, \boldsymbol{x}^*)$ and obtain a minimum elementary cycle $C^* \subseteq E'$.
If a minimum elementary cycle exists, set $W_{C^*} = \sum_{e \in C^*} \widetilde{\boldsymbol{w}}(e)$; else, set $W_{C^*} = \infty$.

Step 2.2 Find an edge $e^* \in E'' = \{e' \in E \mid \boldsymbol{x}^*(e') + 2 \le \boldsymbol{b}(e')\}$ such that $\boldsymbol{w}(e^*) = \min_{e' \in E''} \boldsymbol{w}(e')$.

If $E'' \neq \emptyset$, set $W_{e^*} = 2\boldsymbol{w}(e^*)$; else, set $W_{e^*} = \infty$.

Step 3. In the case that $W_{C^*} = W_{e^*} = \infty$, then say "none exist" and stop.

If $W_{e^*} \le W_{C^*}$, then set $\boldsymbol{x}^{2\mathrm{nd}} = \boldsymbol{x}^* + 2\boldsymbol{u}^{e^*}$; otherwise, set $\boldsymbol{x}^{2\mathrm{nd}} = \boldsymbol{x}^* \triangle C^*$.

Output $\boldsymbol{x}^{2\mathrm{nd}}$ and stop.

To show the computational effort of the above algorithm, let $n = |V|$ and $m = |E|$. In Step 2, the problem $P(G, \boldsymbol{x}^*)$ can be solved in polynomial time since the problem finding a minimum elementary cycle on a graph without negative cycle is reduced to the minimum-cost perfect matching problem (for a detail, see Lawler [10] Section 6.2). In the above algorithm, we already have a minimum-cost perfect matching. Hence, it is sufficient to apply a post optimal algorithm for non-bipartite matching problems [1, 3, 4] in Step 2. The computational effort required in other steps is less than $O(n + m)$. The overall complexity of the above algorithm is $O(m + n + nT(n + m, m))$, where $T(s, t)$ denotes the time complexity of a post optimal algorithm for non-bipartite matching problems on a graph with s vertices and t edges.

4. An extension of the algorithm to K-best CPP. In this section, we develop an algorithm that finds K best solutions of **CPP**. Our algorithm is based upon the *binary partitioning method,* which is used in [2, 7] for solving some K-best problems. More precisely, we partition all the feasible solutions of the given **CPP** into two subsets iteratively. Such a partition is realized by constructing two **CPP**s.

For the convenience, the problem **CPP** with graph G, weight function $\boldsymbol{w}$, lower and upper bound $\boldsymbol{a}$, $\boldsymbol{b}$ and vertex subset V_1 is denoted by $\mathbf{CPP}(G, \boldsymbol{w}, \boldsymbol{a}, \boldsymbol{b}, V_1)$. We assume that an optimal solution $\boldsymbol{x}^*$ and a second best solution $\boldsymbol{x}^{2\mathrm{nd}}$ of $\mathbf{CPP}(G, \boldsymbol{w}, \boldsymbol{a}, \boldsymbol{b}, V_1)$ are obtained.

Since $\boldsymbol{x}^* \neq \boldsymbol{x}^{2\mathrm{nd}}$, there exists an edge $e \in E$ such that $\boldsymbol{x}^*(e) \neq \boldsymbol{x}^{2\mathrm{nd}}(e)$. With respect to the edge e, we define the two integer vectors $\boldsymbol{a}'$, $\boldsymbol{b}'$ indexed by E as:

$$\boldsymbol{a}'(e') = \begin{cases} \boldsymbol{a}(e'), & \text{if } e' \neq e, \\ \min\{\boldsymbol{x}^*(e), \boldsymbol{x}^{2\mathrm{nd}}(e)\} + 1, & \text{if } e' = e, \end{cases}$$

$$\boldsymbol{b}'(e') = \begin{cases} \boldsymbol{b}(e'), & \text{if } e' \neq e, \\ \min\{\boldsymbol{x}^*(e), \boldsymbol{x}^{2\mathrm{nd}}(e)\}, & \text{if } e' = e. \end{cases}$$

In our algorithm, two problems $\mathbf{CPP}(G, \boldsymbol{w}, \boldsymbol{a}, \boldsymbol{b}', V_1)$ and $\mathbf{CPP}(G, \boldsymbol{w}, \boldsymbol{a}', \boldsymbol{b}, V_1)$ are constructed and maintained. Then it is clear that each feasible solution of the original problem $\mathbf{CPP}(G, \boldsymbol{w}, \boldsymbol{a}, \boldsymbol{b}, V_1)$ is feasible to exactly one of two problems $\mathbf{CPP}(G, \boldsymbol{w}, \boldsymbol{a}, \boldsymbol{b}', V_1)$ and $\mathbf{CPP}(G, \boldsymbol{w}, \boldsymbol{a}', \boldsymbol{b}, V_1)$. In addition, when the solution $\boldsymbol{x}^*$ is feasible to one of these two problems, then $\boldsymbol{x}^{2\mathrm{nd}}$ is feasible to another one. Here we denote these two problems by $\mathbf{CPP}(G, \boldsymbol{w}, \boldsymbol{a}_1, \boldsymbol{b}_1, V_1)$ and $\mathbf{CPP}(G, \boldsymbol{w}, \boldsymbol{a}_2, \boldsymbol{b}_2, V_1)$, and we may assume that $\boldsymbol{x}^*$ is feasible to $\mathbf{CPP}(G, \boldsymbol{w}, \boldsymbol{a}_1, \boldsymbol{b}_1, V_1)$ and $\boldsymbol{x}^{2\mathrm{nd}}$ is feasible to $\mathbf{CPP}(G, \boldsymbol{w}, \boldsymbol{a}_2, \boldsymbol{b}_2, V_1)$ without loss of generality. From the definition of these two problems, it is obvious that $\boldsymbol{x}^*$ is a matching type optimal solution of $\mathbf{CPP}(G, \boldsymbol{w}, \boldsymbol{a}_1, \boldsymbol{b}_1, V_1)$ and $\boldsymbol{x}^{2\mathrm{nd}}$ is a matching type optimal solution of $\mathbf{CPP}(G, \boldsymbol{w}, \boldsymbol{a}_2, \boldsymbol{b}_2, V_1)$. Thus, the conditions of Theorem 2.5 are maintained and the algorithm 2-best finds second best solutions of these two problems respectively.

By using the *best first search rule,* the following algorithm is applied.

The algorithm *K-best*

Inputs: Graph $G = (V, E)$, weight function $\boldsymbol{w}$, lower bound $\boldsymbol{a}$, upper bound $\boldsymbol{b}$, vertex subset V_1 and positive integer K.

Outputs: Sequence of K distinct solutions $\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^K$ feasible to $\mathbf{CPP}(G, \boldsymbol{w}, \boldsymbol{a}, \boldsymbol{b}, V_1)$ such that $\boldsymbol{w}\boldsymbol{x}^1 \leq \boldsymbol{w}\boldsymbol{x}^2 \leq \cdots \leq \boldsymbol{w}\boldsymbol{x}^K \leq \boldsymbol{w}\boldsymbol{x}'$ for any feasible solution $\boldsymbol{x}' \neq \boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^K$, if it exists; else say "none exist".

Step 0. Solve $\mathbf{CPP}(G, \boldsymbol{w}, \boldsymbol{a}, \boldsymbol{b}, V_1)$ and find a matching type best solution $\boldsymbol{x}^*$.
Find a second best solution $\boldsymbol{x}^{\text{2nd}}$ of $\mathbf{CPP}(G, \boldsymbol{w}, \boldsymbol{a}, \boldsymbol{b}, V_1)$ w.r.t. $\boldsymbol{x}^*$.
Set $\mathcal{P} = \{(\mathbf{CPP}(G, \boldsymbol{w}, \boldsymbol{a}, \boldsymbol{b}, V_1), \boldsymbol{x}^*, \boldsymbol{x}^{\text{2nd}})\}$.
Output $\boldsymbol{x}^*$ as $\boldsymbol{x}^1$ (a best solution).
Set $k = 2$.

Step 1. If $k > K$, then stop. Else if $\mathcal{P} = \emptyset$, then say "none exist" and stop.

Step 2. Let $(\mathbf{CPP}(G, \boldsymbol{w}, \widetilde{\boldsymbol{a}}, \widetilde{\boldsymbol{b}}, V_1), \boldsymbol{x}^*, \boldsymbol{x}^{\text{2nd}})$ be an element of $\mathcal{P}$ such that;

$$\boldsymbol{w}\boldsymbol{x}^{\text{2nd}} = \min\{\boldsymbol{w}\boldsymbol{x}'' \mid (\mathbf{CPP}(G, \boldsymbol{w}, \boldsymbol{a}', \boldsymbol{b}', V_1), \boldsymbol{x}', \boldsymbol{x}'') \in \mathcal{P}\}.$$

Output $\boldsymbol{x}^{\text{2nd}}$ as $\boldsymbol{x}^k$ (a k-th best solution).
Delete $(\mathbf{CPP}(G, \boldsymbol{w}, \widetilde{\boldsymbol{a}}, \widetilde{\boldsymbol{b}}, V_1), \boldsymbol{x}^*, \boldsymbol{x}^{\text{2nd}})$ from $\mathcal{P}$.

Step 3. Construct two problems $\mathbf{CPP}(G, \boldsymbol{w}, \boldsymbol{a}_1, \boldsymbol{b}_1, V_1)$ and $\mathbf{CPP}(G, \boldsymbol{w}, \boldsymbol{a}_2, \boldsymbol{b}_2, V_1)$.

Step 4.1. Find a second best solution $\boldsymbol{x}'$ of $\mathbf{CPP}(G, \boldsymbol{w}, \boldsymbol{a}_1, \boldsymbol{b}_1, V_1)$.
If no second best solution exists, then go to Step 4.2.
Else, add $(\mathbf{CPP}(G, \boldsymbol{w}, \boldsymbol{a}_1, \boldsymbol{b}_1, V_1), \boldsymbol{x}^*, \boldsymbol{x}')$ to $\mathcal{P}$.

Step 4.2. Find a second best solution $\boldsymbol{x}''$ of $\mathbf{CPP}(G, \boldsymbol{w}, \boldsymbol{a}_2, \boldsymbol{b}_2, V_1)$.
If no second best solution exists, then go to Step 5.
Else add $(\mathbf{CPP}(G, \boldsymbol{w}, \boldsymbol{a}_2, \boldsymbol{b}_2, V_1), \boldsymbol{x}^{\text{2nd}}, \boldsymbol{x}'')$ to $\mathcal{P}$.

Step 5. Set $k = k + 1$, and go to Step 1.

Now we discuss the memory requirement and the time complexity of the above algorithm.

In each iteration, we delete one **CPP** from the set of problems $\mathcal{P}$ and add at most two **CPP**s to $\mathcal{P}$; *i.e.,* the number of problems in the set $\mathcal{P}$ increases at most 1. Hence, the memory requirement of the algorithm is less than $O(K(n + m))$.

By applying Edmonds' technique in [5], the ordinary Chinese postman problem is reduced to a non-bipartite matching problem and we can obtain a matching type optimal solution of $\mathbf{CPP}(G, \boldsymbol{w}, \boldsymbol{a}, \boldsymbol{b}, V_1)$ in polynomial time [1, 3, 4]. Here we denote the computational efforts required to obtain a matching type optimal solution in Step 1 by $S(n, m)$. In Section 3, we described an $O(m + n + nT(n + m, m))$ algorithm for solving a 2-best **CPP**, where $T(s, t)$ denotes the time complexity of a post optimal algorithm for non-bipartite matching problems defined on a graph with s vertices and t edges [1, 3, 4]. Since the number of problems in the set $\mathcal{P}$ is bounded by $O(K)$, we can find a triplet $(\mathbf{CPP}(G, \boldsymbol{w}, \widetilde{\boldsymbol{a}}, \widetilde{\boldsymbol{b}}, V_1), \boldsymbol{x}^*, \boldsymbol{x}^{\text{2nd}})$ and delete it from $\mathcal{P}$ in Step 2 in $O(n + m + \log K)$ time and two triplets are added in Step 4.1 and 4.2 with $O(n + m + \log K)$ computational efforts, by using a comfortable data structure. The above algorithm outputs one solution and solves two 2-best **CPP**s in each iteration. Thus overall time complexity is $O(S(n, m) + K(n + m + \log K + nT(n + m, m)))$.

5. Conclusion. In this paper, we treat the 2-best Chinese postman problem that finds a second best solution of the problem. We also consider the K best solutions of the problem as an extension of the 2-best problem. We developed an algorithm to solve the problem.

REFERENCES

[1] M. Ball and U. Derigs, *An analysis of alternative strategies for implementing matching algorithms*, Networks, 13 (1983), pp. 517–549.

[2] C. R. Chegireddy and H. W. Hamacher, *Algorithms for finding k-best perfect matchings*, Discrete Applied Math., 18 (1987), pp. 155–165.

[3] W. Cunningham and A. Marsh, *A primal algorithm for optimum matching*, Mathematical Programming Study, 8 (1983), pp. 517–549.

[4] U. Derigs, *A shortest augmenting path method for solving minimal perfect matching problems*, Networks, 11 (1981), pp. 379–390.

[5] J. Edmonds, *Path, trees, and flowers*, Canadian J. Math., 17 (1965), pp. 449–467.

[6] ———, *Matching, Euler tour and the chinese postman*, Mathematical Programming, 5 (1973), pp. 88–124.

[7] H. Hamacher and M. Queyranne, *k-best solutions to combinatorial optimization problems*, Annals of Operations Research, 4 (1985/6), pp. 123–143. Research Report No 83-5 Industrial and Systems Engineering Department, University of Florida, (1981).

[8] M. ko Kwan, *Graphic programming using odd or even points*, Chinese Mathematics, 1 (1962), pp. 237–277.

[9] E. L. Lawler, *A procedure for computing the k-th best solutions to discrete optimization problems and its application to the shortest path problem*, Management Science, 18 (1972), pp. 401–405.

[10] ———, *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart and Winston, New York, 1976.

[11] L. Lovász and M. D. Plummer, *Matching Theory*, North-Holland, 1986.

[12] K. G. Murty, *An algorithm for ranking all the assignments in order of increasing cost*, Operations Research, 16 (1968), pp. 682–687.