Communications in Combinatorics and Optimization
Vol. ?? No. ??, ?? pp.1-12
DOI: 10.22049/CCO.2018.26231.1087

CCO
Commun. Comb. Optim.

On the inverse maximum perfect matching problem under the bottleneck-type Hamming distance

Javad Tayyebi

Department of Industrial Engineering, Birjand University of Technology, Birjand, Iran
javadtayyebi@birjandut.ac.ir & javadtayyebi@birjand.ac.ir

Received: 18 May 2018; Accepted: 30 October 2018
Published Online: 1 November 2018

Communicated by ***Stephan Wagner***

Abstract: Given an undirected network $G(V, A, \mathbf{c})$ and a perfect matching M of G, the inverse maximum perfect matching problem consists of modifying minimally the elements of $\mathbf{c}$ so that M becomes a maximum perfect matching with respect to the modified vector. In this article, we consider the inverse problem when the modifications are measured by the weighted bottleneck-type Hamming distance. We propose an algorithm based on the binary search technique for solving the problem. Our proposed algorithm has a better time complexity than the one presented in [13]. We also study the inverse assignment problem as a special case of the inverse maximum perfect matching problem in which the network is bipartite and present an efficient algorithm for solving the problem. Finally, we compare the algorithm with those presented in the literature.

Keywords: Inverse problem, Hamming distance, perfect matching, binary search

AMS Subject classification: 05C70, 05C85

1. Introduction

Suppose $G(V, A, \mathbf{c})$ is an undirected network where $V = \{1, 2, \ldots, n\}$ is the node set, A is the set of m arcs and $\mathbf{c}$ is the profit vector for arcs. A matching M is a subset of A so that no two arcs in M share a common node, i.e., for each two arcs $(i, j), (k, l) \in M$, we have $i \notin \{k, l\}$ and $j \notin \{k, l\}$. A perfect matching is a matching which matches all nodes of the network. Clearly, a necessary condition for the existence of a perfect matching is that the number of nodes is even. Thus, we assume that n is an even positive integer. The well-known maximum perfect matching problem is to find a perfect matching M so that its total profit (i.e., $\sum_{(i,j)\in M} c_{ij}$) is maximized. There exist several strongly polynomial-time algorithms for solving the maximum perfect

© 2018 Azarbaijan Shahid Madani University. All rights reserved.

matching problem [7, 10].
For a maximum perfect matching problem defined on $G(V, A, \mathbf{c})$ and a perfect matching M of G, the inverse maximum perfect matching problem is to modify the profit vector $\mathbf{c}$ into $\hat{\mathbf{c}}$ minimally so that M becomes a maximum perfect matching. Therefore, the inverse problem can be formulated as follows [13]:

$$\begin{aligned} &\min d(\mathbf{c}, \hat{\mathbf{c}}), \\ &M \text{ is a maximum perfect matching in } G(V, A, \hat{\mathbf{c}}), \\ &l_{ij} \leq \hat{c}_{ij} - c_{ij} \leq u_{ij} \quad \forall (i,j) \in A, \end{aligned} \tag{1}$$

where $\hat{\mathbf{c}}$ is to be determined, l_{ij} and $u_{ij}, (i,j) \in A$, are the given bounds for decreasing and increasing c_{ij}, respectively and $d(.,.)$ is a distance function which measures the modifications.
Liu and Zhang [14] studied the problem (1) when the modifications are measured by the l_1 and l_∞ norms. They presented strongly polynomial-time algorithms for solving the problem. Demange and Monnot [5] provided an introduction to inverse combinatorial optimization under the l_1 norm. As a special case, they considered the problem (1) with the additional constraint that each component of the vector $\hat{\mathbf{c}}$ belongs to a discrete finite set. They showed that this problem is NP-hard. They also showed the problem is polynomially solvable when the discrete sets are restricted to $\{0, 1\}$ and the network is bipartite.
Over the past decade, inverse optimization problems under the Hamming distances received attention (readers are referred to [2, 8, 9, 11, 12, 16, 17] and papers cited therein). The inverse maximum perfect matching problems under the Hamming distances have been considered in [13]. The authors studied both the sum-type and the bottleneck type cases. In the sum-type case, it is shown that the problem is NP-hard. In the bottleneck-type case, an $O(mn^3)$ algorithm is proposed to solve the problem.
In this article, we consider the inverse maximum perfect matching problem under the weighted bottleneck-type Hamming distance. We present an algorithm based on the binary search approach for solving the problem. The proposed algorithm runs in $O(n^3 \log n)$ time which has better time complexity than one presented in [13].
The binary search approach has been used for solving some inverse optimization problems under the l_∞ norm and the bottleneck-type Hamming distance. For instance, Liu and Zhang [14] presented a polynomial-time algorithm by using the binary search approach to solve the inverse maximum perfect matching problem under the l_∞ norm. Inverse multi-objective combinatorial optimization problem under the l_∞ norm has been considered in [15]. The authors used the binary search approach to present a method for solving the problem. Duin and Volgenant [6] also applied this approach for solving some inverse combinatorial problems under the bottleneck-type Hamming distance such as the inverse shortest path problem, inverse maximum spanning tree problem and the inverse assignment problem. They presented three approaches for solving the inverse assignment problem which run in $O(n^2 \log n)$, $O(n^3 \log n)$ and $O(mn^2)$ time, respectively.

As a special case of the inverse maximum perfect matching problem, we also consider the inverse assignment problem and present an efficient algorithm for solving this problem. Finally, we compare the algorithm with those presented in [6].
The rest of this article is organized as follows: In Section 2, the inverse perfect matching problem is studied and an algorithm based on the binary search approach is proposed. In Section 3, the inverse assignment problem is considered. Finally, some concluding remarks are provided in Section 4.

2. Inverse perfect matching problem

In this section, we consider the inverse maximum perfect matching problem when the modifications are measured by the weighted bottleneck-type Hamming distance. Suppose that we incur a penalty $w_{ij} \geq 0$ for modifying c_{ij}, $(i,j) \in A$. Thus, the problem can be formulated as follows [13]:

$$\begin{aligned} &\min \max_{(i,j)\in A} w_{ij} H(c_{ij}, \hat{c}_{ij}), \\ &M \text{ is a maximum perfect matching in } G(V, A, \hat{\mathbf{c}}), \\ &l_{ij} \leq \hat{c}_{ij} - c_{ij} \leq u_{ij} \quad \forall (i,j) \in A, \end{aligned} \tag{2}$$

where $H(c_{ij}, \hat{c}_{ij})$ is the Hamming distance between c_{ij} and $\hat{c}_{ij}$, i.e., $H(c_{ij}, \hat{c}_{ij}) = 1$ if $c_{ij} \neq \hat{c}_{ij}$ and $H(c_{ij}, \hat{c}_{ij}) = 0$ otherwise. The problem parameters are defined as in the problem (1).

Definition 1. *For an undirected network $G(V, A, \mathbf{c})$, a cycle C is called an alternating cycle with respect to a prefect matching M if the arcs around C alternately belong to M and $A \backslash M$. For an alternating cycle C, its cost is defined as $\mathbf{c}(C) = \sum_{(i,j)\in C\cap M} c_{ij} - \sum_{(i,j)\in C\backslash M} c_{ij}$.*

The following lemma due to Berge [4] states the optimality condition of a perfect matching M for the maximum perfect matching problem.

Lemma 1. *A perfect matching M of $G(V, A, \mathbf{c})$ is a maximum perfect matching if each alternating cycle with respect to M has a nonnegative cost.*

Note that each objective value of the problem (2) belongs to the set $W = \{w_{ij} : (i,j) \in A\} \cup \{0\}$ because if the initial profit vector $\mathbf{c}$ is feasible to the problem (2), then its objective value is 0. Suppose that the elements of W are arranged in nondecreasing order: let $w_0 = 0 \leq w_1 \leq w_2 \leq \ldots \leq w_m$ be the sorted list of penalties. For $k \in \{0, 1, 2, \ldots, m\}$, we introduce the solution $\hat{\mathbf{c}}^{(k)}$ as

$$\hat{c}_{ij}^{(k)} = \begin{cases} c_{ij} + u_{ij} & (i,j) \in A_k \cap M, \\ c_{ij} - l_{ij} & (i,j) \in A_k \backslash M, \\ c_{ij} & (i,j) \in A \backslash A_k, \end{cases} \qquad \forall (i,j) \in A, \tag{3}$$

where $A_k = \{(i,j) \in A : w_{ij} \leq w_k\}$. The following theorem is on the feasibility of $\hat{\mathbf{c}}^{(k)}$.

Theorem 1. *For $k \in \{0, 1, \ldots, m\}$, if the problem (2) has a feasible solution with objective value less than or equal to w_k, then $\hat{\mathbf{c}}^{(k)}$ defined by (3) is a feasible solution with the objective value w_k*

Proof. It is obvious that $\hat{\mathbf{c}}^{(k)}$ satisfies the bound constraints $l_{ij} \leq \hat{c}_{ij} - c_{ij} \leq u_{ij}$, $(i,j) \in A$.
Let $\hat{\mathbf{c}}$ be a feasible solution to the problem (1) with objective value less than or equal to w_k. By the definition (3) and the fact that the objective value of $\hat{\mathbf{c}}$ is less than or equal to w_k, we have

$$\begin{aligned}
\hat{c}_{ij}^{(k)} &= \hat{c}_{ij} = c_{ij} \quad (i,j) \in A \backslash A_k, \\
\hat{c}_{ij}^{(k)} &= c_{ij} + u_{ij} \geq \hat{c}_{ij} \quad (i,j) \in A_k \cap M, \\
\hat{c}_{ij}^{(k)} &= c_{ij} - l_{ij} \leq \hat{c}_{ij} \quad (i,j) \in A_k \backslash M.
\end{aligned}$$

and consequently,

$$\begin{aligned}
\hat{c}_{ij}^{(k)} &\geq \hat{c}_{ij} \quad (i,j) \in M, \\
\hat{c}_{ij}^{(k)} &\leq \hat{c}_{ij} \quad (i,j) \in A \backslash M.
\end{aligned} \tag{4}$$

Let C be an arbitrary alternating cycle with respect to M. Based on Lemma 1, $\hat{\mathbf{c}}(C) \geq 0$ due to the feasibility of $\hat{\mathbf{c}}$. By using (4),

$$\hat{\mathbf{c}}^{(k)}(C) = \sum_{(i,j) \in C \cap M} \hat{c}_{ij}^{(k)} - \sum_{(i,j) \in C \backslash M} \hat{c}_{ij}^{(k)} \geq \sum_{(i,j) \in C \cap M} \hat{c}_{ij} - \sum_{(i,j) \in C \backslash M} \hat{c}_{ij} = \hat{\mathbf{c}}(C) \geq 0.$$

Thus, $\hat{\mathbf{c}}^{(k)}$ is a feasible solution to the problem (2).
Since the modified components of $\hat{\mathbf{c}}^{(k)}$ belong to A_k, it follows that its objective value is w_k. This completes the proof. □

The following results are two immediate consequences of Theorem 1.

Corollary 1. *The problem (2) is feasible if and only if M is a maximum perfect matching of $G(V, A, \hat{\mathbf{c}}^{(m)})$ where $\hat{\mathbf{c}}^{(m)}$ is defined by (3) for $k = m$.*

Proof.
Sufficient condition. If M is a maximum perfect matching of $G(V, A, \hat{\mathbf{c}}^{(m)})$, then the solution $\hat{\mathbf{c}}^{(m)}$ is feasible to the problem (2) and consequently, the problem is feasible.
Necessary condition. Suppose that M is not a maximum perfect matching of $G(V, A, \hat{\mathbf{c}}^{(m)})$ by contradiction. Thus, the solution $\hat{\mathbf{c}}^{(m)}$ is not feasible to the problem

(2). Based on Theorem 1, there exists no solution with objective value less than or equal to w_m. This together with the fact that the greatest objective value of the problem is w_m show that the problem is infeasible. □

Corollary 2. *If $k \in \{0, 1, \ldots, m\}$ is the least index such that $\hat{\mathbf{c}}^{(k)}$ is feasible to the problem (2), then the solution $\hat{\mathbf{c}}^{(k)}$ is optimal.*

Proof. Let k be the least index such that $\hat{\mathbf{c}}^{(k)}$ is feasible to the problem (2). Thus, $\hat{\mathbf{c}}^{(k-1)}$ is not feasible. By using Theorem 1, there exists no feasible solution with objective value less than or equal to w_{k-1}. Therefore, the feasible solution $\hat{\mathbf{c}}^{(k)}$ is optimal. □

From Corollary 2, the problem (2) reduces to finding the least index k so that $\hat{\mathbf{c}}^{(k)}$ is a feasible solution to the problem. It is obvious that $\hat{\mathbf{c}}^{(k)}$, $k = 0, 1, \ldots, m$, satisfies the bound constraints $l_{ij} \leq \hat{c}_{ij} - c_{ij} \leq u_{ij}$, $(i, j) \in A$. Thus, the feasibility of $\hat{\mathbf{c}}^{(k)}$ is guaranteed if M is a maximum perfect matching of $G(V, A, \hat{\mathbf{c}}^{(k)})$. Our proposed algorithm applies the binary search approach to find the least index k so that $\hat{\mathbf{c}}^{(k)}$ is feasible. In each iteration, it uses the fact that $\hat{\mathbf{c}}^{(k)}$ is feasible if $\hat{\mathbf{c}}^{(k)}(M^*) = \hat{\mathbf{c}}^{(k)}(M)$ where M^* is a maximum perfect matching of $G(V, A, \hat{\mathbf{c}}^{(k)})$. Let us state formally our proposed algorithm.

Algorithm 1 to solve the problem (2) by the binary search approach.

Input: A network $G(V, A, \mathbf{c})$, a perfect matching M of G, a penalty vector $\mathbf{w}$ and bound vectors $\mathbf{l}$ and $\mathbf{u}$.
Output: An optimal solution $\hat{\mathbf{c}}^*$ with the objective value w^*.
Set $k = m$ and $s = m$.
Solve the maximum perfect matching problem on $G(V, A, \hat{\mathbf{c}}^{(k)})$. Let M^* be a maximum perfect matching.
if $\hat{\mathbf{c}}^{(k)}(M^*) \neq \hat{\mathbf{c}}^{(k)}(M)$ **then**
    The problem (2) is infeasible and stop (see Corollary 1).
end if
Set $\hat{\mathbf{c}}^* = \hat{\mathbf{c}}^{(k)}, w^* = w_k$, $s = [\frac{s}{2}], k = k - s$.
while $s \neq 0$ **do**
    Solve the maximum perfect matching problem on $G(V, A, \hat{\mathbf{c}}^{(k)})$. Let M^* be the optimal perfect matching.
    if $\hat{\mathbf{c}}^{(k)}(M^*) = \hat{\mathbf{c}}^{(k)}(M)$ **then**
        Update $\hat{\mathbf{c}}^* = \hat{\mathbf{c}}^{(k)}, w^* = w_k$, $s = [\frac{s}{2}], k = k - s$.
    else
        Update $s = [\frac{s}{2}], k = k + s$.
    end if
end while
if the problem (2) is feasible **then**
    $\hat{\mathbf{c}}^*$ is an optimal solution to it with the objective value w^*.
end if

$\hat{\mathbf{c}}^*$ stores the last feasible solution of the problem (2) found by Algorithm 1 and w^* is its objective value.
We now analyze the time complexity of the algorithm. It is obvious that the bottleneck

operation in each iteration is to solve a maximum perfect matching problem which can be done by a modified version of Edmonds' algorithm in $O(n^3)$ time [10]. Since Algorithm 1 uses the binary search approach to find an optimal index k in the range 0 to m, the number of iterations is $O(\log(m+1)) = O(\log n)$. Consequently, the time complexity of Algorithm 1 is $O(n^3 \log n)$. Thus, we have established the following result.

Theorem 2. *Algorithm 1 solves the problem (2) in $O(n^3 \log n)$ time.*

3. Inverse assignment problem

In this section, we consider the inverse assignment problem under the weighted bottleneck-type hamming distance. This problem is a special case of the problem (2) when $G(V, A, \mathbf{c})$ is a bipartite network. We present an efficient Algorithm to solve the inverse assignment problem. Finally, we compare the algorithm with those presented in [6].

The assignment problem is one of the fundamental combinatorial optimization problems. The problem can be described as follows: Suppose that $V_1 = \{1, 2, \ldots, n\}$ and $V_2 = \{1', 2', \ldots, n'\}$ are the sets of persons and tasks, respectively. Any person $i \in V_1$ can be assigned to perform any task $j' \in V_2$ incurring some cost $c_{ij'}$. An assignment is a pattern that assigns exactly one person to each task and exactly one task to each person. The assignment problem is to find an assignment in such a way that its total cost is minimized.

The assignment problem is a special case of the maximum perfect matching problem when the network is bipartite and its cost vector is the opposite of the profit vector of the maximum perfect matching problem. Note that each perfect matching M corresponds to exactly one assignment and vice versa. Hence, we use the words 'assignment' and 'perfect matching' interchangeably.

In the operations research literature, the assignment problem is also a special case of the minimum cost flow problem [1]. Thus, one can use the fundamental notions of network optimization for solving the assignment problem. Here, we consider the assignment problem as a minimum cost flow problem. Suppose that $G = (V, A, \mathbf{c})$ is a bipartite directed network where $V = V_1 \cup V_2$, $A = V_1 \times V_2$ and $c_{ij'}$ is the cost of sending a unit of flow on $(i, j') \in A$. Let each node $i \in V_1$ be a supply node with 1 unit of supply and each node $j' \in V_2$ a demand node with 1 unit of demand. The assignment problem is to send n unit of flow from supply nodes to demand nodes with minimum cost. Each perfect matching $M \in G$ corresponds to the zero-one feasible flow $\mathbf{x}$ defined by $x_{ij'} = 1$ if $(i, j') \in M$ and $x_{ij'} = 0$ else. The integrality property of optimal solutions in the minimum cost flow problems implies that the assignment problem is equivalent to this special case of the minimum cost flow problem [1].

Suppose that $\mathbf{x}$ is a feasible flow on $G(V, A, \mathbf{c})$. The residual network $G'(V, A', \mathbf{c}')$ with respect to $\mathbf{x}$ is constructed as follows:

Algorithm 2 Constructing the residual network

The node set is still V.
for $(i, j') \in A$ **do**
  if $x^0_{ij'} < 1$ **then**
    Add (i, j') to A' with $c'_{ij'} = c_{ij'}$.
  end if
  if $x^0_{ij'} > 0$ **then**
    Add (j', i) to A' with $c'_{j'i} = -c_{ij'}$.
  end if
end for

The following lemma states an optimality condition for a given feasible solution of the assignment problem.

Lemma 2. ***(Negative cycle optimality condition)****[1] A feasible flow* $\mathbf{x}^0$ *is optimal to the assignment problem if and only if the corresponding residual network does not contain any negative (cost) directed cycle.*

Now, we are ready to describe our algorithm. Since the inverse assignment problem is a special case of the inverse perfect matching problem, Algorithm 1 can be used directly to solve an instance of the inverse assignment problem. For using Algorithm 1, we must apply an algorithm as a subroutine for solving assignment problems in each iteration to test the optimality of the given assignment M. Instead of solving assignment problems, one can check directly whether or not the given assignment M is optimal by some optimality conditions to reduce the time complexity of Algorithm 1. Here, we suggest that the optimality of M is determined by the negative cycle optimality condition (see Lemma 2). This argument implies the following algorithm.

Algorithm 3 to solve the inverse assignment problem

Input: A bipartite network $G(V_1 \cup V_2, V_1 \times V_2, \mathbf{c})$, an assignment M of G, a penalty vector $\mathbf{w}$ and bound vectors $\mathbf{l}$ and $\mathbf{u}$.
Output: An optimal solution $\hat{\mathbf{c}}^*$ with the objective value w^*.
Set $k = m$ and $s = m$.
Construct the feasible flow $\mathbf{x}^0$ on G as

$$x_{ij}^0 = \begin{cases} 1 & (i,j) \in M, \\ 0 & (i,j) \notin M. \end{cases}$$

Construct the residual network of $G(V_1 \cup V_2, V_1 \times V_2, \hat{\mathbf{c}}^{(m)})$ with respect to $\mathbf{x}^0$ by Algorithm 2.
if the residual network contains at least one negative cycle **then**
  The inverse assignment problem is infeasible and stop.
end if
Set $\hat{\mathbf{c}}^* = \hat{\mathbf{c}}^{(k)}, w^* = w_k$, $s = [\frac{s}{2}], k = k - s$.
while $s \neq 0$ **do**
  Construct the residual network of $G(V_1 \cup V_2, V_1 \times V_2, \hat{\mathbf{c}}^{(k)})$ with respect to $\mathbf{x}^0$ by Algorithm 2.
  if the residual network contains no negative cycle **then**
    Update $\hat{\mathbf{c}}^* = \hat{\mathbf{c}}^{(k)}, w^* = w_k$, $s = [\frac{s}{2}], k = k - s$.
  else,
    Update $s = [\frac{s}{2}], k = k + s$.
  end if
end while
if the inverse assignment problem is feasible **then**
  $\hat{\mathbf{c}}^*$ is an optimal solution to the problem with the objective value w^*.
end if

We now analyze the time complexity of Algorithm 3. It is obvious that the bottleneck operation is to detect a negative cycle in the current residual network. Since the presence of a negative cycle can be identified by the FIFO label-correcting algorithm, it follows that each iteration can be done in $O(mn)$ time [1]. The number of iterations is $O(\log m) = O(\log n)$ because of the binary search approach. Thus, we have established the following result.

Theorem 3. *The inverse assignment problem can be solved in* $O(mn \log n)$.

3.1. Comparison of algorithm efficiency

Duin and Volgenant [6] studied the general inverse combinatorial optimization problem. As a special case, they considered the inverse assignment problem. They used the linear search approach together with the concepts of sensitivity analysis to present an $O(n^2 m)$ algorithm for solving the inverse assignment problem. They also proposed an $O(n^3 \log n)$ algorithm based on the binary search approach. Their second algorithm is similar to Algorithm 1 and solves an assignment problem in each iteration. Note that our proposed algorithm has a better time complexity than both their algorithms. To present a faster algorithm, Duin and Volgenant[6] introduced a method for computing the reduced costs of arcs with this property that the reduced cost of each arc corresponds to an alternating cycle. They applied their method together with the

reduced cost optimality condition instead of solving an assignment problem in each iteration to reduce the time complexity of the second algorithm to $O(n^2 \log n)$ (see Theorem 4 in [6]). Here, we give an example to show their method for computing reduced costs is not valid in the general case.

Let us first review some notions. Each basic feasible solution $\mathbf{x}$ of the assignment problem corresponds to a spanning tree T of G where $x_{ij'} = 0$ if $(i, j') \notin T$. Therefore, one can construct a basic feasible solution $\mathbf{x}$ from a given assignment M by adding $n-1$ additional arcs to M such that M becomes a spanning tree of G. An assignment M corresponds to several basic feasible solutions because there exist various ways for choosing additional arcs. For a basic feasible solution corresponding to tree T, we refer to the arcs belonging to T as tree arcs and arcs not belonging to T as nontree arcs.

Suppose that we assign a node potential p_i to each $i \in V$. The reduced cost of each arc (i, j') is defined as $c^p_{ij'} = c_{ij'} - p_i + p_{j'}$. For a basic feasible solution, one can obtain the node potentials by the fact that $c^p_{ij'} = 0$ for each tree arc (i, j') and then, compute the reduced cost of each nontree arc. Another approach is to compute the reduced costs by using the fact that the reduced cost of each nontree arc (i, j') equals $\mathbf{c}'(C_{ij'}) = \sum_{(k,l') \in C_{ij'}} c'_{kl'}$ where $C_{ij'}$ is the unique cycle contained in $T \cup \{(i, j')\}$. The following lemma states an optimality condition for a basic feasible solution of the assignment problem.

Lemma 3. ***(Reduced cost optimality condition)*** *A basic feasible solution* $\mathbf{x}$ *is optimal to the assignment problem if the reduced cost of each nontree arc is nonnegative.*

Proof. Since the assignment problem is a special case of the minimum cost flow problem, the proof is immediate by the reduced cost optimality condition to the minimum cost flow problem (see Chapter 11 of [1]). □

Lemma 3 is not a necessary condition, i.e., it is possible that some reduced costs of a basic feasible flow are negative while the corresponding assignment is optimal due to degeneracy [1, 3].

Now, we review the method computing the reduced costs presented by Duin and Volgenant [6]. An assignment M of the network $G(V_1 \cup V_2, V_1 \times V_2, \mathbf{c})$ can be represented as $(i, \pi(i))$ for $i = 1, 2, \ldots, n$ where π is a bijective mapping from V_1 onto V_2. For computing the reduced cost of the arcs (i, j') emanating from some node i, they introduced a basic solution tree T_i which has $n-1$ additional arcs $(l, \pi(l+1))$ for each $l \in V_1 \backslash \{i\}$ with assuming $\pi(n+1) = \pi(1)$ (see Figure 1 (a)). It is remarkable that the cycle $C_{ij'}$ is an alternating cycle for each $j' \in V_2$. Then, one can compute $c^p_{ij'}$ either by obtaining the cost of the corresponding alternating cycle $C_{ij'}$ or computing node potentials.

Example 1. The assignment problem associated with cost values shown in Table 1 is considered.

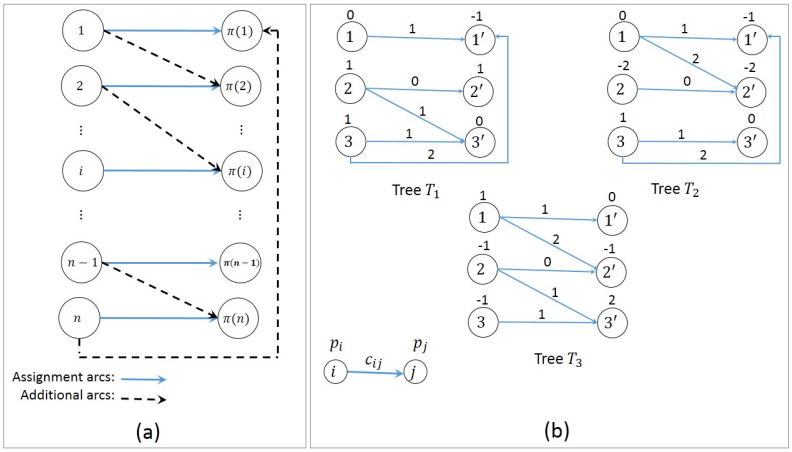

Figure 1. **(a) A tree T_i; (b) trees T_1, T_2 and T_3 for Example 1.**

Persons \ Tasks	$1'$	$2'$	$3'$
1	1	2	0
2	0	0	1
3	2	0	1

Table 1. **Cost values of Example 1**

It is easy to see that an optimal assignment is $1 \leftrightarrow 3', 2 \leftrightarrow 1'$ and $3 \leftrightarrow 2'$ with the objective value 0. Here, we consider the assignment $1 \leftrightarrow 1', 2 \leftrightarrow 2'$ and $3 \leftrightarrow 3'$. Clearly, it is not optimal. Figure 1 (b) illustrates the basic solution trees T_1, T_2 and T_3 and also, the node potentials p_i, $i \in V$. Table 2 provides the reduced cost of each nonbasic arc by the method proposed in [6].
Since all of the reduced cost values are nonnegative, Lemma 3 implies that the assignment is optimal which leads to a contradiction. Therefore, their method is not valid.

Basic tree	Associated nonbasic arcs	Reduced cost of arc (i, j') $(c^p_{ij'} = c_{ij'} - p_i + p_{j'})$
T_1	$(1, 2')$ $(1, 3')$	$c^p_{12'} = 2 - 0 + 1 = 3$ $c^p_{13'} = 0 - 0 + 0 = 0$
T_2	$(2, 1')$ $(2, 3')$	$c^p_{21'} = 0 - (-2) + (-1) = 1$ $c^p_{23'} = 1 - (-2) + 0 = 3$
T_3	$(3, 1')$ $(3, 2')$	$c^p_{31'} = 2 - (-1) + 0 = 3$ $c^p_{32'} = 0 - (-1) + (-1) = 0$

Table 2. **Basic trees of Example 1**

Their method is based on changing dynamically basic trees in such a way that each cycle $C_{ij'}$ becomes an alternating cycle and consequently, the reduced cost $c^p_{ij'}$ equals the cost of $C_{ij'}$. The major problem is that they applied Lemma 3 for a given assignment which corresponds to several basic feasible solutions while Lemma 3 is on the optimality of one basic feasible solution. Therefore, it can not be applied in a situation like this one.
Although each cycle $C_{ij'}$ corresponds to an alternating cycle, Lemma 1 can not be used for checking the optimality because some negative alternating cycles are not identified by their method. As an example, in the tree T_1, the reduced cost of $(2, 1')$, i.e., $c^p_{21'} = c_{21'} - p_2 + p_{1'} = 0 - 1 + (-1) = -2 < 0$, corresponds to the negative alternating cycle $2 - 1' - 3 - 3' - 2$ with the cost equal to -2 which is not identified by the method.

4. Conclusions

In this article, we considered the inverse maximum perfect matching problem under the bottleneck-type Hamming distance. We proposed an algorithm based on the binary search approach for solving the problem. The algorithm runs in $O(n^3 \log n)$ time which has better time complexity than the one presented in [13] with an $O(mn^3)$ time bound. As a special case, we also considered the inverse assignment problem and showed that one of the approaches proposed in [6] does not solve the problem in the general case. Finally, we proposed an algorithm for solving the problem in $O(mn \log n)$ time. This algorithm has better time complexity than the other approaches presented in [6] which run in $O(n^3 \log n)$ and $O(mn^2)$ time, respectively.

Acknowledgments

The author wishes to thank the anonymous referees for valuable comments which improves the paper.

References

[1] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin, *Network flows*, Prentice-Hall, Englewood Cliffs, 1993.
[2] M. Aman and J. Tayyebi, *Capacity inverse minimum cost flow problem under the weighted hamming distances*, Iranian Journal of Operations Research **5** (2014), no. 2, 12–25.
[3] M.S. Bazaraa and J. Jarvis, *Linear programming and network flows*, Wiley, New York, 1997.

[4] C. Berge, *Two theorems in graph theory*, Proceedings of the National Academy of Sciences **43** (1957), no. 9, 842–844.

[5] M. Demange and J. Monnot, *An introduction to inverse combinatorial problems, in: Paradigms of combinatorial optimization (problems and new approaches)*, Wiley, London-Hoboken (UK-USA), Vangelis Th. Paschos, 2010, 547–586.

[6] C.W. Duin and A. Volgenant, *Some inverse optimization problems under the hamming distance*, European J. Oper. Res. **170** (2006), no. 3, 887–899.

[7] J. Edmonds, *Maximum matching and a polyhedron with 0, 1-vertices*, J. Res. Natl. Bureau Stand. B **69B** (1965), no. 1-2, 125–130.

[8] Y. He, B. Zhang, and E. Yao, *Weighted inverse minimum spanning tree problems under hamming distance*, J. Comb. Optim. **9** (2005), no. 1, 91–100.

[9] Y. Jiang, L. Liu, B. Wu, and E. Yao, *Inverse minimum cost flow problems under the weighted hamming distance*, European J. Oper. Res. **207** (2010), no. 1, 50–54.

[10] E.L. Lawler, *Combinatorial optimization: Networks and matroids*, Holt, Rinehart and Winston, New York, 1976.

[11] L. Liu and Q. Wang, *Constrained inverse min–max spanning tree problems under the weighted hamming distance*, J. Global Optim. **43** (2009), no. 1, 83–95.

[12] L. Liu and E. Yao, *Inverse minmax spanning tree problem under the weighted sum-type hamming distance*, Theoret. Comput. Sci. **396** (2008), 28–34.

[13] ———, *Weighted inverse maximum perfect matching problems under the hamming distance*, J. Global Optim. **55** (2013), no. 3, 549–557.

[14] Z. Liu and J. Zhang, *On inverse problems of optimum perfect matching*, J. Comb. Optim. **7** (2003), no. 3, 215–228.

[15] J. Roland, Y.D. Smeta, and J. Figueira, *Inverse multi-objective combinatorial optimization*, Discrete Appl. Math. **161** (2013), no. 16-17, 2764–2771.

[16] J. Tayyebi and M. Aman, *Note on "inverse minimum cost flow problems under the weighted hamming distance"*, European J. Oper. Res. **234** (2014), no. 3, 916–920.

[17] ———, *On inverse linear programming problems under the bottleneck-type weighted hamming distance*, Discrete Appl. Math. **240** (2018), 92–101.