A fast and efficient algorithm for many-to-many matching of points with demands in one dimension

Fatemeh Rajabi-Alni\textsuperscript{a}, Behrouz Minaei-Bidgoli\textsuperscript{a}, and Alireza Bagheri\textsuperscript{b}

\textsuperscript{a}Department of Computer Engineering, Iran University of Science and Technology, Tehran, Iran;
\textsuperscript{b}Computer Engineering Department, Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran

\textbf{ARTICLE HISTORY}
Compiled November 8, 2024

\textbf{ABSTRACT}
Given two point sets $S$ and $T$, the minimum-cost \textit{many-to-many matching with demands} (MMD) problem is the problem of finding a minimum-cost many-to-many matching between $S$ and $T$ such that each point of $S$ (respectively $T$) is matched to at least a given number of the points of $T$ (respectively $S$). We propose the first $O(n^2)$ time algorithm for computing a \textit{one dimensional MMD} (OMMD) of minimum cost between $S$ and $T$, where $|S| + |T| = n$. In an OMMD problem, the input point sets $S$ and $T$ lie on the real line and the cost of matching a point to another point equals the distance between the two points. We also study a generalized version of the MMD problem, the \textit{many-to-many matching with demands and capacities} (MMDC) problem, that in which each point has a limited capacity in addition to a demand. We give the first $O(n^2)$ time algorithm for the minimum-cost \textit{one dimensional MMDC} (OMMDC) problem.

\textbf{KEYWORDS}
Many-to-many point matching; One dimensional point-matching; Demands; Capacities

1. Introduction

Suppose we are given two point sets $S$ and $T$ with $|S| + |T| = n$, a \textit{many-to-many matching} (MM) between $S$ and $T$ assigns each point of one set to one or more points of the other set \cite{1}. The minimum-cost MM problem has been solved using the Hungarian method in $O(n^3)$ time \cite{2}. Colannino et al. \cite{11} presented an $O(n \log n)$ time dynamic programming solution for finding an MM of minimum-cost between two sets on the real line. For more discussion on the MM, see \cite{3, 4}.

A general case of the MM problem is the \textit{limited capacity many-to-many matching} (LCMM) problem where each point has a capacity, i.e. each point can be matched to at most a given number of the points. A special case of the LCMM problem, the \textit{one dimensional LCMM} (OLCMM) problem, is that in which both $S$ and $T$ lie on the real line. Rajabi-Alni and Bagheri \cite{5} proposed an $O(n^2)$ time algorithm for the minimum-cost OLCMM.

F. Rajabi-Alni. Email: f.rajabialni@aut.ac.ir; fatemehrajabialni@yahoo.com
B. Minaei-Bidgoli. Email: b_minaei@iust.ac.ir
A. Bagheri. Email: ar_bagheri@aut.ac.ir
In this paper, we consider another generalization of the MM problem, where each point has a demand, that is each point of one set must be matched to at least a given number of the points of the other set. Let $S = \{s_1, s_2, \ldots, s_y\}$ and $T = \{t_1, t_2, \ldots, t_z\}$. We denote the demand sets of $S$ and $T$ by $D_S = \{a_1, a_2, \ldots, a_y\}$ and $D_T = \{\beta_1, \beta_2, \ldots, \beta_z\}$, respectively. In a many-to-many matching with demand (MMD), each point $a_i \in S$ must be matched to at least $a_i$ points in $T$ and each point $t_j \in T$ must be matched to at least $\beta_j$ points in $S$. We study the one dimensional MMD (OMMD), where $S$ and $T$ lie on the real line and the cost of matching $s_i \in S$ to $t_j \in T$ equals their distance on the line for $1 \leq i \leq y$ and $1 \leq j \leq z$. We propose an $O(n^2)$ algorithm for finding a minimum-cost OMMD between $S$ and $T$. We also give an $O(n^2)$ time algorithm for a general version of the OMMD problem, the one dimensional many-to-many matching with demands and capacities (OMMDC) problem, where each point has a demand and a capacity.

One of the important motivations for the study of the OMMD and OMMDC problems is their applications in wireless networks. For example, consider a wireless network deployed along a one-dimensional line in applications such as environmental boundary monitoring and target tracking, or performing border patrol, or vehicular ad hoc network on highways [3–10]. As an example for the MMD problem, consider the target coverage problem where each target should be monitored by at least a given number of sensors [11].

Another important application of the MM problem is the data matching [12, 13].

2. Preliminaries

In this section, we proceed with some useful definitions and assumptions. Let $S = \{s_1, s_2, \ldots, s_y\}$ and $T = \{t_1, t_2, \ldots, t_z\}$ be two point sets with $|S| + |T| = n$. Assume $D_S = \{a_1, a_2, \ldots, a_y\}$ and $D_T = \{\beta_1, \beta_2, \ldots, \beta_z\}$ be the demand sets of $S$ and $T$, respectively. We denote the points in $S$ in increasing order by $(s_1, \ldots, s_y)$, and the points in $T$ in increasing order by $(t_1, \ldots, t_z)$. Let $S \cup T$ be partitioned into maximal subsets $A_0, A_1, A_2, \ldots$ alternating between subsets in $S$ and $T$ such that the largest point of $A_i$ lies to the left of the smallest point of $A_{i+1}$ for all $i \geq 0$ (see Figure 1).

W.l.o.g. we assume that all points $p \in S \cup T$ are distinct.

Let $A_w = \{a_1, a_2, \ldots, a_w\}$ with $a_1 < a_2 < \cdots < a_w$ and $A_{w+1} = \{b_1, b_2, \ldots, b_t\}$ with $b_1 < b_2 < \cdots < b_t$. For $w > 0$, $b_0$ or $a_0$ represents the largest point of $A_{w-1}$ (see Figure 2). We denote the demand of each point $a \in S \cup T$ by $\text{Demand}(a)$. The cost of matching each point $a \in S$ to a point $b \in T$ is considered as $|a - b|$. For any point $q$, let $C(q, j)$ be the cost of a minimum-cost OMMD between the points $\{p \in S \cup T | p \leq q\}$ such that all the demands of each point $p$ with $p < q$ are satisfied, but $j$ number of the demands of $q$ are satisfied. So, $C(q, 0)$ is the cost of a minimum-cost OMMD between the points $\{p \in S \cup T | p \leq q\}$ such that $\text{deg}(p) \geq \text{Demand}(p)$ for all $p < q$ and $\text{deg}(q) = 0$. Thus, $C(b, 0) = C(b_{i-1}, \text{deg}(b_{i-1}))$ for $i > 1$ and $C(b_1, 0) = C(a_s, \text{deg}(a_s))$. Note that $\text{deg}(p)$ denotes the number of the points that has been matched to $p$ in the OMMD. Let $M(b_i, k)$ denote the point that satisfies the $k$th demand of $b_i$. Note that if $k > \text{Demand}(b_i)$, we suppose that $M(b_i, k)$ is the $k$th point that is matched to $b_i$. 


3. Our algorithms

In this section, we first present an $O(n^2)$ time algorithm based on the algorithm of [14] for finding a minimum-cost OMMD between two sets $S$ and $T$ lying on the real line. Then, we generalize our algorithm for computing an OMMDC between $S$ and $T$.

3.1. The OMMD algorithm

We first begin with some useful lemmas.

**Lemma 3.1.** Let $b < c$ be two points in $S$, and $a < d$ be two points in $T$ such that $a \leq b < c \leq d$. If a minimum-cost OMMD, denoted by $M$, contains both of $(a, c)$ and $(b, d)$, then $(a, b) \in M$ or $(c, d) \in M$.

**Proof.** Suppose that the lemma is false, and $M$ contains both $(a, c)$ and $(b, d)$, but neither $(a, b) \in M$ nor $(c, d) \in M$ (see Figure 3). Then, we can remove the pairs $(a, c)$ and $(b, d)$ from $M$ and add the pairs $(a, b)$ and $(c, d)$: the result $M'$ is still an OMMD which has a smaller cost, a contradiction. \qed

**Figure 3.** $(a, c)$ and $(b, d)$ do not both belong to an optimal matching.

3
Lemma 3.2. Let \( a \in S, b \in T \) and \( c \in S, d \in T \) such that \( a \leq b < c \leq d \). Let \( M \) be a minimum-cost OMMD. If \((a, d) \in M\), then either \((a, b) \in M\) or \((c, d) \in M\) or both [5].

Corollary 3.3. Let \( a \in A_i \) and \( d \in A_j \) for some \( i, j \geq 0 \). For any matching \((a, d)\) in a minimum-cost OMMD, if \( j > i + 1 \), then either \((a, b) \in M\) for all points \( b \in A_{i+1} \cup A_{i+3} \cup \cdots \cup A_{j-2} \) or \((c, d) \in M\) for all points \( c \in A_{i+2} \cup A_{i+4} \cup \cdots \cup A_{j-1} \) [5].

Note that we use Corollary 3.3 for satisfying the demands of \( a \in A_i \) by the points of the sets \( A_{i+1}, A_{i+3}, \ldots, A_j \) or for satisfying the demands of \( d \in A_j \) by the points of the sets \( A_{j-1}, A_{j-3}, \ldots, A_i \).

Theorem 3.4. Let \( S \) and \( T \) be two sets of points on the real line with \(|S| + |T| = n\). Then, a minimum-cost OMMD between \( S \) and \( T \) can be determined in \( O(n^2) \) time.

Proof. Our algorithm is as follows (see Algorithm 1). Initially, partition \( S \cup T \) into maximal subsets \( A_0, A_1, \ldots \) and let \( C(p, k) = \infty \) for all \( p \in S \cup T \) and \( 1 \leq k \leq n \) (Lines 1–5). Obviously, the number of the pairs of a minimum-cost OMMD is equal to \( \max(\sum_{i=1}^{\mid S \mid} \alpha_i, \sum_{j=1}^{\mid T \mid} \beta_j) \). Note that we assume there exists at least one OMMD between \( S \) and \( T \).

Let \( A_w = \{a_1, a_2, \ldots, a_s\} \) and \( A_{w+1} = \{b_1, b_2, \ldots, b_t\} \) (Line 8). Assume that we have computed \( C(p, h) \) for all \( p < b_1 \) and \( 1 \leq h \leq n \) including \( C(a_s, \deg(a_s)) \), i.e. the cost of an OMMD for the points \( \{p \in S \cup T \mid p \leq a_s\} \). And now we want to compute \( C(b_i, k) \) for all \( 1 \leq i \leq t \) and \( 1 \leq k \leq k' \), where \( \text{Demand}(b_1) \leq k' \leq n \) (see Figure 2). In fact, we use the idea of [14], that is we first examine the point \( b_1 \in A_{w+1} \) and determine that whether matching the points \( p \leq b_1 \) to the point \( b_1 \) decreases the cost of the OMMD or not. Then, if \( \deg(b_1) < \text{Demand}(b_1) \), we should satisfy the remaining demands of \( b_1 \) such that the cost of the OMMD is minimized. In other words, we compute \( C(b_1, k) \) for \( k = 1, 2, \ldots, k' \) for \( \text{Demand}(b_1) \leq k' \), respectively. Then, we examine the point \( b_2 \) and compute \( C(b_2, k) \) for \( k = 1, 2, \ldots, k' \) for \( \text{Demand}(b_2) \leq k' \), respectively, and so on. So, our algorithm is in two steps (Lines 7–23):

Step 1. In this step, by Corollary 3.3, we should determine that whether matching the points \( p \leq b_1 \) with \( \deg(p) \leq \text{Demand}(p) \) to \( b_1 \) decreases the cost of the OMMD or not (Line 11 of Algorithm 1). We distinguish two cases:

\( \circ \) \( \deg(p) = \text{Demand}(p) \) and \( p \) has been matched to at least a point \( a' \leq p \).

Let \( q \) denote the number of points \( a' \leq p \) that have been matched to \( p \). If we have \( C(p, q) > C(p, q - 1) + b_1 - p \), we remove the pair \((p, M(p, q))\) from the OMMD and add the pair \((p, b_1)\) to it.

\( \circ \) \( \deg(p) < \text{Demand}(p) \). Then, we add the pair \((p, b_1)\) to the OMMD.

Let \( \text{list}(w) \) denote a doubly linked list of the points \( p \in A_w \) which is initially empty (Line 9 of Algorithm 1). We construct \( \text{list}(w) \) such that \( \text{Match}(b_i, w) \) denote the pointer to a sublist of the elements in \( \text{list}(w) \) (with the first element \( \text{list}(w), \text{first} \)) to which \( b_i \) is matched. Observe that \( \text{Match}(b_i, w) \) is a doubly linked list of the points of \( A_w \) that are matched to \( b_i \). Initially, \( \text{Match}(b_i, w) = \text{null} \) (Line 1 of Algorithm 2).

Firstly, if we have \( i = 1 \), starting from \( a_s \), we examine the points of \( A_w \), i.e., \( a_s, a_{s-1}, \ldots, a_1 \), respectively to check whether matching them to \( b_1 \) decreases the cost of the OMMD or not until reaching the point \( a_0 \) (Lines 2–16 of Algorithm 2; see Figure 2). Observe that when the point \( a_j \) is matched to \( b_1 \), we add \( a_j \) to the end of \( \text{list}(w) \) (Lines 9 and 14 of Algorithm 2). Finally, we update \( \text{Match}(b_1, w) \).
Step 2. This step consists of two sub steps, Step 2.1 and Step 2.2, running iteratively (Line 16 of Algorithm 2). Note that \( \text{list}(w).first \) and \( \text{list}(w).end \) are pointers to the first and last elements of \( \text{list}(w) \), respectively.

Note that if matching the point \( p \leq b_{i-1} \) to \( b_{i-1} \) does not decrease the cost of the OMMD, then matching \( p \) to \( b_i \) does not decrease the cost of the OMMD, too. Thus, for \( i \geq 1 \) we do as follows. Let \( \text{tempset} \) denote the set of the indices of the partitions \( A_{w'} \) that some of whose points are matched to \( b_{i-1} \), that is \( \text{tempset} = \{ w' : M(b_{i-1}, k) \in A_{w'} \} \), in the descending order (Line 1 of Algorithm 3). We should determine that whether matching the points \( M(b_{i-1}, k) \) to \( b_i \) decreases the cost of the OMMD or not. Thus, starting from the largest index \( w' \), we search the doubly linked lists \( \text{Match}(b_{i-1}, w') \), with the first element \( \text{list}(w').first \) and the last element \( \text{Match}(b_{i-1}, w') \), for all \( w' \in \text{tempset} \) (Lines 20–36 of Algorithm 3).

We construct two linked lists \( \text{templist1} \) and \( \text{templist2} \) of the points \( a_j \in A_{w'} \) that matching which to \( b_i \) decreases the cost of the OMMD (Lines 27 and 33 of Algorithm 3) and does not decrease (Lines 29 and 35 of Algorithm 3), respectively. Then, we concatenate \( \text{templist1} \) and \( \text{templist2} \) using the function \( \text{Concatenate}(\text{templist2}, \text{templist1}) \) to get a linked list \( \text{templist3} \) such that \( \text{templist1}.end.next = \text{templist2}.first \) (Line 37 of Algorithm 3). And, we also copy the values of the elements of \( \text{templist3} \) into the elements of \( \text{Match}(b_{i-1}, w') \) (Line 39 of Algorithm 3). Assume that \( k' = \text{num}(\text{templist1}) \) denotes the number of the elements of \( \text{templist1} \) (Line 38 of Algorithm 3). Finally, we set \( \text{Match}(b_i, w'') \) to points to the \( k' \)th element of \( \text{list}(w'') \) which is denoted by \( \text{list}(w'', k') \) (Line 40 of Algorithm 3).

Note that this step can be considered as a preprocessing step for each point \( b_i \in A_{w+1} \), and has the time complexity \( O(n) \). Since the number of the points \( p \leq b_i \) in \( S \cup T \) is at most \( O(n) \). Then, if \( \text{deg}(b_i) \geq \text{Demand}(b_i) \) we have done, otherwise we go to Step 2. Note that if \( \text{deg}(b_i) > \text{Demand}(b_i) \), we add \( b_i \) to the linked list \( \text{LL}(w + 1) \) (Lines 12–13 of Algorithm 1). More details will be given about \( \text{LL}(w + 1) \) when we discuss Step 2.

Step 2. This step consists of two sub steps, Step 2.1 and Step 2.2, running iteratively until \( \text{deg}(b_i) > \text{Demand}(b_i) \) or \( w' \leq 0 \) (Lines 16–22 of Algorithm 1). Note that initially \( w' = w + 1 \) (Line 15 of Algorithm 1).

Step 2.1 In this step, we should check the points \( p \in A_{w} \) with \( p < b_i \) in the descending order to find the first (i.e. the largest) point \( b'_j \in A_{w'} \) with \( \text{deg}(b'_j) > \text{Demand}(b'_j) \), which is maintained in the list \( \text{LL}(w') \) (Lines 12–13 of Algorithm 1). Note that \( \text{LL} \) is a set of lists, one list for each partition \( A_j \) for \( j \geq 1 \).

Let \( \text{templist} = \text{LL}(w') \) and \( \text{tempset} = \emptyset \) (Line 1 of Algorithm 4). While \( \text{templist} \neq \emptyset \) and \( \text{deg}(b_i) < \text{Demand}(b_i) \), we search \( A_{w'} \) as follows (Lines 2–18 of Algorithm 4). Firstly, if \( \text{tempset} = \emptyset \), we remove the last (the largest) point of \( \text{templist} \) denoted by \( b'_j \) (Line 4 of Algorithm 4). Let \( \text{tempset} = \{ w'' : M(b'_j, k) \in A_{w'} \} \). Then, we search the doubly linked lists \( \text{Match}(b'_j, w'') \) for the first element \( w'' \) from \( \text{tempset} \) (Line 6 of Algorithm 4). Let \( \text{current} = \text{Match}(b'_j, w'') \) (Line 7 of Algorithm 4). While \( \text{current} = \text{Match}(b'_j, w'') \) (Line 7 of Algorithm 4). While
- \( \text{Match}(b_i, w'') \neq \text{current} \),
- and \( \text{deg}(b_i) < \text{Demand}(b_i) \),
- and \( \text{deg}(b'_j) > \text{Demand}(b'_j) \),

starting from \( \text{current} \), we seek \( \text{list}(w'') \) as follows (Lines 8–17 of Algorithm 4).
Let \( p' \) be the value of the element whose pointer is \textit{current} in \textit{list}(\textit{w''}) (Line 9 of Algorithm 4). We should match \( b_j \) to \( p' \) and remove \((b_j', p')\) from the OMMD (Lines 10–12 of Algorithm 4). Since otherwise, \( b_i \) would be matched to a point \( p \) for which one of the following statements holds:

- either \( p \in \{M(b_j', v)\}_{v=1}^{\deg(b_j')}\) (see Figure 4). Then, if we remove the pair \((p, b_j')\) from the OMMD, we get an OMMD with a smaller cost.

- or \( p \notin \{M(b_j', v)\}_{v=1}^{\deg(b_j')}\) (see Figure 5). Then, two cases arise:
  * either \( \deg(p) = \text{demand}(p) \). Note that \( \deg(b_j') > \text{Demand}(b_j') \) implies that matching the point \( M(b_j', v) \) to \( b_j' \) for \( v \in \{1, 2, \ldots, \deg(b_j')\} \) decreases the cost of the OMMD, and matching \( p \) to \( b_j' \) does not decrease the cost of the OMMD. Therefore, we have:
    \[
    C(p, v') < b_j' - p + C(p, v' - 1),
    \]
    and
    \[
    0 < b_j' - p + C(p, v' - 1),
    \]
    for all \( 1 \leq v' \leq \deg(p) \). Therefore,
    \[
    b_i - b_j' < b_i - b_j' + b_j' - p + C(p, v' - 1).
    \]
    Thus, we have
    \[
    C(b_i, k - 1) + b_i - b_j' < C(b_i, k - 1) + b_i - p + C(p, v' - 1),
    \]
    which means if we add \((M(b_j', v), b_i)\) to the OMMD and remove \((M(b_j', v), b_j')\) for an arbitrary \( v \in \{1, 2, \ldots, \deg(b_j')\} \), say \( v'' \), we get a smaller cost. We assume w.l.o.g that \( M(b_j', v'') = \min_{v=1}^{\deg(b_j')} M(b_j, v) \).
  * or \( \deg(p) > \text{demand}(p) \). This case implies that
    \[
    b_i - p < b_i - b_j',
    \]
    contradicting \( p < b_j' \).

Then, we update \textit{current} (Line 13 of Algorithm 4). Observe that if \( \deg(b_j') = \text{Demand}(b_j') \), we remove \( b_j' \) from \textit{LL}(\textit{w'}) and let \textit{tempset} = \emptyset (Lines 15–17 of Algorithm 4).

If there does not exist such a point \( b_j' \leq b_i \) with \( \deg(b_j') > \text{Demand}(b_j') \), by the following claim, we check whether the points \( b_j' \in A_{\textit{w'}} \) have been matched to any points \( p \) with \( \deg(p) > \text{Demand}(p) \), which are maintained in the list link \( LL2(\textit{w'}) \) (Lines 19–25 of Algorithm 4). Note that \( LL2(\textit{w'}) \) is a linked list of the points of \( \{M(b_j', v) : \deg(M(b_j', v)) > \text{Demand}(M(b_j', v))\}_{j=1}^{\text{Demand}(b_j')} \) for \( 1 \leq j \leq t \), in decreasing order (Lines 12–13 of Algorithm 5). Therefor:

\[
\alpha_h' = \max(\{M(b_j', v) : \deg(M(b_j', v)) > \text{Demand}(M(b_j', v))\}_{v=1}^{\text{Demand}(b_j')}).
\]
Then, if \( a'_h \) has not been matched to \( b_i \), we match \( b_i \) to \( a'_h \) (Lines 23–25 of Algorithm 4). It is easy to show that thank the link lists \( LL(w') \) and \( LL2(w') \), Step 2.1 runs in \( O(n) \) time for each \( b_i \in A_{w+1} \).

**Claim 1.** Let \( a'_h \) be the largest point with \( \text{deg}(a'_h) > \text{Demand}(a'_h) \) that has been matched to \( b'_j \) in \( A_{w'} \), then \( b_i \) would also be matched to \( a'_h \).

**Proof.** Note that \( \text{deg}(a'_h) > \text{Demand}(a'_h) \) implies that \( a'_h \) decreases the cost of the OMMD. Also note that \( ||b'_j - a|| < ||b - a|| \) for all points \( a \in S \cup T \) with \( a \leq b'_j \). In the minimum-cost OMMD, \( b'_j \) has been matched to \( a'_h = M(b'_j, v) \) for \( v \in \{1, 2, \ldots, \text{Demand}(b'_j)\} \) instead of any other point \( a'_q \) with \( \text{deg}(a'_q) \geq \text{Demand}(a'_q) \) (see Figure 5), so:

\[
C(b'_j, v - 1) + b'_j - a'_h < C(b'_j, v - 1) + b'_j - a'_q + \min(-C(a'_q, \text{deg}(a'_q)) + C(a'_q, \text{deg}(a'_q) - 1), 0),
\]

and thus:

\[-a'_h < -a'_q + \min(-C(a'_q, \text{deg}(a'_q)) + C(a'_q, \text{deg}(a'_q) - 1), 0),\]

If we add \( C(b_i, k - 1) \) and \( b_i \) to both sides of the above inequality, then we have:

\[
C(b_i, k - 1) + b_i - a'_h < C(b_i, k - 1) + b_i - a'_q + \min(-C(a'_q, \text{deg}(a'_q)) + C(a'_q, \text{deg}(a'_q) - 1), 0),
\]

so \( b_i \) is also matched to \( a'_h \). \( \square \)

![Figure 4](image-url) An example for the case where \( b_i \) has been matched to a point \( p \in \{M(b_j, v)\}_{v=1}^{\text{deg}(b_j)} \).

If \( \text{deg}(b_i) \geq \text{Demand}(b_i) \) we have done. Otherwise, we let \( w' = w' - 1 \) and go to Step 2.2 (Lines 18–20 of Algorithm 4).

**Step 2.2** In this step, by Corollary 3.3, we seek the partition \( A'_w \) to satisfy the demands of \( b_i \) (Algorithm 5; see Figure 2). Let \( A_{w'} \neq \emptyset \) be the set of the points in \( A_{w'} \) that have been matched to \( u \geq 0 \) smaller points. Let \( R \) denote
the set of the numbers $u \geq 0$ with $A_{w'u} \neq \emptyset$ and $a''_u$ be the largest point of $A_{w'u}$.

Claim 2. Assume that $a'_l, a'_m \in A_{w'u}$ with $a'_m < a'_l$. In this step, $b_i$ can be matched to $a'_l$ but not to $a'_m$.

Proof. Suppose by contradiction that the point $b_i$ is matched to $a'_m$ (the pair $(M(a'_m, u), a'_m)$ might be removed from the OMMD and the pair $(a'_m, b_i)$ is added to the OMMD). Then, it is easy to show that there might exist at least one point $b' \leq a'_m$ such that $(b', a'_l) \in OMMD$ but $(b', a'_m) \notin OMMD$; by Lemma 3.1 two pairs $(b', a'_l)$ and $(a'_m, b_i)$ contradict the optimality (see Figure 6).

Thus, by Claim 2, we determine that $b_i$ should be matched to which of the points $a''_u$ for $u \in R$. Note that if we match $a''_u$ to $b_i$, we must check that whether the OMMD still includes the pair $(M(a''_u, u), a''_u)$ or not, i.e. it is possible that after matching $a''_u$ to $b_i$, the pair $(M(a''_u, u), a''_u)$ is removed form the OMMD.

Let $u(p)$ be the number of the points $M(p, h)$ with $M(p, h) < p$, i.e.,

$$u(p) = |\{M(p, h) : M(p, h) < p\}_{h=1}^{deg(p)}|,$$

for $p \in A_{w'}$ (Lines 1–2 of Algorithm 5). Assume $\text{templist}$ denotes the list of the points $p \in A_{w'}$ in the ascending order of

$$C(b_i, k - 1) + b_i - p + \min(-C(p, u(p)) + C(p, u(p) - 1), 0),$$

i.e., the cost of matching $b_i$ to $p$ (Line 4 of Algorithm 5). Let $\text{templist} = \text{templist} - \{p \in A_{w'} : (p, b_i) \in OMMD\}$ (Line 5 of Algorithm 5). While $\text{templist} \neq \emptyset$ and $deg(b_i) < Demand(b_i)$, we remove the first point of $\text{templist}$ denoted by $p'$ (the point that matching $b_i$ to it has the smallest cost), and match $b_i$ to $p'$ (Lines 8–9 of Algorithm 5). Then, we remove the pair $(M(p', u(p')), p')$ from the OMMD, if it is necessary (Lines 10–11 of
Algorithm 5. Now, if $\deg(p') > \text{Demand}(p')$, we add $p'$ to the end of the linked list $LL2(w')$ which later might be used in Step 2.1 (Lines 12–13 of Algorithm 5).

This step runs in $O(n)$ time, since the number of the points $p \leq b_i$ is at most $n$. □

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure6.png}
\caption{An example for illustration of proof of Claim 2.}
\end{figure}

### 3.2. The OMMDC algorithm

Given two sets of points $S$ and $T$ with $|S| + |T| = n$ on the real line, in this section, we provide an algorithm which is based on the OMMD algorithm for finding a minimum-cost OMMDC between $S$ and $T$ in $O(n^2)$ time. In an OMMDC, we should consider the limited capacities of points in addition to their demands thanks to the algorithm given in [14]. The correctness of the OMMDC algorithm follows from the proof of correctness of the OMMD algorithm, since these two algorithms are almost identical except for a very slight modification for the limited capacities of points.

**Theorem 3.5.** Let $S$ and $T$ be two sets of points on the real line with $|S| + |T| = n$. Then, a minimum-cost OMMDC between $S$ and $T$ can be computed in $O(n^2)$ time.

**Proof.** We use the same notations as for Algorithm [1]. Suppose that we have computed $C(p, h)$ for all $\{p \in S \cup T | p < b_i\} \text{ and } 1 \leq h \leq \text{Cap}(p)$, where $\text{Cap}(p)$ is the capacity of $p$, i.e. the number of points that can be matched to $p$. Now, we compute $C(b_i, k)$ for all $1 \leq k \leq \text{Cap}(b_i)$ as follows. There exist two steps:

Step 1. We first check that whether matching the points $\{p \in S \cup T | p \leq b_i\}$ to $b_i$ decreases the cost of the matching or not as Step 1 of the OMMD algorithm until $\deg(b_i) = \text{Cap}(b_i)$:

- If $i = 1$, we must determine that whether matching the points $a_s, \ldots, a_1$ to $b_i$ decreases the cost of the OMMDC or not.
Algorithm 1: OMMD Algorithm($S,T,D_S,D_T$)

1. Partition $S \cup T$ to $A_0, A_1, \ldots$;
2. Let $n = |S \cup T|$;
3. forall $p \in S \cup T$ do
   4. forall $1 \leq k \leq n$ do
      5. $C(p, k) = \infty$;
6. Let $w = 0$ and $OMMD = \emptyset$;
7. while $|OMMD| < \max(\sum_{i=1}^{|S|} \alpha_i, \sum_{j=1}^{|T|} \beta_j)$ do
   8. Let $A_w = \{a_1, a_2, \ldots, a_s\}$, and $A_{w+1} = \{b_1, b_2, \ldots, b_t\}$;
   9. Let list($w$) be an empty linked list;
10. for $i = 1$ to $t$ do
   11.   Step1($S, T, A_0, A_1, \ldots$);
12.      if deg($b_i$) > Demand($b_i$) then
13.         Add $b_i$ to the end of the linked list LL($w + 1$);
14.      for $i = 1$ to $t$ do
15.         Let $w' = w + 1$;
16.         while deg($b_i$) < Demand($b_i$) and $w' \geq 1$ do
17.            Step 2.1($S, T, A_0, A_1, \ldots, w', b_i$);
18.            if deg($b_i$) < Demand($b_i$) then
19.               Let $w' = w' - 1$;
20.            Step 2.2($S, T, A_0, A_1, \ldots, w', b_i$);
21.            if deg($b_i$) < Demand($b_i$) then
22.               Let $w' = w' - 1$;
23.         $w = w + 1$;

Algorithm 2: Step1:PartI($S, T, A_0, A_1, \ldots$)

1. Let $Match(b_i, w) = null$;
2. if $i = 1$ then
3.   $j = s$;
4. while $j \geq 1$ do
5.   Let $q$ be the number of the points of $\{M(a_j, h) : M(a_j, h) \leq a_j\}_{h=1}^{deg(a_i)}$;
6.   if deg($a_j$) = Demand($a_j$) and $q \neq 0$ then
7.      if $C(a_j, q) > C(a_j, q - 1) + b_1 - a_j$ then
8.         Add the pair $(b_1, a_j)$ to OMMD and remove $(a_j, M(a_j, q))$;
9.         Add $a_j$ to the end of list($w$);
10.        $C(b_1, deg(b_1)) = C(b_1, deg(b_1) - 1) + b_1 - a_j - C(a_j, q) + C(a_j, q - 1)$;
11.      else if deg($a_j$) < Demand($a_j$) then
12.         Add the pair $(b_1, a_j)$ to OMMD;
13.         $C(b_1, deg(b_1)) = C(b_1, deg(b_1) - 1) + b_1 - a_j$;
14.         Add $a_j$ to the end of list($w$);
15.      $j = j - 1$;
16.    Let $Match(b_i, w) = list(w).end$;
Algorithm 3: Step1:PartII($S, T, A_0, A_1, \ldots$)

Let tempset denote the set \{w' : $M(b_{i-1}, k) \in A_{w'}^{\deg(b_{i-1})}$\} in the descending order;

while tempset $\neq \emptyset$ do

Remove the first (largest) index $w''$ from the set tempset;

Let current = list($w''$).first;

while current $\neq$ Match($b_{i-1}, w''$).prev do

Let $a'_j = current$.val;

Let $q$ be the number of the points of \{$M(a'_j, h) : M(a'_j, h) \leq a'_j \}^{\deg(a'_j)}$;

if $\deg(a'_j) = \text{Demand}(a'_j)$ and $q \neq 0$ then

if $C(a'_j, q) > C(a'_j, q - 1) + b_i - a'_j$ then

Add the pair ($b_i, a'_j$) to OMMD and remove ($a'_j, M(a'_j, q)$);

$C(b_i, \deg(b_i)) = C(b_i, \deg(b_i) - 1) + b_i - a'_j - C(a'_j, q) + C(a'_j, q - 1)$;

Add $a'_j$ to the end of templist1;

else

Add $a'_j$ to the end of templist2;

else if $\deg(a'_j) < \text{Demand}(a'_j)$ then

Add the pair ($b_i, a'_j$) to OMMD;

$C(b_i, \deg(b_i)) = C(b_i, \deg(b_i) - 1) + b_i - a'_j$;

Add $a'_j$ to the end of templist1;

else

Add $a'_j$ to the end of templist2;

end if

current = current.next;

end while

templist3 = Concatenate(templist2, templist1);

$k' = \text{num}(\text{templist1})$;

Copy the values of templist3 into Match($b_{i-1}, w''$);

$\text{Match}(b_i, w'') = \text{list}(w'', k')$;
Algorithm 4: Step2.1($S, T, A_0, A_1, \ldots, w', b_i$)

1. Let $\text{templist} = LL(w')$ and $\text{tempset} = \emptyset$;
2. while $\text{templist} \neq \emptyset$ and $\text{deg}(b_i) < \text{Demand}(b_i)$ do
   3. if $\text{templist} = \emptyset$ then
      4. Remove the last point of $\text{templist}$ denoted by $b_j$;
      5. Let $\text{tempset}$ denote the set \{ $w^{''} : M(b_j', k) \in A_{w^{''}}$ \}$_{k=1}^{\text{deg}(b_j')}$ in the ascending order;
   6. Remove the first (smallest) index $w^{''}$ from the set $\text{tempset}$;
   7. Let $\text{current} = \text{Match}(b_j', w^{''})$;
   8. while $\text{Match}(b_i, w^{''}) \neq \text{current}$ and $\text{deg}(b_i) < \text{Demand}(b_i)$ and $\text{deg}(b_j') > \text{Demand}(b_j')$ do
      9. Let $p' = \text{current}.\text{val}$ and assume $p' = M(b_j', k')$;
      10. if $k' \neq \text{deg}(b_j')$ then
      11. Let $M(b_j', k') = M(b_j', \text{deg}(b_j'))$ and $M(b_j', \text{deg}(b_j')) = p'$;
      12. Match $b_i$ to $p'$ and remove ($b_j', p'$) from the OMMD;
      13. Let $\text{current} = \text{current}.\text{prev}$;
      14. Let $k = \text{deg}(b_i)$ and $C(b_i, k) = C(b_i, k - 1) + b_i - b_j'$;
      15. if $\text{deg}(b_j') = \text{Demand}(b_j')$ then
          16. Remove $b_j'$ from $LL(w')$;
          17. Let $\text{tempset} = \emptyset$;
      18. Let $\text{Match}(b_i, w) = \text{Match}(b_j', w^{''})$ and $\text{Match}(b_j', w^{''}) = \text{current}$;
   19. Let $\text{templist} = LL2(w')$;
20. while $\text{templist} \neq \emptyset$ and $\text{deg}(b_i) < \text{Demand}(b_i)$ do
21. Let $k = \text{deg}(b_i) + 1$;
22. Remove the first point of $\text{templist}$ called $a_h'$;
23. if ($a_h', b_i$) $\notin$ OMMD then
      24. Match $b_i$ to the point $a_h'$;
      25. $C(b_i, k) = C(b_i, k - 1) + b_i - a_h'$;
Algorithm 5: Step2.2($S, T, A_0, A_1, \ldots, w', b_i$)

1 for $p \in A_{w'}$ do
2   Let $u(p) = |\{M(p, h) : M(p, h) < p\}^{deg(p)}|$;
3   Let $k = deg(b_i) + 1$;
4   Let $templist$ be the list of the points $p \in A_{w'}$ in the ascending order of
   $C(b_i, k - 1) + b_i - p + \min(-C(p, u(p)) + C(p, u(p) - 1), 0)$;
5   Let $templist = templist - \{p \in A_{w'} : (p, b_i) \in OMMD\}$;
6 while $deg(b_i) < Demand(b_i)$ and $templist \neq \text{null}$ do
7   Let $k = deg(b_i) + 1$;
8   Remove the first point of $templist$ denoted by $p'$;
9   Match $b_i$ to the point $p'$;
10  if $C(p', u(p')) > C(p', u(p') - 1)$ then
11    Remove $(M(p', u(p')), p')$ from $OMMD$;
12  if $deg(p') > Demand(p')$ then
13    Add $p'$ to the end of the linked list $LL2(w')$;

Let $\{a'_1, a'_2, \ldots, a'_s\}$ be the set of points matched to $b_{i-1}$. We also check
that whether matching the points $a'_1, a'_2, \ldots, a'_s$ to $b_i$ decreases the cost of
the OMMDC.

If $deg(b_{i-1}) = Cap(b_{i-1})$, starting from the smallest point that has been
matched to $b_{i-1}$, i.e. $M(b_{i-1}, Cap(b_{i-1}))$, we should determine that whether
matching the points $p \leq M(b_{i-1}, Cap(b_{i-1}))$ to $b_i$ decreases the cost of
the OMMDC or not (see Figure 7). This case follows from the limited capacities
of the points $b_1, b_2, \ldots, b_{i-1}$ (see the OLCMM algorithm given in [14]).

Figure 7. In Step 1, we check that whether matching the points $p \leq M(b_{i-1}, Cap(b_{i-1}))$ to $b_i$ decreases the
cost of the OMMDC.

Let $w' = w + 1$. Now, while $deg(b_i) < Demand(b_i)$ and $w' \geq 1$, two following
sub steps run, iteratively:

Step 2.1 In this step, while $deg(b_i) < Demand(b_i)$, we do as follows:

○ If there exists a point $b'_j \in A_w$ with $b'_j \leq b_i$ and $deg(b'_j) > Demand(b'_j)$, we
remove the pair \((p', b_j')\) from the OMMDC and add the pair \((p', b_i)\), where 
p' = M(b_j', \text{deg}(b_j')).

- Otherwise, if there exists a point \(b_j'\) in \(A_w\) that has been matched to a point 
  \(a_h'\) with \(\text{Demand}(a_h') < \text{deg}(a_h') < \text{Cap}(a_h')\), by Claim\(^1\) we must match \(b_i\) to \(a_h'\).

  If still \(\text{deg}(b_i) < \text{Demand}(b_i)\), we let \(w' = w' - 1\) and go to Step 2.2.

Step 2.2 This step is as Step 2.2 of the OMMD algorithm.

4. Conclusions and Future Research

In this paper, we proposed an \(O(n^2)\) time algorithm for computing a one dimensional many-to-many matching with demands (OMMD), i.e. a many-to-many matching between two point sets on a line where each point is matched to at least a given number of the points of the other set. We also briefly described an \(O(n^2)\) time algorithm for a generalized version of the OMMD problem, OMMDC problem, where each point has a limited capacity (the number of the points that can be matched to each point is a limited number). Our algorithms are the first \(O(n^2)\) time algorithms for the OMMD and OMMDC problems. As a future research, we can solve the 2 dimensional version of the MMD problem. We also can develop new algorithms for other versions of the OMMD and OMMDC problems depending the real world properties.

References

[1] J. Colannino, M. Damian, F. Hurtado, S. Langerman, H. Meijer, S. Ramaswami, D. Souvaine, G. Toussaint, Efficient many-to-many point matching in one dimension, Graph. Combinator. 23 (2007) 169–178.
[2] T. B. Eiter, H. Mannila, Distance measures for point sets and their computation, Acta Inform. 34 (1997) 109–133.
[3] M. Imanparast, S. N. Hashemi, A linear time randomized approximation algorithm for Euclidean matching, J. Supercomput. 75 (2019) 2648–2664.
[4] F. Rajabi-Alni, A. Bagheri, Computing a many-to-many matching with demands and capacities between two sets using the hungarian algorithm, J. Math. (2023). doi:10.1155/2023/7761902
[5] F. Rajabi-Alni, A. Bagheri, An \(O(n^2)\) algorithm for the limited-capacity many-to-many point matching in one dimension, Algorithmica 76(2) (2016) 381–400.
[6] S. Susca, F. Bullo, S. Martinez, Monitoring environmental boundaries with a robotic sensor network, IEEE Trans. Control Syst. Technol. 16(2) (2008) 288–296.
[7] P. Frasca, F. Garin, B. Gerencsér, J. M. Hendrickx, Optimal one-dimensional coverage by unreliable sensors, SIAM J. Control Optim. 53(5) (2015) 3120–3140.
[8] L. Li, B. Zhang, J. Zheng, A study on one-dimensional k-coverage problem in wireless sensor networks, Wirel. Commun. Mob. Comput. 13 (2013) 1–11.
[9] D. Hu, J. Wu, P. Fan, On the outage probability of interference-limited multi-hop linear vehicular ad-hoc network, IEEE Access 6 (2018) 75395–75406.
[10] A. Aleksandrov, A. Fradkov, A. Semenov, Delayed and switched control of formations on a line segment: delays and switches do not matter, IEEE Trans. Autom. Control 65(2) (2020) 794–800.
[11] R. Yarinezhad, S. N. Hashemi, A sensor deployment approach for target coverage problem
in wireless sensor networks, J. Ambient Intell. Hum. Comput. (2020). doi:10.1007/s12652-020-02195-6.

[12] D. B. Blumenthal, S. Bougleux, A. Dignös, J. Gamper, Enumerating dissimilar minimum cost perfect and error-correcting bipartite matchings for robust data matching, Inf. Sci. 596 (2022) 202–221.

[13] M. M. Fredrickson, J. Errickson, B. B. Hansen, Comment: Matching methods for observational studies derived from large administrative databases, Stat. Sci. 35 (3) (2020) 361–366. doi:10.1214/19-STS740.

[14] F. Rajabi-Alni, A. Bagheri, A faster algorithmu for the limited-capacity many-to-many point matching in one dimension (2019). arXiv:1904.03015.