Bottleneck Partial-Matching Voronoi Diagrams

Matthias Henze* Rafel Jaume†

Abstract

Given two point sets in the plane, we study the minimization of the (lexiocographic) bottleneck distance between the smaller set and an equally-sized subset of the larger set under translations. We relate this problem to two Voronoi-type diagrams, which are of interest for related problems. Their complexity is shown to be polynomial in the sizes of the sets and algorithms for their construction are given.

1 Introduction

Applications often demand algorithms to find an occurrence of a point pattern in a bigger set of points. It is also common to define a similarity measure between the pattern and the point set as the minimum among the images of the pattern under a set of allowed transformations.

One of the most studied similarity measures for two point sets $A, B \subset \mathbb{R}^2$ is the directed Hausdorff distance, which pairs each point of B to its nearest neighbor in A, and then takes the maximum of the distances between pairs. A drawback of this approach is that only one of the points of B is actually relevant and removing the others would not affect the value of the distance. However, if one imposes that the pairs form a matching, i.e., they are assigned in an injective way, this effect is avoided. The resulting distance is called the *bottleneck distance* and it was introduced for equally sized sets in [1] as

$$\Delta(B, A) = \min_{\sigma: B \hookrightarrow A} \max_{b \in B} \|b - \sigma(b)\|,$$

where $\|\cdot\|$ denotes the Euclidean norm. In contrast to the directed Hausdorff distance, the distance $\Delta(B, A)$ has the advantage of being symmetric for equally sized sets. On the other hand, it is harder to compute, since the points cannot be regarded independently but complicated interactions may occur. A peculiarity of this distance is that the matching attaining the minimum is in general not uniquely defined, even when the pair $\ell(B, A) \in B \times A$ that attains $\Delta(B, A)$ is unique. To avoid this ambiguity, we consider the matching that lexicographically minimizes the distances between matched points. This problem was introduced in [3] and revisited in [11] in terms of weighted bipartite graphs. We consider a special case of this problem where the graph is defined on the vertex set $A \cup B$ and the weight of an edge $\{a, b\}$ is the Euclidean distance between a and b. Formally, given finite point sets A and B with $k = |B| \leq |A| = n$, the matching $\pi : B \hookrightarrow A$ is the *lex-bottleneck matching* $m(B, A)$ if and only if

$$\|b_0 - \pi(b_0)\| = \max_{b \in B} \|b - \pi(b)\| \leq \max_{b \in B} \|b - \sigma(b)\|,$$

for every other matching $\sigma : B \hookrightarrow A$ and the restriction of π to $B \backslash \{b_0\}$ is $m(B \backslash \{b_0\}, A \backslash \{\pi(b_0)\})$, defining $m(\emptyset, \cdot) = \emptyset$.

To the best of our knowledge, the problem of computing the bottleneck distance between point sets of different sizes under translations has not been addressed so far. For equally sized sets, Alt et al. [1] gave an $O(n^6 \log n)$ algorithm to compute the bottleneck distance between the sets under translations. An algorithm running in $O(n^5 \log^2 n)$ time was later found by Efrat et al. in [6]. We follow Rote [10] introducing a Voronoi-type diagram that partitions the space of translations according to the (partial) matching that minimizes the least-squares distance between one of the sets and a translated copy of the other set (see [7] for a follow-up study). As a by-product, we get an algorithm that computes the bottleneck distance under translations in time quadratic in the size of the bigger set. The first partition we are interested in is defined by the cells

$$V_\pi = \{t \in \mathbb{R}^2 : \pi = m(B + t, A)\},$$

for $\pi : B \hookrightarrow A$. We call this tessellation the *lex-bottleneck partial-matching Voronoi diagram* (LVD for short). We also consider the *bottleneck partial-matching Voronoi diagram* (BVD), having cells

$$V_{b \mapsto a} = \{t \in \mathbb{R}^2 : (b, a) = \ell(B + t, A)\},$$

for $b \in B$ and $a \in A$. These two structures are polyhedral complexes in the wider sense that their cells are possibly non-convex polygons. We show that the complexity of both BVD and LVD is $O(n^2 k^6)$. In addition, we provide algorithms to construct them that find the optimal matching under translations and run in $O(n^2 k^8)$ and $O(n^2 k^{9.5} \log k)$ time, respectively.

*Institute of Computer Science, Freie Universität Berlin, Germany, matthias.henze@fu-berlin.de, Research supported by ESF EUROCORES programme EuroGIGA-VORONOI, (DFG): RO 2338/5-1.

†Institute of Computer Science, Freie Universität Berlin, Germany, jaume@mi.fu-berlin.de, Research supported by La-Caixa and the DAAD.

2 Bottleneck matchings in bipartite graphs

Let G be a bipartite graph with edge set E and vertex set partitioned into components U and V of sizes $k = |U| \leq |V| = n$. A *maximal matching* in G is a matching $m \subseteq E$ such that the number of matched vertices is maximized. In order to simplify the notation, we denote the edge $\{a, b\}$ by ab. We identify a matching in G with the injection of U into V it induces. We assume that G is complete and the weights are given by a bijection $w : E \leftrightarrow \{1, \ldots, kn\}$.

An injection $\pi : U \hookrightarrow V$ is called a *bottleneck matching* with respect to w if

$$\max_{i \in U} w(i\pi(i)) \leq \max_{i \in U} w(i\sigma(i)),$$

for every injection $\sigma : U \hookrightarrow V$.

One of the best-known algorithms to find a maximal matching in a bipartite graph is due to Hopcroft and Karp [8]. It is based on iteratively augmenting a partial matching $m \subseteq E$. An *augmenting path for* m is a path $p \subseteq E$ starting and ending on non-matched vertices and alternating between edges in m and $E \setminus m$. Note that the matching $m' = (m \setminus p) \cup (p \setminus m)$ has one more matched vertex than m. The algorithm starts with $m = \emptyset$ and at each iteration increases the matching using augmenting paths. If no augmenting path is found, Berge's lemma [2] implies that the matching is maximal. During the execution, a directed graph $\overrightarrow{G}$ representing the current matching is maintained. At first, all the edges in E are transformed into arcs from U to V. When an augmenting path is found, its arcs are flipped, so that the arcs pointing from V to U represent the edges of the matching.

A careful analysis (as the one in [9]) shows that the Hopkroft-Karp algorithm runs in $O(|E|\sqrt{k})$ for complete graphs. Since G is complete, we can show that any bottleneck matching uses only edges among the k of smallest weight incident to every vertex in U. We say that an edge is *relevant* if it is one of these k^2 edges. Thus, in our special setting, after pruning the useless edges and vertices in $O(nk \log n)$ time, a maximal matching can be found in $O(k^2\sqrt{k})$.

It was already proposed in [6] to conduct a binary search to find a bottleneck matching. We assume that the weights of the relevant edges range from 1 to k^2. If it is not the case, we can convert the weights into this form in $O(k^2 \log k)$ time. Given $r \in \{1, \ldots, k^2\}$, we define $G(r)$ to be the graph that remains after removing all the edges from G whose weight is not among the r smallest. A complete matching m in $G(r^*)$ is bottleneck if and only if $G(r^* - 1)$ has no complete matching. Thus, the bottleneck matching can be found conducting a binary search consisting of $O(\log k)$ executions of the Hopcroft-Karp algorithm.

An injection $\pi : U \hookrightarrow V$ is the *lex-bottleneck matching* with respect to w, and denoted by $m(U, V, w)$, if

$$w(u_0\pi(u_0)) = \max_{u \in U} w(u\pi(u)) \leq \max_{u \in U} w(u\sigma(u)),$$

for every matching $\sigma : U \hookrightarrow V$, and

$$\pi|_{U \setminus \{u_0\}} = m(U \setminus \{u_0\}, V \setminus \{\pi(u_0)\}, w),$$

defining $m(\emptyset, \cdot, \cdot) = \emptyset$.

In order to find the lex-bottleneck matching it is enough to translate the definition into an incremental algorithm, applying k times the previous procedure to find a bottleneck matching and removing the corresponding vertex and edge from the graph after each iteration.

3 Complexity of the matching Voronoi diagrams

In this section, we derive combinatorial bounds that help us to analyze the algorithms in Section 4. Before establishing the complexity bounds for the diagrams, we need the following technical lemma resulting from applying the Clarkson-Shor technique (see [5]).

Lemma 1 *The number of bisectors supporting an edge of the j-th order Voronoi diagram of a set of n points in the plane for some $j \leq k$ is $O(nk)$.*

Proof. Let A be a set of n points in the plane and consider the set of bisectors defined by pairs of points in A. For each such bisector, we define its conflict set as one $Q \subset A$ of minimal cardinality such that there is an edge in the Voronoi diagram of $A \setminus Q$ supported by the bisector. The weight of a bisector is the size of its conflict set. We denote by $N_{<k}(n)$ the maximum number of bisectors of weight smaller than k defined by n points in the plane and by $N_0(n)$ the maximum number of bisectors with empty conflict set. The bound of Clarkson and Shor gives $N_{<k}(n) = O(k^2 N_0(n/k))$. Note that a bisector supports an edge of j-th order for $j \leq k$ if and only if its weight is smaller than k. Hence, the number we want to bound is indeed $N_{<k}(n)$. Since the Voronoi diagram of a point set $R \subset \mathbb{R}^2$ of size n/k has $O(n/k)$ edges, at most that many bisectors can support an edge of weight zero of R. Thus, the desired bound $N_{<k}(n) = O(nk)$ follows. □

In the next theorem, we show that the LVD and the BVD are coarsenings of a polyhedral tessellation of the plane and we bound its complexity (total number of faces) using the previous lemma.

Theorem 2 *Let $A, B \subset \mathbb{R}^2$ be sets of k and n points. The LVD and the BVD of A and B are coarsenings of a polyhedral tessellation of $O(n^2k^6)$ complexity.*

Proof. Note first that the (lex-)bottleneck matching for a fixed translation t depends only on the total order of the values $\{\|b+t-a\| : b \in B, a \in A\}$. Moreover, the discussion in the previous section reveals that it is sufficient to consider this order between relevant edges. Given two edges incident to the same $b \in B$, this order is fixed in any cell of the overlay of the j-th order Voronoi diagrams of $A - b$ for all $j \leq k$. It is less obvious that the relations between a relevant edge incident to $b \in B$ and a relevant edge incident to a different $b' \in B$ are fixed in the overlay of the j-th order diagrams of $S = (A - b) \cup (A - b')$ for all $j \leq 2k$. This is because Voronoi edges of order bigger than $2k$ must be between regions of points $a - b$ and $a' - b'$ such that either a is not among the k closest points of $b + t$ or a' is not among the k closest points of $b' + t$. Lemma 1 ensures that all the edges of j-th order Voronoi diagrams for $j \leq 2k$ are supported by $O(nk)$ lines. The arrangement of such lines for the $O(k^2)$ mentioned diagrams must be a refinement of LVD and BVD and it has the indicated complexity. □

Note that the number of regions of the BVD is actually bounded by nk. However, since the regions are not necessarily convex, the bound derived in the previous theorem is not trivial.

4 Construction of the matching Voronoi diagrams

Let $\mathcal{H}$ be the arrangement of lines described in the proof of Theorem 2. We now construct the matching Voronoi diagrams by coarsening $\mathcal{H}$. Given a cell C in $\mathcal{H}$, let G_C be the complete bipartite graph on the vertex set $A \cup B$ whose edges are weighted by the total order w_C on the distances $\|b + t - a\|$ attained at any point t interior to C for $a \in A$ and $b \in B$. We denote by $m(C)$ the lex-bottleneck matching in G_C and by $\ell(C)$ the longest edge of $m(C)$.

Lemma 3 *Let C, D be cells of $\mathcal{H}$ sharing an edge on which $\|b + t - a\| = \|b' + t - a'\|$ and such that $w_C(ab) > w_C(a'b')$. If $ab \notin m(C)$, $a'b' \in m(C)$ and $G_D(w_D(ab))$ has a complete matching (ignoring vertices of B matched by longer edges), then $ab \in m(D)$ and $a'b' \notin m(D)$. Otherwise, $m(C) = m(D)$.*

Proof. Let $w_1 = w_C(ab)$ and $w_2 = w_C(a'b')$. Note that it has to be $w_1 = w_2 + 1$. It is easy to see that $G_D(j) = G_C(j)$ for all $j \neq w_2$. Observe also that if $ab \in m(C)$, then $ab \in m(D)$. Otherwise, $m(D)$ would be lexicographically better than $m(C)$ in C. The counterpositive of the symmetric argument implies that if $a'b' \notin m(C)$, then $a'b' \notin m(D)$. If $ab \in m(C)$ and $a'b' \notin m(C)$, the previous rules imply $ab \in m(D)$ and $a'b' \notin m(D)$. The smaller edges must coincide as well due to the optimality of $m(C)$ in C. If $ab, a'b' \in m(C)$, the rules imply $ab \in m(D)$. If $a'b' \notin m(D)$, it would be $m(D)$ lexicographically better than $m(C)$ in C. Hence, $ab, a'b' \in m(D)$ and $m(D) = m(C)$, by arguments similar to the previous case. If $ab, a'b' \notin m(C)$, the rules imply $a'b' \notin m(D)$. If $ab \in m(D)$, it would be $m(C)$ lexicographically better than $m(D)$ in D. Hence, $ab, a'b' \notin m(D)$ and $m(D) = m(C)$.

If $ab \notin m(C)$, $a'b' \in m(C)$, the rules do not help. However, exactly one of ab and $a'b'$ belongs to $m(D)$, since the two other options would contradict the optimality of $m(C)$ in C or of $m(D)$ in D, as in the two previous cases. If $G_D(w_2)$ has no complete matching (after removing the elements of B matched by the longer edges common to C and D), then $a'b'$ must belong to $m(D)$ and $m(D) = m(C)$. If it has a complete matching, then $ab \in m(D)$ and $a'b' \notin m(D)$, as stated. □

Besides providing some insight into the dynamic behavior of the lex-bottleneck matching, the previous lemma helps to prove the main result of this section.

Theorem 4 *Let $A, B \subset \mathbb{R}^2$ be sets of k and n points. The BVD and the LVD of A and B can be computed in $O(n^2k^8)$ and $O(n^2k^{9.5} \log k)$ time, respectively.*

Proof. We focus first in the computation of the BVD. We construct first the line arrangement $\mathcal{H}$ in $O(n^2k^6)$ time. To do so, we need to select the lines involved in the $O(k^2)$ diagrams described in the proof of Theorem 2. We use the algorithm from Chan [4] that constructs the facial structure of the $(\leq k)$-level of an arrangement of n planes in $O(n \log n + nk^2)$ expected time or the deterministic version running in $O(nk^2(\log n/\log k)^{O(1)})$ time. We can then traverse these facial structures and collect the $O(nk^3)$ relevant lines to finally construct their arrangement, remembering which planes (corresponding to edges) induce every lie. We choose then a translation t interior to a cell C of $\mathcal{H}$. We compute and all the distances for that translation and sort them to get the edge weights in $O(nk \log n)$ time. We prune the non-relevant edges and normalize the weights such that they are contained in $\{1, \ldots, k^2\}$ in $O(k^2 \log k)$ time. We conduct a binary search as described in Section 2 and obtain a bottleneck matching in time $O(k^{2.5} \log k)$. We store the longest edge $\ell(C)$ and the directed graph $\overrightarrow{G_C(w_C(\ell(C)))}$ computed by the previous algorithm to certificate $c = \ell(C)$. Next, we traverse $\mathcal{H}$ updating the changes in the structures and we merge cells when the matching does not change. Specifically, we maintain, for each $b \in B$, an ordered list of its k closest neighbors in A and the weight of each element in these lists with respect to the total order w_C. To do that, if we cross a bisector between two elements of the same list, we just swap their position in the list and their weights. If one of the edges is in the

list and the other is not, the second one substitutes the first one in the list and takes its weight when the bisector is crossed. If none of the edges is in the corresponding list, no change is needed. If the bisector is between elements of different lists, we swap their weights. All these updates can be done in $O(k)$ time. Lemma 3 ensures that $\ell(C)$ may change only when its weight is increased by a swap with another edge e when entering a cell D. In such a case, we need to decide if $G_D(w_D(e))$ has a complete matching. This is equivalent to remove the directed edge associated to $\ell(C)$ from the graph $\overrightarrow{G_C(w_C(\ell(C)))}$ representing the bottleneck matching in C, adding the directed edge corresponding to e, and deciding if there is an augmenting path from the non-matched point of B to some non-matched point in A in the new graph. This can be done in $O(k^2)$ time.

A similar approach can be used to construct the LVD. We need to maintain the graphs $\overrightarrow{G_C(w_C(e))}$ for all edges e of the current matching. When one of the heaviest edges in one of these graphs becomes heavier, a test for an augmenting path must be done. If the result is affirmative, we may need to recompute the matching for all the lighter edges, which can require up to $O(k^{3.5} \log k)$ time. □

In order to find the minimal bottleneck distance under translations and the translation for which it is attained, we only need to incorporate to the previous algorithm an optimization in each (convex) cell of $\mathcal{H}$.

Corollary 5 *Let $A, B \subset \mathbb{R}^2$ be sets of k and n points. The minimal bottleneck matching for A and B under translations can be found in $O(n^2k^8)$ time.*

Proof. The previous theorem ensures that the BVD of A and B can be constructed in $O(n^2k^8)$. Once we have the arrangement, we only need to optimize t in every region and keep the minimum throughout the diagram. In every region, we can equivalently minimize the square of the Euclidean norm instead of the norm itself. More precisely, in $V_{b\mapsto a}$ we will minimize the quadratic function $f(t) = \|b + t - a\|^2$. Since the regions of the BVD are in general non-convex, we will minimize f in every cell of $\mathcal{H}$ separately. Let C be one of the cells of $\mathcal{H}$ in $V_{b\mapsto a}$. Let $t_0 = a - b$ be the global minimum of the function $f(t)$. If $t_0 \in C$, obviously $f(t_0) = 0$ is the minimum of f in C. Otherwise, it can be shown that the minimum is attained at a vertex of C or at the intersection of an edge with the line perpendicular to it through t_0 (if the intersection belongs to the edge). Note that every edge will be examined twice and every vertex once for every two-dimensional face containing it. Thus, the total time needed to optimize in all the cells is proportional to the complexity of the diagram. □

5 Conclusion

We considered the problem of minimizing the bottleneck distance between two point sets in the plane when we are allowed to arbitrarily translate one of them. In the spirit of earlier studies, we consider two tessellations associated with this problem which we think are of independent interest and whose cells correspond to "locally optimal" assignments. We derive bounds on the combinatorial complexity of these structures and present algorithms to construct them.

An example of a lex-bottleneck matching Voronoi diagram having complexity $\Omega(n^2k^2)$ can be easily constructed. Therefore, only the dependency on k of the complexity bounds and the algorithms might be improved.

Acknowledgements. We thank Micha Sharir for communicating a proof of Lemma 1.

References

[1] H. Alt, K. Mehlhorn, H. Wagener, and E. Welzl. Congruence, similarity, and symmetries of geometric objects. *Discrete & Computational Geometry*, 3(1):237–256, 1988.

[2] C. Berge. Two theorems in graph theory. In *Proc. of the National Academy of Sciences of the United States of America*, 43:842–844, 1957.

[3] R. E. Burkard and F. Rendl. Lexicographic Bottleneck Problems. *Operations Research Letters*, 10(5): 303–308, 1991.

[4] T. M. Chan. Random sampling, halfspace range reporting, and construction of ($\leq k$)-levels in three dimensions. *SIAM Journal on Computing*, 30(2):586–595, 2000.

[5] K. L. Clarkson and P. W. Shor. Applications of Random Sampling in Computational Geometry, II. *Discrete Comput. Geom.*, 4:387–421, 1989.

[6] A. Efrat, A. Itai, and M. J. Katz. Geometry Helps in Bottleneck Matching and Related Problems. *Algorithmica*, 31(1):1–28, 2001.

[7] M. Henze, R. Jaume, and B. Keszegh. On the complexity of the partial least-squares matching voronoi diagram. In *Proc. 29th European Workshop on Computational Geometry*, pages 193–196, 2013.

[8] J. E. Hopcroft and R. M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2(4):225–231, 1973.

[9] L. Ramshaw and R. E. Tarjan. On minimum-cost assignments in unbalanced bipartite graphs. HP Labs Technical Report HPL-2012-40, 2012.

[10] G. Rote. Partial Least-Squares Point Matching under Translations. In *Proc. 26th European Workshop on Computational Geometry*, pages 249–251, 2010.

[11] P. T. Sokkalingam and Y. P. Aneja. Lexicographic bottleneck combinatorial problems. *Operations Research Letters*, 23(1–2):27–33, 1998.