Computing Minimum Limited-Capacity Matching in one-Dimensional space and for the Points Lying on Two Perpendicular Lines

Fatemeh Panahi* Ali Mohades†

Abstract

Let $A = \{a_1, a_2, ..., a_s\}$, and $B = \{b_1, b_2, ..., b_r\}$ be two sets of points such that $s + r = n$. Also let $C_A = \{\alpha_1, \alpha_2, ..., \alpha_s\}$ and $C_B = \{\beta_1, \beta_2, ..., \beta_r\}$ be the capacities of points in A and B. We define minimum limited capacity matching and call it MLC-matching that matches each point $a_i \in A$ to at least one and at most α_i points in B and matches each $b_j \in B$ to at least one and at most β_j points in A, for all i,j where $1 \leq i \leq s$, $1 \leq j \leq r$, such that sum of all the matching costs is minimized. Cost of matching $a_i \in A$ to $b_j \in B$ is equal to the distance between a_i and b_j. In one-dimensional space, we present an $O(kn^2)$ algorithm to compute MLC-matching, where $k = \min\{\max(\alpha_i, \beta_j), n\}$, which also works when points of A and B lie on two parallel lines and also on two unparallel lines, on one side of the cross point. We improved the algorithm to $O(n \log n)$ for the points lying on two perpendicular lines.

1 Introduction

Point matching is a practical method to determine and measure the relation between two sets. A matching between two sets maps individual points in one set with individual points in the other one [1]. It has various applications in many fields such as computer vision, pattern recognition, music information retrieval, and philosophy of science to name a few.

There are different types of point matching problems, depending on the application they are raised from. In point set pattern matching (PSPM), the problem is determining the similarity of two point sets which represent two different objects, under different transformations such as translation, and rigid motion or congruence [2]. Generally, the resemblance of two point sets is expressed by the distance between them. Eiter and Mannila reviewed some of distance measures including Hausdorff distance, sum of minimum distances, surjection distance, and fair surjection distance [3]. All of them were computed in polynomial times. They also proposed a distance measure called link distance. A linking between two sets A and B is a set of matched pair, L, which maps all elements of A with at least an element in B and vice-versa. The link distance is sum of the distances between each two points in L. The link distance has been first described as a measure of similarity between two theories in a logical language and computed in $O(n^3)$ time. Colannino and Toussaint proposed an $O(n^2)$ algorithm for computing the link distance in one dimensional space [4]. In another work, Colannino et al. improved the $O(n^2)$ algorithm to an $O(n \log n)$ one and called it many-to-many matching [1]. One-to-one matching is a perfect matching, which matches each point in one set exactly with one point in the other set [5].

The assignment problem, which is one of the fundamental combinatorial optimization problems, aims at finding the minimum perfect matching in two point sets. The two point sets represent agents or server centers and tasks or client centers, while matches represent assignments. Any task can be assigned to any agent, and each task-agent assignment has a specific cost. The problem is to assign exactly one agent to each task to perform all tasks such that the total cost of the assignment is minimized. The Hungarian algorithm is a well known algorithm that solves the minimum perfect matching in $O(n^3)$ time where n is the total number of input points [5]. Besides the advantages of one to one matching, in some applications like management, industrial sciences and transition networks, it is desired to assign each center to multiple ones. When it is required to map each center with at least one center in the other set, the algorithm of computing minimum many-to-many matching could be useful. To prevent mapping the points to a large number of points in the other set, it is required to impose some restrictions. In this paper, we consider the case that the capacities of the points are limited, and therefore the number of matches to each point should be limited. Let $A = \{a_1, a_2, ..., a_s\}$ and $B = \{b_1, b_2, ..., b_r\}$ be two sets of points such that $s + r = n$. $C_A = \{\alpha_1, \alpha_2, ..., \alpha_s\}$ and $C_B = \{\beta_1, \beta_2, ..., \beta_r\}$ are the capacities of points in A and B respectively. We define minimum limited capacity $MLC - matching$ that matches each point $a_i \in A$ to at least one and at most α_i points in B and matches

*Laboratory of Algorithms and Computational Geometry, Department of Mathematics and Computer Science, Amirkabir University of Technology, Tehran, Iran fatemehpanahi@aut.ac.ir

†Laboratory of Algorithms and Computational Geometry, Department of Mathematics and Computer Science, Amirkabir University of Technology, Tehran, Iran mohades@aut.ac.ir

each $b_j \in B$ to at least one and at most β_j points in A, for all i, j where $1 \leq i \leq s$ and $1 \leq j \leq r$, such that sum of the all matching costs is minimized. Cost of matching $a_i \in A$ to $b_j \in B$ is equal to the distance between a_i and b_j. Generally, the distance between the centers is the Euclidian distance in 2D space. In some production line models, the resources and the client centers are located on one line or two separated lines.
For the general case MLC-matching can be computed in $O(n^3)$ by phrasing the problem as a network-flow problem and computing a minimum cost flow. In this work, we present an $O(kn^2)$ algorithm, where $k = \min\{\max(\alpha_i, \beta_j), n\}$ for computing MLC-matching problem in one-dimensional space. We improved the algorithm to $O(n \log n)$ for the case that the points of two sets lie on two perpendicular lines.

If a node matches to the maximum number of points which could be matched, i.e. its capacity, we call it a full node. If a node matches to exactly one point, it is referred to as a single-matched node.

We start with some properties of this matching.

2 Algorithms

The matching between A and B is a set of matched pairs (a, b) where $a \in A$ and $b \in B$. Limited capacity matching is defined on two point sets $A = \{a_1, a_2, ..., a_s\}$ and $B = \{b_1, b_2, ..., b_r\}$ which have the capacity sets $C_A = \{\alpha_1, \alpha_2, ..., \alpha_s\}$ and $C_B = \beta_1, \beta_2, ..., \beta_r$, respectively. In limited capacity matching, each point $a_i \in A$ is matched to at least one and at most a_i points in B and each $b_j \in B$ to at least one and at most $_j$ points in A, for all i,j where $1 \leq i \leq s$ and $1 \leq j \leq r$. Cost of matching $a_i \in A$ to $b_j \in B$ is equal to the distance between a_i and b_j. MCL-matching is a limited capacity matching in which the sum of the all matching costs is minimized.

If a node matches to the maximum number of points which could be matched, i.e. its capacity, we call it a full node. If a node matches to exactly one point, it is referred to as a single-matched node. We start with some properties of this matching.

Observation 1 *MLC-matching is a set of trees in which at most one node in each tree has degree greater than one.*

Observation 2 *For each four nodes a_i, $a_{j'} \in A$ and b_j, $b_{j'} \in B$, if*
$distance(a_i, b_j) + distance(a_{i'}, b_{j'}) < distance(a_i, b_{j'}) + distance(a_{i'}, b_j)$,
then edges (a_i, b'_j) and (a'_i, b_j) can not both be in the matching.

2.1 The Algorithm for one-dimensional space

Assume that $A = \{a_1, a_2, .., a_r\}$ and $B = \{b_1, b_2, .., b_s\}$ lie on the real line, and $C_A = \{\alpha_1, \alpha_2, ..., \alpha_r\}$ and $C_B = \{\beta_1, \beta_2, ..., \beta_s\}$ are their capacities. We can suppose that the points lie on the $x-$ axis and are sorted in ascending order of their $x-$coordinates. We present a recursive algorithm to find a MLC-matching for A and B.

Lemma 1 *The matched pair (a_r, b_s) exists in a MLC-matching.*

Proof. Suppose that the lemma is false. Let M be a MLC-matching that does not contain (a_r, b_s). a_r should be matched to some node like b_k, where $k < s$. In this case, according to observation 2, b_r could not be matched to any of nodes, a contradiction. □

Lemma 2 *If a_i is matched to nodes, b_k and $b_{k'}$ $(k < k')$, in a MLC-matching, then it is matched to all b_j for each $k < j < k'$.*

Proof. It is clear according to observation 2. □

Let $S(i, j)$ be the function which computes the cost of MLC-matching for two sets, $A = \{a_1, a_2, .., a_i\}$ and $B = \{b_1, b_2, .., b_j\}$, and their capacities, $C_A = \{\alpha_1, \alpha_2, ..., \alpha_i\}$ and $C_B = \{\beta_1, \beta_2, ..., \beta_j\}$. Considering lemma 2, (a_i, b_j) is in the MLC-matching. For convenience of presentation the points in figure 1 are shown on the separated lines.

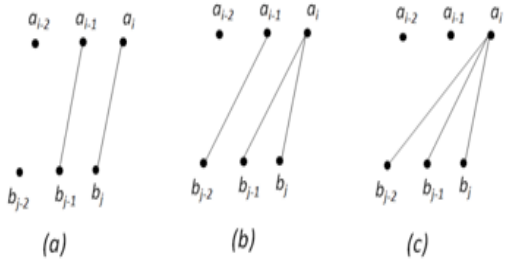

Figure 1: The Possible cases for matched nodes to a_i in MLC-matching

The possible cases for a_i in MLC-matching are as follow:

- a_i is matched to b_j and they are not mapped to any other nodes. In this case, MLC-matching contains (a_i, b_j) and all matched pairs in $A = \{a_1, a_2, .., a_{i-1}\}$ and $B = \{b_1, b_2, .., b_{j-1}\}$ (figure 1(a)).

- a_i is matched just to b_j and b_{j-1}, while b_j and b_{j-1} are not mapped to any other node (figure 1(b)),

- a_i matches to three nodes, b_j, b_{j-1}, and b_{j-2} (figure 1(c)),
- ... .
- a_i is full and matches to α_i nodes.

There are similar cases for b_j.

With regards to the above cases, we can define the function S recursively as,

$S(i,j) = \min\{|a_ib_j| + S(i-1,j-1), |a_ib_j| + |a_{i-1}b_j| + S(i-2,j-1), ..., \Sigma_{m=0}^{\beta_j}|a_{i-m}b_j| + S(i-\beta_j, j-1), |a_ib_j| + |a_ib_{j-1}| + S(i-1,j-2), ..., \Sigma_{m=0}^{\alpha_i}|a_ib_{j-m}| + S(i-1, j-\alpha_i)\}$.

To compute the value of S, dynamic programming can be used. All cases of $S(i,j)$ are $O(n^2)$ and each of which need at most $O(k)$ time where $k = \max_{1\le i\le r, 1\le j\le s}(\alpha_i, \beta_j)$. Totally, the algorithm can be run in $O(kn^2)$ time where k can be fixed as a constant. Theorem 3 shows this result. Since this problem is enough significant in application, improving this algorithm using more complicated data structures and algorithms has considered as a future work. One of the advantages of the proposed algorithm is that it can be applied in the cases that points lie on two parallel lines and also on two unparallel lines, on one side of the cross point, which has applications in some production line models.

Theorem 3 *Let A and B be two sets of points in one-dimensional space. The MLC-Matching of the sets can be computed in $O(kn^2)$, where $k = \min\{\max(\alpha_i, \beta_j), n\}$. Such an algorithm can be applied in the cases that points lie on two parallel lines and also on two unparallel lines, on one side of the cross point, which has applications in some production line models.*

2.2 The Algorithm for the Points Lying on two Perpendicular Lines

In this section, we consider the case that $A = \{a_1, a_2, .., a_r\}$ and $B = \{b_1, b_2, .., b_s\}$ lie on two perpendicular lines, L_1 and L_2, which cross each other in the point O, for example the $x-$ and $y-$axes. Suppose that $s \le r$ and the elements of A and B are sorted in ascending order of their distance from O.

Lemma 4 *For each i, i' where $|a_iO| < |a_{i'}O|$, if a_i and $a_{i'}$ are matched to b_j and $b_{j'}$, respectively, then $|b_jO| < |b_{j'}O|$.*

Proof. Suppose that the lemma is not true. Let M be a MLC-matching including the pairs (a_i, b_j) and $(a_{i'}, b_{j'})$ where $|a_iO| < |a_{i'}O|$ and $|b_jO| > |b_{j'}O|$. The nodes a_i and $a_{i'}$, also b_j and $b_{j'}$, can lie on two opposite sides of the cross point O, or on the same side of it. The following cases may take place:

- both a_i , $a_{i'}$ and b_j , $b_{j'}$ are on the same side of the cross point O (figure 2(a)). Since $|a_iO| < |a_O|$ and $|b_jO| > |b_{j'}O|$, (a_i, b_j) and $(a_{i'}, b_{j'})$, will cross each other. According to proposition 2 and the triangular inequality, it contradicts the assumption that the matching is optimum.
- a_i , $a_{i'}$ are on two opposite sides and b_j , bj' are on the same side of the cross point O (figure 2(b)). Consider the mirror image of a_i or $a_{i'}$ with respect to O on L_1. In figure 2(b) the mirror image of $a_{i'}$, $M(a_{i'})$, is demonstrated. As $|a_{i'}b_{j'}| = |M(a_{i'})b_{j'}|$, we can consider $M(a_{i'})$ instead of $a_{i'}$). (a_i, b_j) and $(M(a_{i'}), b_{i'})$ cross each other and this contradicts proposition 2.
- b_j , $b_{j'}$ are on two opposite sides and a_i , $a_{i'}$ are on the same side of the cross point O. this case is similar to the previous one.
- Both a_i, $a_{i'}$ and b_j , $b_{j'}$ are on two opposite sides of O (figure 2(c)). Consider the mirror image of a_i or $a_{i'}$ with respect to O on L_1 and the mirror image of of b_j or $b_{j'}$ with respect to O on L_2. The symmetries of a_i and b_j, $M(a_i)$ and $M(b_j)$, are shown in figure 2(c). We can consider $M(a_i)$ and $M(b_j)$ instead of a_i and b_j. $(a_{i'}, b_{j'})$ and $(M(a_i), M(b_j))$ cross each other and this contradicts proposition 2.

□

It is observed that the value of a solution does not change if we replace a point at $(x, 0)$ with $x < 0$ by the point $(-x, 0)$. Similarly we can remove all negative y-coordinates. So without loss of generality we can assume that all points lie on the positive $x-$ and $y-$axes.

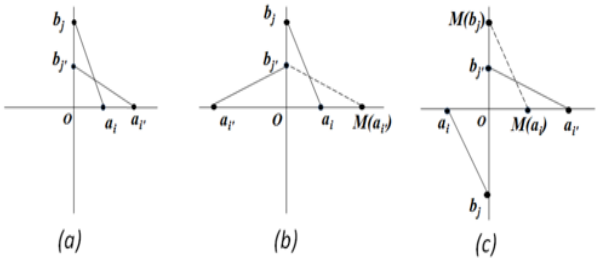

Figure 2: The possible cases for nodes a_i ,$a_{i'}$, b_j, and $b_{j'}$ with respect to the cross point O

Lemma 5 *In MLC-matching all points of A, i.e. the larger set, are single-matched.*

Proof. Assume that the lemma is not true and there is a point like $a_i \in A$ which is matched to b_j and b_k $(k < l)$. In this case, we claim that all points of B should be single-matched. Assume that there is a point $b_j \in B$ which is not single-matched and matches to a_x and a_y : If $j < k$ (figure 4(a)), remove the pairs (a_i, b_k) and (a_y, b_j) and add (a_y, b_k), If $j > k$ (figure 4(b)), remove the pairs (a_i, b_l) and (a_x, b_j) and add (a_x, b_l), The resulted matching is still a limited capacity matching and has a smaller cost. There is a point in A matching to more than one point and all points of B are single-matched. It is in contradiction with the assumption that A is the larger set. □

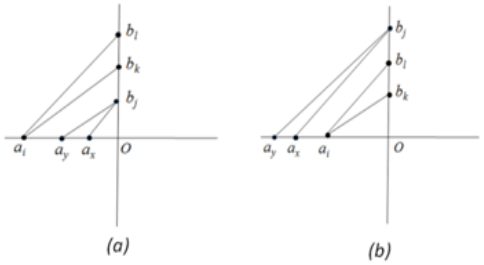

Figure 3: The position of b_j with respect to b_l and b_k

Lemma 6 *If a_i is not a single-matched point, then all points which are closer than a_i to O, will be full.*

Figure 4 illustrates an example of a MLC-matching. It is observed that all points closer than b_3 to O are full and all points farther than b_3 to O are single-matched.

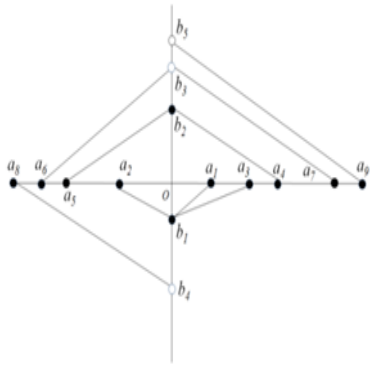

Figure 4: MLC-matching for $A = \{a_1, a_2, ..., a_9\}$ anb $B = \{b_1, b_2, .., b_5\}$ and their capacities $C_A = \{1, 3, 2, 4, 2, 1, 3, 3, 2\}$ and $C_B = \{3, 2, 3, 4, 3\}$.

There is a point b_k such that b_1, b_2,..,b_{k-1} are full and b_{k+1}, $b_(k+2)$,..,b_s are single-matched. There are $s - k$ single-matched points in B, which are matched to $s - k$ end points of A. The number of points in A matched to full nodes in B is $\Sigma_{j=1}^{k-1}\beta_j$. The remaining points of A, are matched to b_k.

$$\text{Therefore, } s - k + \Sigma_{j=1}^{k-1}\beta_j < r \ , \ s - k + \Sigma_{j=1}^{k-1}\beta_j \geq r$$

Indeed, k is the greatest integer number that $s - k + \Sigma_{j=1}^{k-1}\beta_j < r$.

By computing amount of k it is clear what the solution is. Theorem 7 shows the result.

Theorem 7 *Let A and B be two sets of points lying on two perpendicular lines. The MLC-matching of the sets can be computed in $O(n \log n)$ which is the time required for sorting points. The next steps of the algorithm are run in $O(n)$. Therefore, MLC-matching of sorted points can be computed in $O(n)$ time.*

3 Conclusion

In this paper, we studied a variant of point matching problem, called MLC-matching, in which each point has a limited capacity. Such a limitation is practically noticeable where the points represent server-client centers while matches represent assignments. The various centers have different capacities to map to each other. In this paper, the problem was studied from a geometrical view point for the first time. Regarding real word issues, the distance between centers is considered Euclidian distance in one and two dimensional space. We stated the Euclidian distance for computing MLC-matching and in one-dimensional space, presented an $O(kn^2)$ algorithm, where $k = \min\{\max(\alpha_i, \beta_j), n\}$. Also, we improved the algorithm to $O(n \log n)$ for the points lying on two perpendicular lines.

References

[1] J. Colannino, M. Damian, F. Hurtado, S. Langerman, H. Meijer, S. Ramaswami, D. Souvaine, and G. Toussaint Efficient many-to-many point matching in one dimension. *Graphs and Combinatorics, vol. 23, pp. 169-178, 2007.*

[2] A. Bishnu, S. Das, S. C. Nandy, and B. B. Bhattacharya, Simple algorithms for partial point set pattern matching under rigid motion *Pattern Recognition, vol. 39, pp. 1662-1671, 2006.*

[3] T. Eiter and H. Mannila Distance measures for point sets and their computation *Acta Informatica, vol. 34, pp. 109-133, 1997.*

[4] J. Colannino and G. Toussaint A faster algorithm for computing the link distance between two point sets on the real line *Citeseer 2005.*

[5] R. M. Karp and S. Y. R. Li Two special cases of the assignment problem *Discrete Mathematics, vol. 13, pp. 129-142, 1975.*

[6] L. R. Foulds Combinatorial optimization for undergraduates *Springer Verlag, 1984*