Computing a many-to-many matching with demands and capacities between two sets using the Hungarian algorithm

Fatemeh Rajabi-Alni\textsuperscript{1}, and Alireza Bagheri\textsuperscript{1}

\textsuperscript{1}Computer Engineering Department, Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran
Emails: f.rajabialni@aut.ac.ir (F. Rajabi-Alni), ar_bagheri@aut.ac.ir (A. Bagheri)

Abstract

Given two sets $A = \{a_1, a_2, \ldots, a_s\}$ and $B = \{b_1, b_2, \ldots, b_t\}$, a many-to-many matching with demands and capacities (MMDC) between $A$ and $B$ matches each element $a_i \in A$ to at least $\alpha_i$ and at most $\alpha'_i$ elements in $B$, and each element $b_j \in B$ to at least $\beta_j$ and at most $\beta'_j$ elements in $A$ for all $1 \leq i \leq s$ and $1 \leq j \leq t$. In this paper, we present an algorithm for finding a minimum-cost MMDC between $A$ and $B$ using the well-known Hungarian algorithm.

Keywords: Many-to-many matching; Hungarian method; bipartite graph; demands and capacities; minimum perfect matching

Introduction

A matching between two sets $A$ and $B$ defines a relationship between them. A many-to-many matching between $A$ and $B$ maps each element of $A$ to at least one element of $B$ and vice-versa. A perfect matching is a matching where each element is matched to a unique element. Eiter and Mannila\textsuperscript{1} solved the many-to-many matching problem in $O(n^3)$ time by reducing it to the minimum weight perfect matching problem in a bipartite graph, where $|A| + |B| = n$. We refer the readers to\textsuperscript{2} for a comprehensive survey on the matching theory and algorithms.

Let $A = \{a_1, a_2, \ldots, a_s\}$ and $B = \{b_1, b_2, \ldots, b_t\}$ be two sets, a many-to-many matching with demands and capacities (MMDC) matches each element $a_i \in A$ to $\alpha_i \leq d_i \leq \alpha'_i$ elements of $B$, and each element $b_j \in B$ to $\beta_j \leq d'_j \leq \beta'_j$ elements of $A$. MMDC problem is a specific case of the maximum weight degree-constrained subgraph problem in general graphs that in which for each vertex $v$ with degree $\text{deg}(v)$ we have $l(v) \leq \text{deg}(v) \leq u(v)$ ($l(v)$ and $u(v)$ denote integer bounds), and has the time complexity of $O(n^2 \min(m \log n, n^2))$\textsuperscript{3}. In this paper, we present an algorithm that computes a minimum-cost MMDC between $A$ and $B$ with $|A| + |B| = n$ in $O(n^6)$ time using the basic Hungarian algorithm. Also, our algorithm computes an MMDC between two sets of points in the plane in $O(n^4 \text{poly}(\log n))$ time using the modified Hungarian algorithm proposed in\textsuperscript{4}. Moreover, in bipartite graphs with low-range edge weights and dense graphs, our algorithm runs faster than its worst time complexity\textsuperscript{5}. In fact, our algorithm imposes upper and lower bounds on the number of elements that can be matched to each element in any version of the Hungarian algorithm.

Preliminaries

Given an undirected bipartite graph $G = (A \cup B, E)$, a matching in $G$ is a subset of the edges $M \subseteq E$, such that each vertex $v \in A \cup B$ is incident to at most one edge of $M$. Let $\text{Weight}(a, b)$ denote the weight of the edge $(a, b)$,
Lemma 1. Let \( N \) that in which \( l \) with \( T \) vertex \( u \). After updating the labels, four cases arise:

- if \( v \in S \), then \( l'(v) = l(v) + \alpha_l \).
- if \( v \in T \), then \( l'(v) = l(v) - \alpha_l \).
- otherwise, \( l'(v) = l(v) \).

that in which

\[
\alpha_l = \min_{a_i \in S, b_j \not\in T} \{Weight(a_i, b_j) - l(a_i) - l(b_j)\}.
\]

Then, \( l' \) is also a feasible labeling with \( E_l \subset E'_l \).

Proof. Note that \( l \) is a feasible labeling, so we have \( l(a) + l(b) \leq Weight(a, b) \) for each edge \( (a, b) \) of \( E \).

After updating the labels, four cases arise:

- \( a \in S \) and \( b \in T \). In this case, we have
  \[
l'(a) + l'(b) = l(a) + \alpha_l + l(b) - \alpha_l = l(a) + l(b) \leq Weight(a, b).
  \]

- \( a \not\in S \) and \( b \not\in T \). Then, we have
  \[
l'(a) + l'(b) = l(a) + l(b) \leq Weight(a, b).
  \]

- \( a \not\in S \) and \( b \in T \). We see that
  \[
l'(a) + l'(b) = l(a) + l(b) - \alpha_l < l(a) + l(b) \leq Weight(a, b).
  \]

- \( a \in S \) and \( b \not\in T \). In this situation, we have
  \[
l'(a) + l'(b) = l(a) + \alpha_l + l(b).
  \]

Then, two cases arise:
- \( Weight(a, b) - l(a) - l(b) = \alpha_l \). Then,
  \[
l'(a) + l'(b) = l(a) + \alpha_l + l(b) = l(a) - l(a) - l(b) + Weight(a, b) + l(b) = Weight(a, b).
  \]

Hence, \( E_l \subset E'_l \).
In Line 11, we update the feasible labeling $l$. Obviously

$$l'(a) + l'(b) = l(a) + \alpha_l + l(b) \leq \text{Weight}(a, b).$$

\[\square\]

**Theorem 1.** Let $l$ be a feasible labeling and $M$ a perfect matching in $E_l$. Then, $M$ is a minimum weight matching [6].

In the following, we briefly describe the basic Hungarian algorithm which computes a minimum weight perfect matching in an undirected bipartite graph $G = (A \cup B, E)$ with $|A| = |B| = n$ (Algorithm [1] [6, 7]).

**Algorithm 1:** The Basic Hungarian algorithm ($G = (A \cup B, E)$)

1. Let $l(b_j) = 0$, for all $1 \leq j \leq n$;
2. $l(a_i) = \min_{j=1}^n \text{Weight}(a_i, b_j)$ for all $1 \leq i \leq n$;
3. $M = \emptyset$;
4. while $M$ is not perfect do
   5. Select a free vertex $a_i \in A$ and set $S = \{a_i\}, T = \emptyset$;
   6. for $j \leftarrow 1, n$ do
      7. $\text{slack}[j] = l(a_i) + l(b_j) - \text{Weight}(a_i, b_j)$;
   8. repeat
      9. if $N_l(S) = T$ then
         10. $\alpha_l = \min_{b_j \notin T} \text{slack}[j]$;
         11. $\text{Update}(l)$ for all $b_j \notin T$ do
             12. $\text{slack}[j] = \text{slack}[j] + \alpha_l$;
         13. Select $u \in N_l(S) - T$;
         14. if $u$ is not free then
            15. $S = S \cup \{z\}, T = T \cup \{u\}$;
            16. for $j \leftarrow 1, n$ do
               17. $\text{slack}[j] = \min(\text{Weight}(z, b_j) - l(z) - l(b_j), \text{slack}[j])$;
         18. until $u$ is free;
         19. $\text{Augment}(M)$;
   20. return $M$;

In Lines 1–2, we label all vertices of $B$ with zero and each vertex $a_i \in A$ with $\min_{j=1}^n \text{Weight}(a_i, b_j)$ to get an initial feasible labeling. Note that $M$ can be empty (Line 3). In each iteration of the while loop of Lines 4–20, two free vertices $a_i$ and $b_j$ are matched, so it iterates $O(n)$ times. Using the array $\text{slack}[1, \ldots, n]$, we can run each iteration of this loop in $O(n^2)$ time. The repeat loop runs at most $O(n)$ times until finding a free vertex $b_j$. In Line 10, we can compute the value of $\alpha_l$ by:

$$\alpha_l = \min_{b_j \notin T} \text{slack}[j],$$

in $O(n)$ time. After computing $\alpha_l$ and updating the labels of the vertices, we must also update the values of the slacks. This can be done using:

$$\text{for all } b_j \notin T, \text{slack}[j] = \text{slack}[j] + \alpha_l.$$

In Line 11, we update the feasible labeling $l$ such that $N_l(S) \neq T$. In Line 15 of Algorithm [1] when a vertex is moved from $S$ to $S$, the values of $\text{slack}[1, \ldots, n]$ must be updated. This is done in $O(n)$ time. $O(n)$ vertices are moved from $S$ to $S$, so it takes the total time of $O(n^2)$.
The value of $\alpha_i$ may be computed at most $O(n)$ times in $O(n)$ time, so running each iteration takes at most $O(n^2)$ time. So, the time complexity of the basic Hungarian algorithm is $O(n^3)$. The Hungarian algorithm in the worst case does the repeat loop of the algorithm in $O(n^2)$ overall time; updating the labels using the function $Update(l)$ adds only one more edge to $l'$ ($|E_{l'}| = |E_l| + 1$), but we observe that in practice, in bipartite graphs with low-range edge weights and dense graphs, more edges are added to $E_l$ after running $Update(l)$.

**MATCHING ALGORITHM**

We construct a complete bipartite graph such that by applying the Hungarian method on it, the demands and capacity limitations of the elements are satisfied. In the following, we explain how our complete bipartite graph $G$ is constructed.

We represent a set of the related vertices using a rectangle, each connection between two vertices with a line and each vertex with a circle. So, a connection between two vertices is shown using a line that connects the two corresponding circles. A complete connection between two sets is a connection where each vertex of one set is connected to all vertices of the other set. We show a complete connection using a line connecting the two corresponding sets.

Let $S \cup T$ be a bipartition of $G$, where $S = (\bigcup_{i=1}^{s} A_i) \cup (\bigcup_{i=1}^{s} A'_i) \cup (\bigcup_{i=1}^{t} X_j) \cup (\bigcup_{j=1}^{t} W_j)$ and $T = \bigcup_{i=1}^{s} B_{set_i}$. The vertices of the sets $A_i$, $B_{set_i}$, and $A'_i$ for all $1 \leq i \leq s$ are called the main vertices, since they are copies of the input elements. On the other hand, the vertices of the sets $X_j$ and $W_j$ for all $1 \leq j \leq t$ are called the dummy vertices. All edges $(a,b)$ that their both end vertices are main vertices, that is $a \in A_i \cup A'_i$ and $b \in B_{set_i}$ for $1 \leq i \leq s$, are called the main edges.

The Hungarian method computes a perfect matching where each vertex is incident to a unique edge. We aim to find an MMDC matching in which two or more vertices may be matched to the same vertex, that is a vertex may be selected more than once. So, our constructed graph contains multiple copies of each element to simulate this.

Find an MMDC matching in which two or more vertices may be mapped to the same vertex, that is a vertex may be selected more than once. So, our constructed graph contains multiple copies of each element to simulate this.
Figure 1: The constructed complete bipartite graph $G$ by our algorithm.

\[
= \sum_{i=1}^{s} \alpha_i + \sum_{i=1}^{s} \alpha'_i - \sum_{i=1}^{s} \alpha_i + \sum_{j=1}^{t} \beta'_j - \sum_{j=1}^{t} \beta_j + s \ast t - \sum_{j=1}^{t} \beta'_j
\]

and

\[
\left| T \right| = \left| \bigcup_{i=1}^{s} \text{Bset}_i \right| = (s \ast t).
\]

Let $\left| Y \right| = \sum_{i=1}^{s} \alpha'_i - \sum_{j=1}^{t} \beta_j$. The compensator set $Y$ is inserted to $T$ as follows. Note that we have $\sum_{i=1}^{s} \alpha'_i > \sum_{j=1}^{t} \beta_j$. There is a complete connection between $X_j$ and $Y$ that in which the weight of the edges is an arbitrary number $\gamma''$ with $\gamma' < \gamma''$. Consequently, the priority of the vertices of $X_j$ is the vertices of $B_j$ set. Moreover, $A'_i$ is completely connected to $Y$ with $\gamma'$ weighted edges. Our constructed complete bipartite graph $G$ is shown in Figure 1.

We claim that from a minimum weight perfect matching in $G = S \cup T$ denoted by $M$, we can get a minimum-cost MMDC between $A$ and $B$. Let $\text{Mian}(M)$ be the union of the main edges of the minimum weight perfect matching $M$ in $G$. In the following, we prove that the weight of $\text{Mian}(M)$ is equal to the cost of a minimum-cost MMDC between $A$ and $B$, called $L$. Let $c(L)$ denote the cost of $L$.

**Lemma 2.** $\text{Weight}(\text{Main}(M)) \leq c(L)$.

**Proof.** We get from $L$ a perfect matching $M'$ in our complete bipartite graph $G$, such that the weight of the union of the main edges of $M'$, $\text{Main}(M')$, is equal to the cost of $L$, that is $\text{Weight}(\text{Mian}(M')) = c(L)$.

Let $p_i$ be the number of the elements $b_j \in B$ that are matched to $a_i \in A$ in $L$. It is obvious that $\alpha_i \leq p_i \leq \alpha'_i$. Firstly, for each pairing $(a_i, b_j)$ in $L$, we connect $b_{j_1}$ to one of the unmatched vertices of $A_i$, that is $a_{i_k}$ with $1 \leq k \leq \alpha_i$, until there does not exist any unmatched vertex in $A_i$. Then, depending on the value of $p_i$ two cases arise:

- either $p_i = \alpha_i$. In this situation, we add the $\gamma'$ weighted edges of $G$ connecting each $a'_{ij} \in A'_i$ to one of the unmatched vertices of $Y$ for all $1 \leq j \leq (\alpha'_i - \alpha_i)$. 

• or \( p_i > \alpha_i \). In this case, we need to match \( p_i - \alpha_i \) number of the vertices of \( A_i' \) with the vertices of \( B_{set_i} \). So, for each pairing \( (a_i, b_j) \) of the \( p_i - \alpha_i \) remaining pairings, we add an edge of \( G \) connecting \( a_i' j \) to \( b_j \).

Then, if yet there exist other vertices of \( A_i' \), that have not been matched to any vertex (i.e. \( p_i < \alpha_i' \)); for each of them, we select an edge of \( G \) connecting it to an unmatched vertex of \( Y \), and add it to \( M' \).

Then, for each \( w_{j,k} \in W_j \) we add the edge of \( G \) that connects it to an unmatched vertex of \( B_j \). The vertices of \( X_j \) are matched to the vertices of \( B_j \), unless no vertices remain unmatched in \( B_j \). So, we first add the edges connecting the vertices of \( X_j \) to the remaining unmatched vertices of \( B_j \). Then, we add the edges connecting the unmatched vertices of \( X_j \), if exist, to the unmatched vertices of \( Y \).

Since all vertices of \( G \) are selected once, \( M' \) is a perfect matching. For each \( (a_i, b_j) \in L \), there is an edge with equal weight in \( Mian(M') \), so \( \text{Weight}(Mian(M')) = c(L) \).

**Lemma 3.** Let \( M \) be a minimum weight perfect matching in \( G \). Then, for any perfect matching in \( G \) denoted by \( M' \) we have

\[
\text{Weight}(\text{Main}(M)) \leq \text{Weight}(\text{Main}(M')).
\]

**Proof.** Observe that we have:

\[
\text{Weight}(M) = \text{Weight}(\text{Main}(M)) + \text{Weight}(M - \text{Main}(M)).
\]

Note that for a minimum-cost MMDC between \( A \) and \( B \) denoted by \( L \) we have

\[
|L| = \max \left( \sum_{i=1}^{s} \alpha_i, \sum_{j=1}^{t} \beta_j \right).
\]

Given any perfect matching \( M'' \) in \( G \), the set \( M'' - \text{Main}(M'') \) contains:

- the zero weighted edges connecting the vertices of \( W_j \) to the vertices of \( B_j \) for \( 1 \leq j \leq t \), with the total number of \( s \ast t - \sum_{j=1}^{t} \beta_j \),
- the \( \gamma' \) weighted edges connecting \( \sum_{j=1}^{t} \beta_j' - |L| \) number of the vertices of \( X_j \) to the vertices of \( B_j \) for \( 1 \leq j \leq t \),
- the \( \gamma'' \) weighted edges connecting \( |L| - \sum_{j=1}^{t} \beta_j \) number of the vertex of \( X_j \) to \( Y \) for \( 1 \leq j \leq t \),
- the \( \gamma' \) weighted edges connecting \( \sum_{i=1}^{s} \alpha_i' - |L| \) number of the vertex of \( A_i' \) to \( Y \) for \( 1 \leq i \leq s \).

Thus

\[
\text{Weight}(M'' - \text{Main}(M'')) = \left( \sum_{j=1}^{t} \beta_j' - |L| \right) \ast \gamma'
\]

\[
+ (|L| - \sum_{j=1}^{t} \beta_j) \ast \gamma'' + \left( \sum_{i=1}^{s} \alpha_i' - |L| \right) \ast \gamma'.
\]

So, we have:

\[
\text{Weight}(M - \text{Main}(M)) = \text{Weight}(M' - \text{Main}(M')).
\]

Note that \( M \) is a minimum weight perfect matching in \( G \), thus

\[
\text{Weight}(M) \leq \text{Weight}(M'),
\]

and so
\[\text{Weight}(M - \text{Main}(M)) + \text{Weight}(\text{Main}(M)) \leq \] 
\[\text{Weight}(M' - \text{Main}(M')) + \text{Weight}(\text{Main}(M')) ,\]

Therefore, we have:
\[\text{Weight}(\text{Main}(M)) \leq \text{Weight}(\text{Main}(M')).\]

Note that \(\text{Weight}(\text{Main}(M')) = c(L),\) so
\[\text{Weight}(\text{Main}(M)) \leq c(L).\]

\[\square\]

**Lemma 4.** \(c(L) \leq \text{Weight}(\text{Main}(M)).\)

**Proof.** From \(\text{Main}(M),\) we get an MMDC between \(A\) and \(B\) denoted by \(L',\) such that the cost of \(L'\) is equal to the weight of \(\text{Main}(M),\) that is \(\text{Weight}(\text{Main}(M)) = c(L').\) For each edge \(m \in M,\) if \(m = (a_{ik}, b_{ji})\) or \(m = (a'_{ik}, b_{ji}),\) we add the pairing \((a_i, b_j)\) to \(L'.\) Otherwise, no pairing is added to \(L'.\)

For each \(a_i \in A\) for \(1 \leq i \leq s\), there exists the set \(A_i\) in \(G\) with \(\alpha_i\) vertices which are connected only to one set, \(\text{Bset}_i.\) So, the vertices of each \(A_i\) are matched to some vertices of \(\text{Bset}_i,\) that is \(b_{ji}\) for \(1 \leq j \leq t.\) Hence, each \(a_i \in A\) for \(1 \leq i \leq s\) is matched to at least \(\alpha_i\) elements of \(B,\) and the demand of \(a_i\) is satisfied. In \(G\) there exist \(\alpha_i\) plus \(\alpha'_i - \alpha_i\) copies of each element \(a_i,\) that is the vertices of \(A_i\) plus the vertices of \(A'_i.\) So, the number of elements that are matched to each \(a_i \in A\) is at most \(\alpha'_i.\)

Consider the sets \(B_j,\) with \(1 \leq j \leq t,\) recall that \(B_j = \{b_{ji} | 1 \leq i \leq s\}\) and the vertices of \(W_j\) are connected to \(B_j\) for \(1 \leq j \leq t\) by zero weighted edges. \(W_j\) is connected only to \(B_j,\) so the vertices of \(W_j\) are matched to \(s - \beta'_j\) number of the vertices of \(B_j,\) and \(\beta'_j\) number of the vertices remains unmatched in \(B_j.\) Suppose that \(k\) vertices of \(\beta'_j\) vertices in \(B_j\) are matched to the vertices of \(A_i\) sets for \(1 \leq i \leq s,\) so the \(\beta'_j - k\) remaining vertices of \(B_j\) should be matched to the other sets that are connected to it. We discuss two cases, depending on the value of \(k.\)

- if \(k < \beta_j\) then \((\beta'_j - k) > (\beta'_j - \beta_j).\) Then, \(X_j\) selects the \(\beta'_j - \beta_j\) vertices of the remaining vertices of \(B_j,\) we have
  \[(\beta'_j - k) - (\beta'_j - \beta_j) = \beta'_j - k - \beta'_j + \beta_j = \beta_j - k > 0,\]
  so the remaining \(\beta_j - k\) vertices of \(B_j\) are matched to the vertices of \(A'_i\) sets. Note that \(k\) vertices of the vertices \(b_{ji}\) for all \(1 \leq i \leq s\) are matched to the vertices of \(A_i\) sets and \(\beta_j - k\) vertices of them are matched to \(A'_i\) sets. The demand of the element \(b_j\) is satisfied, since
  \[\beta_j - k + k = \beta_j.\]

- if \(k > \beta_j\) then \((\beta'_j - k) < (\beta'_j - \beta_j)\) and all the \((\beta'_j - k)\) remaining members of \(B_j\) are matched to the vertices of \(X_j.\)

The cost of \(L'\) is equal to the weight of \(\text{Main}(M),\) i.e. \(c(L') = \text{Weight}(\text{Main}(M)),\) since for each edge of \(\text{Main}(M),\) we add a pairing with equal cost to \(L'.\) On the other hand, \(L'\) is an MMDC between \(A\) and \(B.\) \(L\) is a minimum-cost MMDC between \(A\) and \(B,\) so \(c(L) \leq c(L').\) Thus
\[c(L) \leq \text{Weight}(\text{Main}(M)).\]

\[\square\]

**Theorem 2.** Let \(M\) be a minimum weight perfect matching in \(G,\) and let \(L\) be a minimum-cost MMDC between \(A\) and \(B.\) Then, \(\text{Weight}(\text{Main}(M)) = c(L).\)
Proof. By Lemma 2 and Lemma 4, we have $\text{Weight}(\text{Main}(M)) \leq c(L)$ and $\text{Weight}(\text{Main}(M)) \geq c(L)$, respectively. So we have $\text{Weight}(\text{Main}(M)) = c(L)$.

Recall that the time complexity of the Basic Hungarian algorithm is $O(n^3)$, where the number of the vertices of the input graph is $O(n)$. The number of the vertices of our complete bipartite graph is $O(n^2)$, so the complexity of our algorithm is $O(n^6)$, but in bipartite graphs with low-range edge weights and dense graphs, our algorithm runs well [5].

Conclusion

We presented an algorithm for the minimum-cost MMDC problem by advantage of the Hungarian algorithm. It is expected that the complexity of the MMDC problem will be reduced by exploiting the geometric information; the one and two dimensional versions of this problem remain open.

Data Availability

No data were used to support this study.

Conflicts of Interest

There is no conflict of interest to declare.

References

[1] T B Eiter and H Mannila. “Distance measures for point sets and their computation”. *Acta Inform.*, vol. 34, 109–133, 1997.

[2] M Imanparast and S N Hashemi. “A linear time randomized approximation algorithm for euclidean matching”. *J. Supercomput.*, vol. 75, 2648–2664, 2019.

[3] H N Gabow. “An efficient reduction technique for degree-constrained subgraph and bidirected network flow problems”. In *Fifteenth Annual ACM Symposium on Theory of Computing*, pages 448–456, 1983.

[4] S. Bandyapadhyay, A. Maheshwari and M. Smid. “Exact and approximation algorithms for many-to-many point matching in the plane”. In *the 32nd International Symposium on Algorithms and Computation (ISAAC 2021), Fukuoka, Japan*, 2021.

[5] Paulo A.C. Lopes, Satyendra Singh Yadav, Aleksandar Ilic and Sarat Kumar Patra. “Fast block distributed CUDA implementation of the Hungarian algorithm”. *J. Parallel Distrib. Comput.*, vol. 130, 50–62, 2019.

[6] H W Kuhn. “The Hungarian method for the assignment problem”. *Nav. Res. Logist. Q.*, vol. 2, 83–97, 1955.

[7] J Munkres. “Algorithms for the assignment and transportation problems”. *J. Soc. Indust. Appl. Math*, vol. 5, 32–38, 1957.