An $O(n^3)$ time algorithm for the maximum weight b-matching problem on bipartite graphs

Fatemeh Rajabi-Alni, Alireza Bagheri and Behrouz Minaei-Bidgoli

Abstract

Background: A matching between two sets $A$ and $B$ assigns some elements of $A$ to some elements of $B$. Finding the similarity between two sets of elements by advantage of the matching is widely used in computational biology for example in the contexts of genome-wide and sequencing association studies. Frequently, the capacities of the elements are limited. That is, the number of the elements that can be matched to each element should not exceed a given number.

Results: We use bipartite graphs to model relationships between pairs of objects. Given an undirected bipartite graph $G = (A \cup B, E)$, the b-matching of $G$ matches each vertex $v$ in $A$ (resp. $B$) to at least 1 and at most $b(v)$ vertices in $B$ (resp. $A$), where $b(v)$ denotes the capacity of $v$. We propose the first $O(n^3)$ time algorithm for finding the maximum weight b-matching of $G$, where $|A| + |B| = O(n)$.

Conclusions: The b-matching has been studied widely for the bipartite graphs with integer weight edges. But our algorithm is the first algorithm for the maximum (respectively minimum) b-matching problem with non positive real (respectively non negative real) edge weights.

Keywords: Hungarian algorithm; Many to many matching; Limited capacity; Bipartite graphs

Background

Given two sets of objects $A$ and $B$, a matching matches each object of $A$ (respectively $B$) to at least one object of $B$ (respectively $A$). The matching has many uses including computational biology and pattern recognition [1–3]. We can represent the sets and their relations using a bipartite graph; for example one part can represent mutated gens and other part outlying gens [2]. Given a weighted bipartite graph $G = (A \cup B, E)$, a matching in $G$ is the set of the vertex disjoint edges $PM \subseteq E$. The weight of the matching $M$ which is denoted by $W(M)$ is the sum of the weights of all the edges in $M$, hence

$$W(M) = \sum_{e \in M} W(e),$$

where $W(e)$ denotes the weight of the edge $e$. A maximum weight matching MWM is a matching that for any other matching $M'$, we have $W(M') \leq W(MWM)$. A perfect matching is a subset of edges $PM \subseteq E$ such that every vertex of $G$ is adjacent to exactly one edge of $PM$. The first polynomial time algorithm for computing the maximum weight perfect bipartite matching (MWPBM) is the basic
Hungarian algorithm [4, 5]. Then, Fredman et al. [6] solved it in $O(mn + n^2 \log n)$ time by implementing the Hungarian algorithm using fibonacci heaps. Later, other algorithms were developed for bipartite graphs with integer weights [7, 8].

The capacity of a vertex $v$ is the number of the vertices that can be matched to $v$, denoted by $b(v)$. We use $\deg(v)$ to refer to the degree of the vertex $v$ in the matching. Given an undirected bipartite graph $G = (A \cup B, E)$, the $b$-matching finds an edge set such that $1 \leq \deg(v) \leq b(v)$ for all $v \in A \cup B$. In many applications the capacities of objects are limited. For example, consider a biological pattern $A$, called the target pattern, as a set of points. When we want to find the target pattern $A$ in other set of points $B$, the number of the points of $A$ which can be matched to each point of $B$ is limited. Assume $|A| + |B| = n$ and $|E| = m$. The best known algorithm for the maximum weight $b$-matching problem has the time complexity of $O(W'\sqrt{\beta' m})$ for integer edge weights [9], where $W' = \max_{e \in E} W(e)$, and $\beta' = \sum_{v \in A \cup B} b(v)$. In this paper, we present an $O(n^3)$-time algorithm for the maximum weight $b$-matching problem for non-positive real weighted edges. Note that a maximum weight $b$-matching in $G$ with non-positive real edge weights $W(e)$ is a minimum weight $b$-matching in $G$ with non-negative real edge weights $F(e) = -W(e)$ for all $e \in E$. Our algorithm is an interesting one for example in dense graphs.

We first review the basic Hungarian algorithm and some preliminary definitions. Then, we present our new algorithm.

**Methods**

Let $G = (A \cup B, E)$ be a weighted bipartite graph such that $|A| = |B| = n$, $|E| = m$ and the edge weights are non-positive real values. A path with the edges alternating between the edges of the matching $M$ and $E - M$ is called an alternating path. Each vertex $v$ that is incident to an edge in $M$ is called a matched vertex; otherwise it is a free vertex. An alternating path with two free endpoints is called an augmenting path. Note that if the edges of an augmenting path that are in $M$ are replaced with the ones that are in $E - M$, an augmentation, its size increases by 1.

A vertex labeling is a function $l : V \rightarrow \mathbb{R}^- \cup \{0\}$ with $V = A \cup B$ that assigns a non-positive real value as a label to each vertex $v \in V$. A vertex labeling that in which $l(a) + l(b) \geq W(a, b)$ for all $a \in A$ and $b \in B$ is called a feasible labeling, where $W(a, b)$ denotes the weight of the edge $(a, b)$. The equality graph of a feasible labeling $l$ is a graph $G = (V, E_l)$ such that $E_l = \{(a, b) \in E | l(a) + l(b) = W(a, b)\}$. The neighbors of a vertex $u \in V$ are defined as $N_l(u) = \{v \in V | (v, u) \in E_l\}$.

Consider a set of vertices $S \subseteq V$, the neighbors of $S$ are $N_l(S) = \bigcup_{u \in S} N_l(u)$.

**Lemma 1** Consider a feasible labeling $l$ of an undirected bipartite graph $G = (A \cup B, E)$ and $S \subseteq A$ with $T = N_l(S) \neq B$, let

$$
\alpha_l = \min_{a_i \in S, b_j \not\in T} (l(a_i) + l(b_j) - W(a_i, b_j)).
$$

If the labels of the vertices of $G$ are updated such that:

$$
l'(v) = \begin{cases} 
  l(v) - \alpha_l & \text{if } v \in S \\
  l(v) + \alpha_l & \text{if } v \in T \\
  l(v) & \text{Otherwise}
\end{cases}
$$
then \( l' \) is a feasible labeling such that \( E_l \subset E_{l'} \).

**Proof.** Obviously in the cases \((a \in S, b \in T), (a \notin S, b \in T)\) and \((a \notin S, b \in T)\), we have:

\[
l'(a) + l'(b) \geq l(a) + l(b) \geq W(a, b).
\]

And for some vertices \(a \in S, b \notin T\), we have

\[
l'(a) + l'(b) = l(a) - \alpha + l(b) = W(a, b).
\]

**Theorem 1** If \( l \) is a feasible labeling and \( M \) is a perfect matching in \( E_l \), then \( M \) is a max-weight matching [4].

**Proof.** Suppose that \( M' \) is a perfect matching in \( G \), since each vertex is incident to exactly one edge of \( M' \) we have:

\[
W(M') = \sum_{(a,b) \in M'} W(a,b) \leq \sum_{v \in (A \cup B)} l(v).
\]

Thus, \( \sum_{v \in (A \cup B)} l(v) \) is an upper bound for each perfect matching. Now assume that \( M \) is a perfect matching in \( E_l \):

\[
W(M) = \sum_{e \in M} l(e) = \sum_{v \in (A \cup B)} l(v).
\]

**Lemma 2** After each augmentation of a matching, the cost of the matching decreases.

**Proof.** Given an augmenting path \( P \), two cases arise:

- \( P = (p_1, p_2) \). According to non-positive real edge weights, this condition is trivial.
- \( P = (p_1, p_2, p_3, \ldots, p_n) \) (see Figure 1). Note that

\[
W(p_i, p_{i+1}) = l(p_i) + l(p_{i+1}) \text{ for } i = 1, 2, \ldots, n - 1, \text{ since all edges of an augmenting path are in } E_l. \text{ Assume for a contradiction that the lemma is false, and thus}
\]

\[
W(p_1, p_2) + W(p_3, p_4) + \ldots + W(p_{n-1}, p_n) > W(p_2, p_3) + W(p_4, p_5) + \ldots + W(p_{n-2}, p_{n-1}). \text{ So it holds that:}
\]

\[
l(p_1) + l(p_2) + \ldots + l(p_n) > l(p_2) + l(p_3) + \ldots + l(p_{n-1}) \text{ and thus: } l(p_1) + l(p_n) > 0.
\]

Note that both \( p_1 \) and \( p_n \) are free, and according to the above feasible labeling we have \( l(p_n) \leq 0 \) and \( l(p_1) \leq 0 \). Contradiction.

Now, we review the basic Hungarian algorithm which computes a MWPBM in an undirected bipartite graph \( G = (A \cup B, E) \) with \(|A| = |B| = n\) (see Algorithm 1). It is shown that the maximum weight matching problem in bipartite graphs can be reduced to the MWPBM problem and solved using the Hungarian algorithm in \( O(n^3) \) time [10].

In lines 2 and 3 of Algorithm 1, the vertices of the input bipartite graph are labeled by a feasible labeling. \( M \) is an initial matching which can be empty (line 4). In each iteration of the while loop of lines 5–21 the size of \( M \) is increased by 1, so it iterates at most \( n \) times. Let

\[
slack[j] = \min_{a_i \in S} (l(a_i) + l(b_j) - W(a_i, b_j)),
\]
Algorithm 1 The Basic Hungarian algorithm(G)

1: Initial \textComment{Find an initial feasible labeling \(l\) and a matching \(M\) in \(E_l\)}
2: \text{Let } \(l(b_j) = 0\), for all \(1 \leq j \leq n\)
3: \(l(a_i) = \max_{j=1}^{n} W(a_i, b_j)\) for all \(1 \leq i \leq n\)
4: \(M = \emptyset\)
5: \textbf{while} \(M\) is not perfect \textbf{do}
6: \textbf{Select a free vertex } \(a_i \in A\) and set \(S = \{a_i\}\), \(T = \emptyset\)
7: \textbf{for} \(j \leftarrow 1\) to \(n\) \textbf{do}
8: \textbf{if} \(l(b_j) \neq 0\) \textbf{then}
9: \textbf{Repeat}
10: \textbf{if} \(N_i(S) = T\) \textbf{then}
11: \(\alpha_t = \min_{b_j \notin T} slack[j]\)
12: \textbf{Update} \(l\)
13: \textbf{for} all \(b_j \notin T\) \textbf{do}
14: \(slack[j] = slack[j] - \alpha_t\)
15: \textbf{Select} \(u \in N_i(S) - T\)
16: \textbf{if} \(u\) is not free \textbf{then} \textComment{(\(u\) is matched to a vertex \(z\), extend the alternating tree)}
17: \(S = S \cup \{z\}\), \(T = T \cup \{u\}\)
18: \textbf{for} \(j \leftarrow 1, n\) \textbf{do}
19: \(slack[j] = \min(l(z) + l(b_j) - W(z, b_j), slack[j])\)
20: \textbf{until} \(u\) is free
21: \textbf{Augment} \(M\)
22: \textbf{return} \(M\)

by advantage of the array \(slack[1..n]\), each iteration of the while loop can be run in \(O(n^2)\) time.

The repeat loop of lines 9–20 runs at most \(O(n)\) times until finding a free vertex \(u\). In line 11, the value of \(\alpha_t\) can be computed by:

\(\alpha_t = \min \{\text{slack}[j]\}\)

in \(O(n)\) time. After computing \(\alpha_t\), in line 12 the feasible labeling \(l\) is updated such that \(N_i(S) \neq T\). The values of the slacks must also be updated (lines 13–14):

\(\text{for all } b_j \notin T, \text{slack}[j] = \text{slack}[j] - \alpha_t\)

A vertex \(u \in N_i(S) - T\) is selected in line 15. Observe that if \(u\) is not a free vertex, the alternating tree should be extended (lines 15–17). Note that in the repeat loop, an alternating tree, a tree with alternating paths, is constructed to find an augmenting path. Once a vertex is moved form \(\tilde{S}\) to \(S\), the values of \(skack[1..n]\) are updated (lines 18–19) in \(O(n)\) time. At most \(O(n)\) vertices are moved from \(\tilde{S}\) to \(S\), so the repeat loop takes the total time of \(O(n^2)\). Therefore, the time complexity of the basic Hungarian algorithm is \(O(n^3)\).

Results and discussion

The maximum weight \(b\)-matching algorithm on bipartite graphs

Let \(G = (A \cup B, E)\) be a bipartite graph with non-negative real edge weights, where \(A = \{a_1, a_2, \ldots, a_s\}\) and \(B = \{b_1, b_2, \ldots, b_t\}\) such that \(s + t = n\). Let \(C_A = \{\alpha_1, \alpha_2, \ldots, \alpha_s\}\) and \(C_B = \{\beta_1, \beta_2, \ldots, \beta_t\}\) denote the capacities of \(A\) and \(B\), respectively. Assume w.l.o.g that \(t \geq s\). We present an \(O(n^3)\) time algorithm for computing a maximum weight \(b\)-matching in \(G = (A \cup B, E)\), where each vertex \(a_i \in A\) must be matched to at least 1 and at most \(\alpha_i\) vertices in \(B\), and each vertex
b_j \in B$ must be matched to at least 1 and at most $\beta_j$ vertices in A for all $1 \leq i \leq s$ and $1 \leq j \leq t$.

Firstly, we construct a bipartite graph $G' = (X \cup Y, E)$ with $X = A \cup A'$ and $Y = B \cup B'$ as follows (see Figure 2). Then, we run our algorithm on it.

In a complete connection between two sets each element of one set is connected to the all elements of the other set. We show each set of the vertices by a rectangle and the complete connection between them by a line connecting the two corresponding rectangles.

Given $A = \{a_1, a_2, \ldots, a_s\}$ and $B = \{b_1, b_2, \ldots, b_t\}$, we construct a complete connection between $A$ and $B$ where the weight of $(a_i, b_j)$ is equal to the cost of matching the point $a_i$ to $b_j$ for all $1 \leq i \leq s$ and $1 \leq j \leq t$.

Let $B'_1 = \{b'_1, b'_2, \ldots, b'_{\beta_j-1}\}$ for $1 \leq j \leq t$, and $B' = \{B'_1, B'_2, \ldots, B'_t\}$. Each vertex of $A$ is connected to the all vertices of $B'$ such that

$$W(a_i, b'_{jk}) = W(a_i, b_{ijk})$$

for all $1 \leq k \leq (\beta_j - 1)$.

Let $A'_1 = \{a'_1, a'_2, \ldots, a'_{\alpha_i-1}\}$ for $1 \leq i \leq s$ and $A' = \{A'_1, A'_2, \ldots, A'_s\}$, we also construct a complete connection between the sets $B$ and $A'$ such that

$$W(a'_{ik}, b_j) = W(a_i, b_j)$$

for all $1 \leq k \leq (\alpha_i - 1)$.

Now, we modify the basic Hungarian algorithm to get a new algorithm, called ModifiedHungarianAlg (see Algorithm 2). In the modified Hungarian algorithm, line 5 of Algorithm 1 is changed; the while loop is iterated until matching the subset $A \subseteq X$. By Lemma 2, using ModifiedHungarianAlg, we get an optimal matching in which the vertices of $A$ are matched, but some vertices of $B$ might be free. The initialization step is also removed.

Our new algorithm consists of two steps (see Algorithm 3); in the first step the vertices of $A \subseteq X$ are matched, and in the second one the vertices of $B \subseteq Y$. We claim that by applying our algorithm on $G'$, Maxweight b-matching Algorithm ($G = (A \cup B, E)$), we get a maximum weight b-matching between $A$ and $B$.

1 **Step I.** Given an undirected bipartite graph $G' = (X \cup Y, E)$ with $X = A \cup A'$ and $Y = B \cup B'$, in this step we call the function ModifiedHungarianAlg($G', A, M, l$). It matches an arbitrary vertex of each $B'_j$ for all $j = \{1, 2, \ldots, t\}$ until there does not exist any unmatched vertex in $A \subseteq X$.

2 **Step II.** In this step, we call the function ModifiedHungarianAlg($G', B, M, l$). We use the final labels and slacks of the vertices in Step I, so the labels of the vertices are feasible labeling. We also use the output matching of Step I as the initial matching of Step II. Once all vertices of $B \subseteq Y$ are matched, this step terminates. Note that by Lemma 2, if we continue matching the vertices of $B'$ the cost decreases. For each $a_i \in A$ for $1 \leq i \leq s$, there exists the set $\{a_i\} \cup A'_i$ in $G'$ with $\alpha_i$ vertices. Also there exist $\beta_j$ copies of each $b_j \in B$ in the set $\{b_j\} \cup B'_j$ in $G'$ for $1 \leq j \leq t$. So, the capacities of the vertices of $A$ and $B$ are satisfied.
Time complexity

The while loop of lines 1–23 of ModifiedHungarianAlg($G', A, M, l$), called the main loop, is iterated until all vertices of $A$ are matched to exactly one vertex of $B \cup B'$. Obviously, it iterates $O(n)$ times, since the number of the vertices of $A$ is $O(n)$. In the following, we show that each iteration of the main loop of ModifiedHungarianAlg($G', A, M, l$) is done in $O(n^2)$ time.

**Observation 1** The labels of all free vertices $b'_{jk} \in B'_j$ are equal for all $1 \leq k \leq \beta_j - 1$.

Initially, we have $l(b'_{jk}) = 0$ for all $1 \leq k \leq \beta_j - 1$. The function $\text{Update}(l)$ updates the labels of all the vertices $b'_{jk} \in T$, i.e. all the vertices $b'_{jk}$ that have been matched to a vertex in $S$. Hence, all free vertices $b'_{jk} \in B'_j$ have equal labels for $1 \leq k \leq \beta_j - 1$.

**Observation 2** The values of the slacks of all free vertices $b'_{jk} \in B'_j$ are equal for all $1 \leq k \leq \beta_j - 1$.

Observation 1 implies that the values of the slacks of all free vertices $b'_{jk} \in B'_j$ with $1 \leq k \leq \beta_j - 1$ are equal.

**Algorithm 2** ModifiedHungarianAlg($G' = (X \cup Y, E), A, M, l$)

1: while $\{u \in A | u \text{ is free} \} \neq \emptyset$ do
2: Select a free vertex $x_i \in A$ and set $S = \{x_i\}, T = \emptyset$
3: $B'' = \emptyset$
4: for $j \leftarrow 1$ to $|B|$ do
5: Select a free vertex $v \in B'_j$
6: $B'' = B'' \cup \{v\}$
7: Let $C$ be the set of matched vertices of $B'$
8: $Y' = B \cup C \cup B''$
9: for all $y_j \in Y'$ do
10: $\text{slack}[j] = l(x_i) + l(y_j) - W(x_i, y_j)$
11: repeat
12: if $N_l(S) = T$ then
13: $\alpha_i = \min_{y_j \in Y' \setminus T} \text{slack}[j]$  
14: $\text{Update}(l)$  
15: for all $y_j \in Y' \setminus T$ do
16: $\text{slack}[j] = \text{slack}[j] - \alpha_i$
17: Select $u \in N_l(S) - T$
18: if $u$ is not free then  
19: $S = S \cup \{u\}, T = T \cup \{u\}$
20: for all $y_j \in Y'$ do
21: $\text{slack}[j] = \min(l(z) + l(y_j) - W(z, y_j), \text{slack}[j])$
22: until $u$ is free
23: $\text{Augment}(M)$
24: return $M$ and $l$

**Algorithm 3** Maxweight b-matching Algorithm($G = (A \cup B, E)$)

1: Construct the bipartite graph $G' = (X \cup Y, E')$ from $G$ with $X = A \cup A'$ and $Y = B \cup B'$
2: Initial  
3: Let $p = |X|$, $q = |Y|$
4: Let $l(y_j) = 0$ for all $1 \leq j \leq q$, and let $l(x_i) = \max_{j=1}^q W(x_i, y_j)$ for all $1 \leq i \leq p$
5: $M = \emptyset$
6: $(M, l) = \text{ModifiedHungarianAlg}(G', A, M, l)$
7: $(M, l) = \text{ModifiedHungarianAlg}(G', B, M, l)$
8: return $M$

Note that the vertices of $B'_j$ are copies of a single vertex, i.e. $b_j$. By Observations 1 and 2, all free vertices of $B'_j$ have equal labels and slacks, so in each iteration of the main loop, we consider only one of the free vertices $b'_{jk} \in B'_j$ for $1 \leq k \leq \beta_j - 1$, 


arbitrarily. Actually, in each iteration of the main loop, all the free vertices \( b'_{jk} \in B'_j \) are considered as a single vertex (line 7). Let \( B'' = b''_1, b''_2, \ldots, b''_n \), where \( b''_j \) is an arbitrary free vertex of \( B'_j \), if exists. Let \( Y' = B \cup C \cup B'' \), where \( C \) is the set of the matched vertices of \( B' \) with respect to \( M \) (lines 7–8). In each iteration of the main loop, we first give the slacks of all vertices \( y_j \in Y' \) initial values in \( O(n) \) time (lines 9–10). Then the repeat loop of lines 11–22 is iterated until we find an augmenting path with the starting vertex \( x_i \). Note that if \( N_i(S) = T \), we can always update the labels of the vertices of \( G' \) to get a new feasible labeling that in which \( N_i(S) \neq T \). Notice that the neighbor set of a vertex \( u \in A \) is defined as \( N_i(u) = \{ v \in Y' | (u, v) \in E_i \} \). Note that there exist at most \( O(n) \) matched vertices, i.e. the vertices of \( A \), so the number of the vertices of \( T \) and \( S \) are at most \( O(n) \) vertices. Hence, in line 13, we get the minimum value in \( O(n) \) time. Also, updating the labels and slacks is done in \( O(n) \) time (lines 14–16 and 20–21). As the first step, we can show that the time complexity of the second step is also \( O(n^3) \). We observe that the values of the slacks and labels of all free vertices \( a'_{ik} \in A'_i \) for all \( 1 \leq k \leq \alpha_i - 1 \) are equal.

**Observation 3** The values of the slacks and labels of all free vertices \( a'_{ik} \in A'_i \) are equal for all \( 1 \leq k \leq \alpha_i - 1 \).

**Theorem 2** Let \( G = (A \cup B, E) \) be a non-positive real weighted bipartite graph with \(|A| + |B| = n\), a maximum weight b-matching in \( G \) can be computed in \( O(n^3) \) time.

**Conclusions**

In this paper, we proposed an \( O(n^3) \) time algorithm for finding a maximum weight b-matching in a bipartite graph \( G = (A \cup B, E) \) with \(|A| + |B| = n\), which can be used for example for computing the overall similarity of two given genomes. We modified the basic Hungarian algorithm and presented a new algorithm for a more general version of the matching problem. The b-matching problem has been studied widely in bipartite graphs with integer weight edges. But to the best of our knowledge, our algorithm is the first algorithm for the maximum weight b-matching problem in a bipartite graph with non positive real edge weights. In the future, we hope to develop new algorithms for other versions of the b-matching problem depending on the properties of two given input sets.

**Abbreviations**

MWPBM: maximum weight perfect bipartite matching

**Declarations**

**Ethics approval and consent to participate**
Not applicable.

**Consent to publish**
Not applicable.

**Availability of data and materials**
Not applicable.

**Competing interests**
The authors declare that they have no competing interests.

**Funding**
Not applicable.
Authors’ Contributions
FR-A developed the theoretical results. FR-A, AB and BM-B discussed extensively about this study and analyzed
the method. FR-A and AB wrote the manuscript. BM-B participated in revisiting the draft. All authors have read
and approved the manuscript.

Acknowledgements
The author would like to thank the referees for their careful reading of the paper and helpful comments.

Author details
1Department of Computer Engineering, Amirkabir University of Technology, Tehran, Iran. 2Department of
Computer Engineering, Iran University of Science and Technology, Tehran, Iran.

References
1. Lo, C., Kim, S., Zakov, S., Bafna, V.: Evaluating genome architecture of a complex region via generalized
bipartite matching. BMC Bioinformatics 14, 13 (2013)
2. Song, J., Peng, W., Wang, F.: A random walk-based method to identify driver genes by integrating the
subcellular localization and variation frequency into bipartite graph. BMC Bioinformatics 20, 238 (2019)
3. Rubert, D., Hoshino, E., Braga, M., Stoye, J., Martinez, F.: Computing the family-free dcj similarity. BMC
Bioinformatics 19, 152 (2018)
4. Kuhn, H.W.: The hungarian method for the assignment problem. Naval Research Logistics Quarterly 2, 83–97
(1955)
5. Munkres, J.: Algorithms for the assignment and transportation problems. J. Soc. Indust. Appl. Math 5, 32–38
(1957)
6. Fredman, M.L., Tarjan, R.E.: Fibonacci heaps and their uses in improved network optimization algorithms. J.
ACM 34(3), 596–615 (1987)
7. Gabow, H.N., Tarjan, R.E.: Faster scaling algorithms for network problems. SIAM J. Comput. 18(5),
1013–1036 (1989)
8. Orlin, J.B., Ahuja, R.K.: New scaling algorithms for the assignment and minimum mean cycle problems. Math.
Program. 56, 41–56 (1992)
9. Huang, C.-C., Kavitha, T.: New algorithms for maximum weight matching and a decomposition theorem.
Math. Oper. Res. 42(2), 411–426 (2017)
10. Eiter, T.B., Mannila, H.: Distance measures for point sets and their computation. Acta Inform. 34, 109–133
(1997)