International Journal of Computational Geometry & Applications

Exact and Approximation Algorithms for Finding an Optimal Bridge Connecting Two Simple Polygons

AMIT M. BHOSLE*

Amazon Software Development Center
3rd Floor, Esteem Arcade, 26 Race Course Road,
Bangalore - 560001, India
bhosle@cs.ucsb.edu

TEOFILO F. GONZALEZ†

Department of Computer Science, University of California,
Santa Barbara, CA - 93106-5110, USA
teo@cs.ucsb.edu

Received (10/18/03)
Revised (4/5/05, 10/10/05)
Communicated by (Joseph S. B. Mitchell)

Given two simple polygons P and Q we define the weight of a bridge (p, q), with $p \in \rho(P)$ and $q \in \rho(Q)$, where $\rho()$ denotes the compact region enclosed by the boundary of the polygon, between the two polygons as $gd(p, P) + d(p, q) + gd(q, Q)$, where $d(p, q)$ is the Euclidean distance between the points p and q, and $gd(x, X)$ is the geodesic distance between x and its geodesic furthest neighbor on X. Our problem differs from another version of the problem where the additional restriction of requiring the endpoints of the bridge to be mutually visible was imposed. We show that an optimal bridge always exists such that the endpoints of the bridge lie on the boundaries of the two polygons. Using this critical property, we present an algorithm to find an optimal bridge (of minimum weight) in $O(n^2 \log n)$ time. We present a polynomial time approximation scheme that for any $\epsilon > 0$ generates a bridge with objective function within a factor of $1 + \epsilon$ of the optimal value in $O(kn \log kn)$ time, where $k = 2 * \lceil \frac{1}{\log(1+\epsilon)} \rceil$. An improved polynomial time approximation scheme and algorithms for generalized versions of our problems are also discussed.

Keywords: Optimal Bridge between Polygons; Polynomial Time Approximation Scheme; Exact Algorithm.

1. Introduction

Given two disjoint simple polygons P and Q, a line segment that connects the two polygons is called a *bridge*. We may think of polygons P and Q being disjoint islands and the objective is to build a bridge between the islands. We may also think of

*Web: http://india.amazon.com
†Web: http://www.cs.ucsb.edu/∼teo

each polygon to be a community or region in which wireless communications may take place. Furthermore, the regions or communities are separated by mountains or obstacles which prevent wireless communications between the two communities. Our problem is to introduce a bridge or wire to allow for inter-community communication. The objective function is to locate a bridge such that the Euclidean length of the path that every message travels from a point in one region to a point in the other region through the bridge is least possible. We use $\rho(X)$ to denote the compact region defined by polygon X, and use $\delta(X)$ to denote the boundary of X. Note that $\rho(X) \cap \delta(X) = \delta(X)$. Formally, for points $p \in \rho(P)$ and $q \in \rho(Q)$ we define the *weight* of the bridge (p, q) as

$$max_{p' \in \rho(P)}\{gd(p', p)\} + d(p, q) + max_{q' \in \rho(Q)}\{gd(q, q')\}, \quad (1)$$

where $d(x, y)$ denotes the Euclidean distance from x to y, and $gd(x, y)$ is the geodesic distance between x and y in their corresponding polygon (i.e., the shortest distance between x and y without leaving the polygon where they are located). A pair (p, q) that minimizes the above expression is called an *optimal bridge*. In Section 2 we show that an optimal bridge always exists between points on the boundary of each of the polygons. Furthermore, points on the bridge just before the bridge endpoints are on the exterior of the two polygons. It is important to note that the endpoints of an optimal bridge are not necessarily *mutually visible* when the polygons are not convex. Figure [1] depicts a problem instance such that the bridge defined by the solid thick line has total weight $\sim x + 2y$, but any bridge between any two visible points (one in P and one in Q) has weight at least $2x + 2y$. Since $x >> y >> z$, it follows that this problem instance does not have an optimal bridge in which its two endpoints are mutually visible. Another way to define the bridge problem is to replace $d(p, q)$ by $gd(p, q)$, (in this case the geodesic path is outside P and Q). We call this problem the *all-geodesic bridge* problem or simply the *ag-bridge* problem. As shown in Figure [1] the endpoints in an optimal ag-bridge (dotted line) are not necessarily *mutually visible* when the polygons are not convex.

Note that an optimal bridge may touch a vertex of P or Q. Also, the geodesic furthest neighbors of the endpoints of an optimal bridge must be vertices in their corresponding polygons.

1.1. *Related Work*

The problem of finding an optimal bridge connecting two *convex* polygons has been solved to optimality. The problem was first studied by Cai, Xu and Zhu[1] for the case when the polygons are convex. They proved that for this case the optimal bridge is between points on the boundary of the polygons. Furthermore, such points are visible from each other. Their algorithm takes $O(n^2 \log n)$ time to construct an optimal bridge between any two given disjoint *convex* polygons. Because this problem is related to other geometric problems (e.g. diameter problems, minimum separations problems, and minimum spanning tree problems), much progress was made immediately after the problem was posed. For the convex polygon case, different optimal

(linear time) algorithms have been developed independently by Bhattacharya and Benkoczi[2], Tan[3], and Kim and Shin[4]. The high-dimensional version of the problem has been studied in Refs. 3 and 5.

The more general version of the problem, when the input polygons are not necessarily convex, has also been studied by several researchers. However, the version of the problem they considered restricts bridges to have endpoints on the boundary of the polygons that are visible from each other. We call this version of the problem the *optimal v-bridge* problem. Note that the optimal v-bridge problem and the optimal bridge problem are identical when the polygons are convex; however, as we have shown in the previous subsection, the problems defined over simple (non-convex) polygons have different solutions. This inequivalence holds even for rectilinear polygons, as shown in the previous subsection. The optimal v-bridge problem for simple polygons has so far resisted linear time algorithms. Kim and Shin[4] introduced this problem and developed a quadratic algorithm to solve this problem. Currently the fastest algorithm for the *exact* solution of the problem is by Tan[6], which runs in $O(n \log^3 n)$ time. This algorithm is quite complex and it makes substantial use of a hierarchical structure that consists of segment trees, range trees and persistent search trees, and a structure that supports dynamic ray shooting and shortest path queries. A restricted version, where the input polygons are simple, but *rectilinear* and the distance between points is measured by the Manhattan distance or L_1 distance, can be solved in linear time using Wang's algorithm[7].

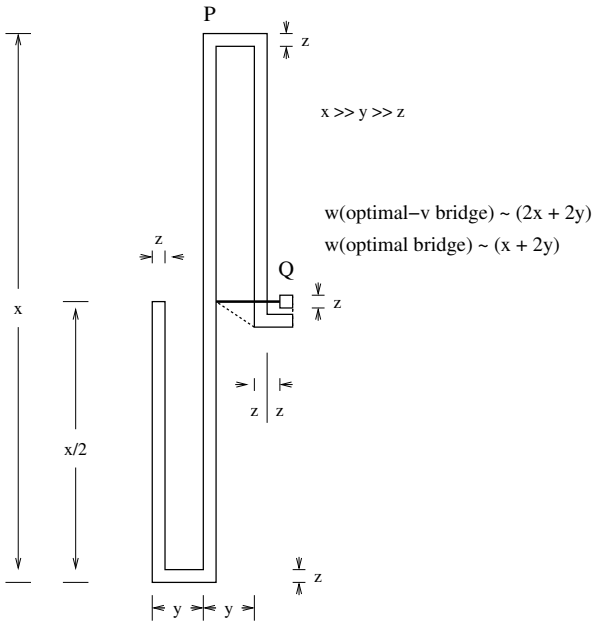

Fig. 1. Problem instance that does not have an optimal bridge defined by two visible points.

A linear-time 2-approximation algorithm, which finds a bridge with objective function value at most twice that of the optimal one, for convex polygons was presented in Ref. 1. Kim and Shin[4] showed that this approximation strategy also works for the v-bridge problem when the polygons are not convex. The time complexity of this approximation scheme is that of finding the *least* distance between (the boundaries of) the two polygons, which can be computed in linear time using the algorithm by Amato[8]. Kim and Shin[4] raise the question as to whether or not a better approximation algorithm exists. Note that this approximation algorithm always generates a bridge whose endpoints are *mutually visible*. Ahn, Cheong, and Shin[9] presented $\sqrt{2}$-approximation algorithm for convex polygons and showed that their technique generalizes to multidimensional space as long as P and Q are are convex regions.

It is easy to see that a closest pair of points between two polygons (the approach in Refs. 1 and 4) forms a bridge with total weight within a factor of 2 times the weight of an optimal bridge between the two polygons. As mentioned before, a closest pair of points between two polygons can be computed in linear time. Since a closest pair of points supports a *valid* v-bridge, an optimal v-bridge between two polygons has weight less than or equal to the one between a closest pair of points between the two polygons. It thus follows that an optimal v-bridge has total weight within a factor of 2 times the weight of an optimal bridge between the two polygons. Furthermore, the problem instance given in Figure 1 can be used to show that the bound of 2 is asymptotically the best possible.

1.2. *Main Results*

In this paper we consider the optimal bridge problem without the restriction imposed in previous papers. Their main difference is that in the former problem the bridge must connect points that are visible to each other (e.g., bridges starting and ending at beaches that are visible from each other), but in the latter problem the bridge endpoints may be inside the region (like the Bay Bridge adjacent to San Francisco, CA). In this situation we want bridges to connect directly to established highways which may be modeled as the actual edges of the polygon that might not be visible from the other polygon. First we establish some important properties of optimal solutions to the bridge problem that will allow us to develop efficient exact and approximation algorithms for this problem. Specifically, we show that an optimal bridge always exists between points on the boundary of each of the polygons. Furthermore, points on the bridge just before the endpoints of the bridge are on the exterior of the polygons. As we showed in the previous section, it is important to note that the endpoints of an optimal bridge are not necessarily *mutually visible* when the polygons are not convex. This is why our problem is different from the optimal v-bridge problem. In Section 2 we show that an optimal bridge always exists between $O(n^2)$ special points defined on the boundary of the polygons.

In Section 2 we establish some important properties of optimal bridges between

two simple polygons. We use these properties to develop two $O(n^2 \log n)$ time algorithms to generate an optimal bridge between two simple polygons. In section 2.1 we present an algorithm to find an optimal vertex bridge (of minimum weight) in $O(n \log n)$ time. Based on this algorithm we present an exact algorithm for the optimal bridge problem that takes $O(n^2 \log n)$ time (Section 2.1). We also present an approximation scheme that given any positive integer k constructs a bridge with objective function value within a factor of $1 + \frac{2}{k+1}$ times that of the optimal one in $O(kn \log kn)$ time. The scheme is presented in Section 3 and it is based on our $O(n \log n)$ time algorithm for the for the optimal vertex bridge problem. In Section 4 we present a polynomial time approximation scheme that given any $\epsilon > 0$ generates a bridge with objective function within a factor of $1 + \epsilon$ times the optimal one in $O(kn \log kn)$ time, where $k = 2 * \lceil \frac{1}{\log(1+\epsilon)} \rceil$. In Section 5 we discuss the changes we need to make in order to apply our results to the ag-bridge problem as well as to other generalized versions of our problem. Our approximation algorithm introduces k artificial points on each line segment and then returns as the bridge an optimal vertex (including the artificial points) bridge. This technique has also been used for finding suboptimal paths between two points in 2D and 3D[10,11,12].

2. Preliminary Results

In this section we establish some important properties of optimal solutions to the bridge problem that will allow us to develop efficient exact and approximation algorithms for this problem. Specifically, we show that an optimal bridge always exists between points on the boundary of each of the polygons. Furthermore, points on the bridge just before the endpoints of the bridge are on the exterior of the polygons. Then we show that an optimal bridge always exists between $O(n^2)$ special points on the boundary of the polygons that we define later. We establish our first result in the following theorem.

Theorem 1. *For every problem instance there is an optimal solution to the bridge problem in which both endpoints of the bridge are boundary points of the polygons. Furthermore, one such optimal bridge, from p to q, is such that in a small neighborhood of p (q) the point on (p, q) closest to p and the one closest to q are on the exterior of P and Q.*

Proof. We prove this theorem by showing that an optimal solution can be transformed to another optimal solution that satisfies the properties of the theorem. Suppose that we have a problem instance in which one or both the endpoints of an optimal bridge are not boundary points of the polygon. Figure 2 gives a problem instance with an optimal bridge between points $p \in \rho(P)$ and $q \in \rho(Q)$ which are not boundary points of the polygons. We now show that there is also an optimal bridge that satisfies the conditions of the theorem. First we show that the bridge from $p \in \rho(P)$ to $q' \in \delta(Q)$, where q' is located at first intersection from q of the boundary of Q and line (p, q), is also an optimal bridge. Let point r be such that

$gd(q, Q) = gd(q, r)$ and r' is such that $gd(q', Q) = gd(q', r')$. Note that r may be the same point as r'.

Since the line (q', q) always stays inside the polygon Q, we have

$$gd(q', r') \leq d(q', q) + gd(q, r')$$

And by definition, we know

$$gd(q, Q) = gd(q, r) \geq gd(q, r')$$

It follows that

$$gd(q', r') \leq d(q', q) + gd(q, r).$$

By substituting the above definitions we know that

$$gd(q', Q) \leq d(q', q) + gd(q, Q).$$

Adding to both sides of the equation $gd(p, P) + d(p, q')$ we know that

$$gd(p, P) + d(p, q') + gd(q', Q) \leq gd(p, P) + d(p, q) + gd(q, Q).$$

This expression establishes that the bridge (p, q') is not worse than bridge (p, q). If the inequality holds in the previous expression then it contradicts that (p, q) is an optimal bridge. But if equality holds, then since (p, q) is an optimal bridge it follows that (p, q') is also an optimal one.

Using the same arguments with point p completes the proof of the first part of the theorem. The proof of the second part of the theorem is simple because if the points in a small neighborhood of p in line (p, q) are inside P or the points in a small neighborhood of q in line (p, q) are inside Q, one may apply arguments similar

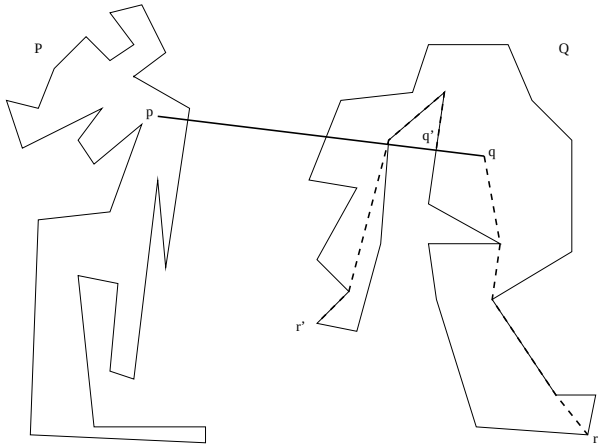

Fig. 2. There is an optimal bridge between boundary points of the polygons.

to the ones above and obtain a new optimal bridge that satisfies the conditions of the theorem. □

The previous theorem simplifies the optimal bridge problem because we only need to consider points on the boundary of the polygons to be the endpoints of optimal bridges. Note that as we point out in Section 2, the two boundary points may not be visible from each other as it is the case of the optimal v-bridge problem.

The next theorem limits the number of points on the boundary of the polygons which may be the endpoints of an optimal bridge. Before we establish this result we need to introduce additional notation.

In Figure 3 the furthest neighbors of point q inside Q are points r and r'. The thick dashed line segments indicate the furthest geodesic paths from q to r and the one from q to r'. As we traverse these paths starting at point q the first vertex of the polygon that we visit is called the *first-vertex* of the corresponding furthest point geodesic path. The line segment from q to the first-vertex is called the *first link*. In the case of Figure 3 the vertices a and a' are the first-vertices and the line segments (q, a) and (q, a') are called the corresponding *first-links*.

A point q on the boundary of Q is called an *anchor* if it is not a vertex of polygon Q, and there are at least two different vertices that are the first-vertex of a geodesic furthest path for q. We define anchors for polygon P in a similar way.

In Figures 3 and 4 we show two bridges. The dashed lines emanating from q in both of these figures indicate all furthest point geodesic paths. Point q in both of these figures is an anchor point. However, point q in Figure 5 is not an anchor point because all the geodesic furthest paths for point q have the same first-vertex, which is vertex a.

A point p on the boundary of P that is not a vertex of the polygon nor an

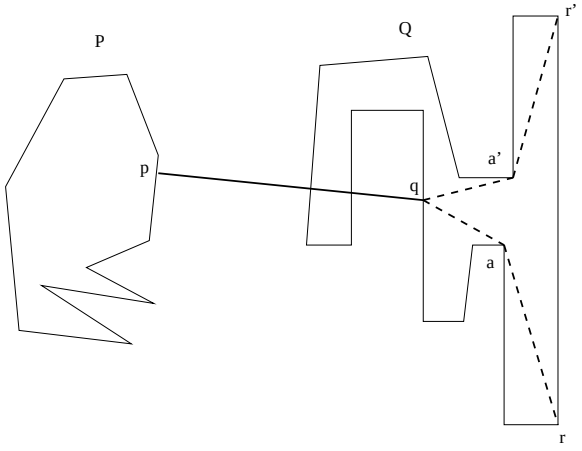

Fig. 3. Point q is an anchor.

anchor point is called a *pseudo-anchor* of P if there is a vertex x in P and a vertex or anchor y in Q such that p lies on the line (x, y), and it is the first point on the boundary of P hit by a ray originating at x in the direction (x, y). In other words, p is the point closest to x among all intersection points between (x, y) and P. We define pseudo-anchors for Q similarly by reversing the roles of P and Q.

The anchor points of P and Q can be identified from the geodesic furthest point Voronoi diagram which can be computed in $O(n \log n)$ time by the algorithm in Ref. 13. More specifically, these points correspond to the intersection of the boundary of the polygon and a Voronoi edge under the assumption that each vertex is not

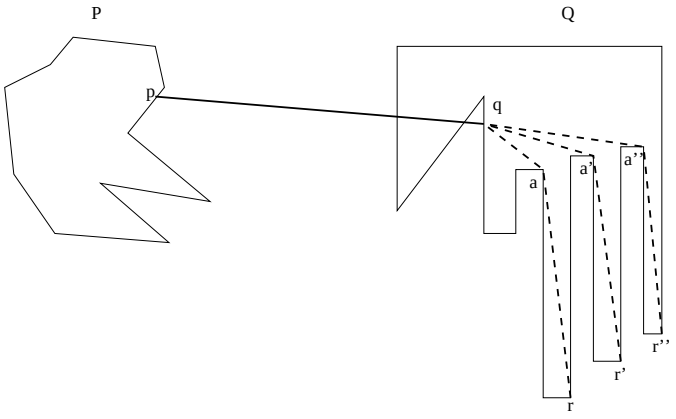

Fig. 4. Point q is an anchor.

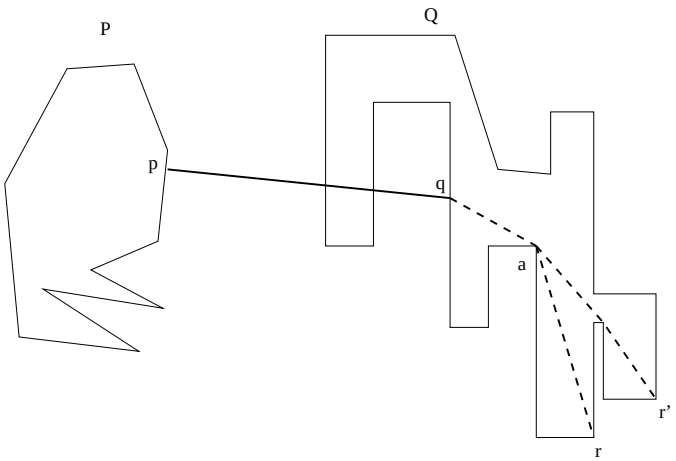

Fig. 5. Point q is not an anchor.

(geodesic) equidistant to two other vertices (note that this is not true for the point q in Figure 5). One way to avoid this condition is to perturb the vertices by moving them slightly. This "equidistant" condition is used to compute the Voronoi edges inside the polygon. However, we do not need to go this far because in the first part of of the algorithm in Ref. 13 the boundary of the polygon is partitioned into regions with the same set of geodesic furthest points. The juncture of any two of such regions are the anchors. There are only $O(n)$ such points. Note that this latter computation does not require the previous "equidistant" condition.

There are $O(n^2)$ pseudo-anchors and they can be computed in $O(n^2 \log n)$ time. Their identification is performed as follows. Since each polygon has $O(n)$ vertices and anchors, we need to consider $O(n^2)$ pairs of points (for each vertex of one polygon, we consider all the vertices and anchors from the other polygon). For each pair we need the first edge on each polygon they intersect. By preprocessing each polygon separately in $O(n \log n)$ time using Chazelle and Guibas technique[14] one may find the first place where each of the $O(n^2)$ lines intersect each of the polygons. Finding each of these (first) intersections takes $O(\log n)$ time[14]. Therefore the $O(n^2)$ pseudo-anchors which can be found in $O(n^2 \log n)$ time.

We now state an important theorem which forms the basis of our exact algorithm for the optimal bridge problem.

Theorem 2. *There is an optimal bridge whose endpoints are vertices, anchors, or pseudo-anchors from P and Q.*

Proof. Let $p \in P$ and $q \in Q$ be an optimal bridge. Suppose that at least one of p and q is not a vertex, anchor or pseudo-anchor of its polygon. We now establish a contradiction. Without loss of generality assume that p is not a vertex, anchor or pseudo-anchor, but point q may be one. Point p must be like point q in Figure 5, i.e., all the furthest geodesic paths for p have the same first-vertex.

Let us consider Figure 6(i) - (iii). Let a be the first-vertex for the furthest point path for vertex $p \in P$. Let γ be the clockwise angle for apq. There are three cases depending on the angle γ. Figure 6(i) - (iii) shows the three cases for point p. Let us now consider each case separately.

Case 1: Angle γ is less than 180 degrees (Figure 6(i)).
Clearly, in this case we can move p slightly higher to a point p^+. In this case the weight of bridge (p^+, q) is slightly smaller than the the one for (p, q) and therefore contradicts that (p, q) is an optimal bridge.

Case 2: Angle γ is greater than 180 degrees (Figure 6(ii)).
Clearly, in this case we can move p slightly lower to a point p^-. In this case the weight of bridge (p^-, q) is slightly smaller than the the one for (p, q) and therefore contradicts that (p, q) is an optimal bridge.

Case 3: Angle γ is equal to 180 degrees (Figure 6(iii)).
If q is a vertex or an anchor of Q, then p is a pseudo anchor of P, which contradicts our assumption. So, point q must be like in Figure 5, i.e., all

the furthest geodesic paths have the same first-vertex. Let b be the first-vertex for the furthest point path for vertex $q \in Q$. Let ζ be the clockwise angle for pqb. There are three cases depending on the angle ζ. Figure 6(iv) - (vi) shows these three cases for point q. Lets now consider the three cases separately.

Case 3.1: The angle ζ is less than 180 degrees (Figure 6(iv)).
Clearly, in this case we can move q slightly higher to a point q^+. In this case the weight of bridge (p, q^+) is slightly smaller than the one for (p, q) and therefore contradicts that (p, q) is an optimal bridge.

Case 3.2: Angle ζ is greater than 180 degrees (Figure 6(v)).
Clearly, in this case we can move q slightly lower to a point q^-. In this case the weight of bridge (p, q^-) is slightly smaller than the one for (p, q) and therefore contradicts that (p, q) is an optimal bridge.

Case 3.3: Angle ζ is equal to 180 degrees (Figure 6(vi)).
In this case there is a straight line from a first-vertex for p to a first vertex for q that includes p and q and by our definition p and q are pseudo-anchor points, a contradiction.

Therefore there is an optimal bridge whose endpoints are vertices, anchors or pseudo-anchors of P and Q. □

Note that a pseudo-anchor can support an optimal bridge only if it lies on the line connecting a vertex or anchor in the other polygon to its *first vertex* (in its own polygon). If this is not the case, then by the previous arguments we can get a better bridge by moving the point in some direction along the edge of the polygon containing it.

At first glance it appears from Figure 6 (vi), that if the line segment (p, q) is an optimal bridge, then so is the line segment (a, b). This would not contradict Theorem 2.1, but would imply a stronger version of Theorem 2.2 that would state that an optimal bridge exists whose endpoints are vertices or anchors. This new theorem would allow us to use a simple brute force $O(n^2)$ time algorithm to solve this problem by simply trying all bridges between vertices and anchors in the two different polygons. Unfortunately, it is not always true that if line segment (p, q) is an optimal bridge, then so is the line segment (a, b). The reason for this is that $gd(a, P)$ $(gd(b, Q))$ is not always equal to $gd(p, P) - d(a, p)$ $(gd(q, Q) - d(b, q))$. There are problem instances for which $gd(a, P) > gd(p, P) - d(a, p)$ $(gd(b, Q) > gd(q, Q) - d(b, q))$.

From Theorem 2.2 we can derive a brute force $O(n^2 \log n)$ time algorithm to find an optimal bridge. First we find all the anchor points ($O(n \log n)$ time). Then we consider every pair of points (vertices and anchors) from the two polygons (there are $O(n)$ such points). From each pair of points we generate at most 4 bridges as follows. Suppose the two points are points a and b in Figure 6 (vi). Find points p and q (Figure 6 (vi)) if they exist, otherwise let p be a and/or q be point b. Then

the four[a] bridges generated are (a, b), (a, q), (p, q) and (p, b). Now we compute the weight of each of these bridges as follows. The geodesic distance between a point and a polygon can be determined in $O(\log n)$ time once we have precomputed the geodesic-furthest-point Voronoi diagram for the polygon, and the distance between two points takes constant time. Then we just find the least weight bridge amongst all the $O(n^2)$ bridges generated. Therefore, the total time complexity is $O(n^2 \log n)$ time.

2.1. *Algorithm for the Optimal Vertex Bridge Problem and Optimal Bridge Problem*

In this section, we present an $O(n \log n)$ time algorithm for finding an optimal vertex bridge, which is a bridge with one endpoint being a vertex of P and the other a vertex of Q. Note that the endpoints of an optimal vertex bridge might not be *mutually visible*. In this section we use our $O(n \log n)$ time algorithm together with the property of optimal solutions that we derived in the previous subsection to produce another algorithm that generates an optimal bridge in $O(n^2 \log n)$ time.

We present an efficient algorithm that solves the optimal vertex bridge problem by using the *additively weighted nearest point* Voronoi diagram[15,16], which can be built in $O(n \log n)$ time. The technique yields an $O(n \log n)$ time algorithm for the optimal vertex bridge problem. In an additively weighted nearest point Voronoi diagram, each input point s has an associated weight w_s. If V is the set of input points, for any given point p in the plane, a weighted-nearest query returns a point

[a]Actually we can argue that if points p and q exist, only the bridge (p, q) needs to be considered since from Theorem 2.1 we know that $w(p, q) \leq w(a, b)$. However, to keep the discussion simple, we look at all the 4 bridges.

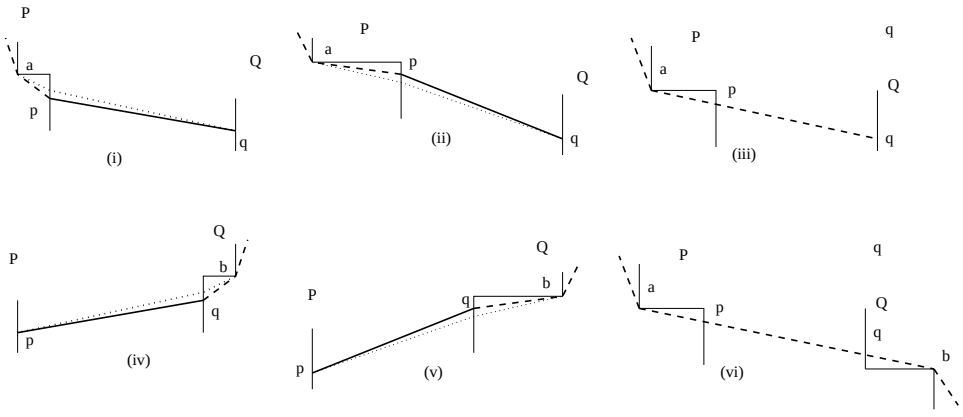

Fig. 6. Possible bridge configurations.

$v \in V$ such that

$$d(p, v) + w_v = MIN_{u \in V}\{d(p, u) + w_u\}$$

The additively weighted nearest point Voronoi diagram can be constructed in $O(n \log n)$ time given a set of n points and their associated weights. Furthermore, any query can be answered in $O(\log n)$ time. This data structure was used in Ref. 1 to obtain an $O(n^2 \log n)$ time algorithm for determining the minimum weight bridge connecting two convex polygons. Using this Voronoi diagram data structure, an optimal vertex bridge can be easily computed by procedure `OVB`.

To compute the weights associated with each point in the above Voronoi diagram, we compute the *furthest-geodesic-point* in P for each vertex of P, and in Q for each vertex of Q. This particular problem can be solved in $O(n \log n)$ time using Suri's algorithm[17].

`Procedure OVB(P,Q): simple polygons P, Q`

Compute the geodesic-furthest-point of each vertex in its polygon using Suri's algorithm[17].

For each vertex q of Q, set $w_q = gd(q, r)$ where $r \in Q$ is the geodesic-furthest-point of q.

Using the w_q values computed above, build an additively weighted nearest point Voronoi diagram[15,16] for the vertices of Q;

Using the above constructed data structure, for each vertex $p \in P$, find the vertex $q \in Q$ such that $gd(p, Q) = d(p, q) + w_q = MIN_{q' \in Q}\{d(p, q') + w_{q'}\}$;

The weight of the optimal bridge with one endpoint as $p \in P$ is $gd(p, P)+gd(p, Q)$, where $gd(p, P)$ is the geodesic distance between p and its geodesic-furthest point in P.

Choose the vertex of P which supports the bridge of minimal weight;

`End Procedure OVB`

Theorem 3. *Given two simple polygons P and Q, algorithm `OVB` finds an optimal vertex bridge between P and Q in $O(n \log n)$ time.*

Proof. It is clear that the distance from every vertex $q \in Q$ to its geodesic-furthest point in Q is computed and stored correctly in w_q by Suri's algorithm[17]. From the additively weighted nearest point Voronoi diagram our procedure computes for every vertex p, $gd(p, Q)$, which is defined as the minimum value of $\{d(p, q) + w_q\}$ for any vertex $q \in Q$. In other words, for every vertex p we have found the best possible vertex bridge with an endpoint at vertex p. In the last step one just finds the vertex p with least $gd(p, P) + gd(p, Q)$ which is the optimal vertex bridge.

Suri's algorithm for computing the geodesic furthest neighbors for all the points of a simple polygon takes $O(n \log n)$ time. Building both types of Voronoi diagrams (geodesic-furthest-point and additively-weighted-nearest-point) takes time

$O(n \log n)$ where $n = MAX(|P|, |Q|)$. Also, each weighted-nearest-point query and geodesic-furthest-point query takes $O(\log n)$ time. There are at most n queries of each type in the procedure OVB, thus taking a total time bounded by $O(n \log n)$. Thus, the overall time complexity of the algorithm OVB is $O(n \log n)$ time. □

The next corollary follows from this theorem, the properties of an optimal bridge established in the previous section, and the fact that the total number of vertices plus anchors plus pseudo-anchors is $O(n^2)$.

Corollary 1. Given two simple polygons P and Q, algorithm `OVB` when using as vertices all the vertices, anchors and pseudo-anchors of the polygons, finds an optimal bridge between P and Q in $O(n^2 \log n)$ time.

3. Approximation Schemes

In this section we present our approximation scheme for the optimal bridge problem. The idea is to introduce $k-1$ artificial vertices along each edge of the input polygons and then find an optimal *vertex* bridge. In the previous section we introduced an $O(n \log n)$ time algorithm to construct an optimal vertex bridge.

3.1. *Artificial Vertex Approximation Scheme*

Our artificial vertex approximation scheme is a simple extension of the algorithm OVB presented above. Given any integer $k \geq 1$, we partition each edge of the given polygons in k equal intervals by introducing artificial vertices. We then invoke the algorithm OVB with the input polygons having as vertices the original vertices plus the artificial ones. Thus, the input polygons now have $O(kn)$ vertices each. Thus, the algorithm OVB takes $O(kn \log kn)$ time to compute a bridge which has as endpoints either the original vertices of the polygons or the newly introduced artificial vertices.

`Procedure AV_OVB(P,Q,k): simple P, Q, int` $k > 0$
Introduce $k-1$ uniformly spaced points on each edge of the input polygons.
Call these polygons with the newly introduced artificial vertices P' and Q', respectively.
Output the result returned by $OVB(P', Q')$
`End Procedure AV_OVB`

Let us now establish our approximation bound for the artificial vertex approximation scheme.

Theorem 4. *Algorithm `AV_OVB(P,Q,k)` generates a bridge such that* $\hat{f}/f^* \leq 1 + \frac{2}{k}$ *in* $O(kn \log kn)$*, where* f^* *is the weight of the optimal bridge and* $\hat{f}$ *is the weight of the bridge computed by the procedure `AV_OVB()`.*

Proof. Assume (p, q) is an optimal bridge, with $p \in \delta(P)$ and $q \in \delta(Q)$ (see Figure 7). Let the optimal bridge have the point p lying on the edge (a, b) of P and q on the edge (c, d) of Q. Also, let u be the geodesic furthest neighbor of p, with $gd(p, u) = x$. Thus, the total weight of the bridge is $w(p, q) = d(p, q) + gd(p, u) + gd(q, Q) = d(p, q) + x + gd(q, Q)$.

Let us consider first the case when $k = 1$, i.e., there are no artificial vertices. The algorithm AV_OVB investigates the bridges through the points a and b of the polygon. We shall compare the weights of the bridges through these vertices to that of the optimal bridge.

Without loss of generality, assume that p is no farther from a than from b and q is no farther from c than from d. We investigate the bridge using a. The weight of the bridge (a, c) is $w(a, c) = gd(a, P) + d(a, c) + gd(c, Q)$.
Since $gd(p, P) = gd(p, u) = x$, we know that

$$d(p, a) = gd(p, a) \leq x$$

$$d(p, b) = gd(p, b) \leq x$$

$$\Rightarrow d(a, b) \leq 2x$$

If $gd(a, P) = gd(a, v)$, then using triangle inequality, we have

$$gd(a, P) = gd(a, v) \leq d(a, p) + gd(p, v) \leq 2x$$

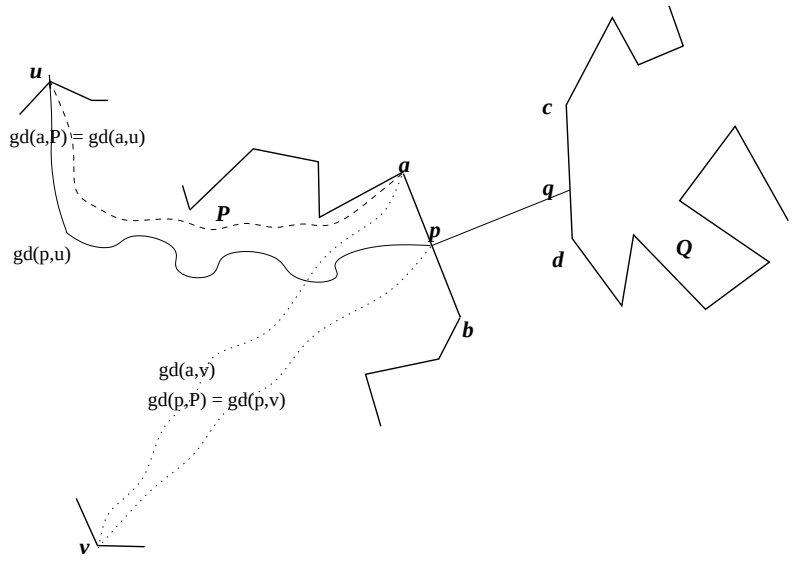

Fig. 7. Case when $k = 1$ and p is on the line ab. Note that $gd(a, u) \leq d(a, p) + gd(p, u) \leq d(a, p) + gv(p, v) \leq 2x$.

Similarly, $gd(c, Q) \leq 2y$, where $y = gd(q, Q)$ and (c, d) is the edge of the polygon Q which contains the point q. Applying the same arguments, we know that the length of the segment (a, c), is

$$d(a, c) \leq d(a, p) + d(p, q) + d(q, c) \leq x + d(p, q) + y.$$

Therefore, the weight of the bridge (a, c) is bounded by

$$w(a, c) \leq 2x + (x + d(p, q) + y) + 2y = 3x + d(p, q) + 3y$$

Hence, $\frac{w(a,c)}{w(p,q)} \leq \frac{3x+d(p,q)+3y}{x+d(p,q)+y} \leq 3$.

Let us now consider the case when $k > 1$. Extending the analysis for this case is straightforward (see Figure 8). Since $d(a, b) \leq 2x$, for any $1 \leq i \leq k-1$, $d(a_i, a_{i+1}) \leq \frac{2x}{k}$. The algorithm AV_OVB investigates the bridges through all the vertices of the polygons and all the artificial vertices introduced on the edges. In other words, if the optimal bridge (p, q) has p lying between (a_i, a_{i-1}) for some $1 \leq i \leq k-1$, the bridges through a_i and a_{i-1} would also be investigated. Proceeding in the fashion exactly as described for the case $k = 1$ above, we have

$$w(a_i, c_j) \leq (x + \frac{x}{k}) + (\frac{x}{k} + d(p, q) + \frac{y}{k}) + (\frac{y}{k} + y) = x(1 + \frac{2}{k}) + d(p, q) + y(1 + \frac{2}{k})$$

Thus the approximation ratio is

$$\frac{w(a_i, c_j)}{w(p, q)} \leq \frac{x(1 + \frac{2}{k}) + d(p, q) + y(1 + \frac{2}{k})}{x + d(p, q) + y} \leq 1 + \frac{2}{k}$$

Since procedure OVB takes $O(n \log n)$ time, where n is the number of vertices and the procedure AV_OVB's main work is invoking procedure OVB with $k * n$ points, it follows that its overall time complexity of AV_OVB is bounded by $O(kn \log kn)$. □

Figure 9 (b) shows the bridge generated by algorithm AV-OVB with $k = 1$ and Figure 9 (a) gives an optimal bridge for the problem instance. Figure 9 (d) shows the bridge generated by algorithm AV-OVB with $k = 2$ and Figure 9 (c) gives an optimal bridge for the problem instance. These problem instances can be easily generalized to problem instances for which our algorithm generates a solution with an approximation bound that can be made arbitrarily close to $1 + \frac{2}{k}$ for k even, and $1 + \frac{2}{k+1}$ for k odd. Therefore our approximation bounds are almost tight. Therefore the approximation bound we have established for our approximation scheme cannot be improved significantly.

3.2. *Improved Approximation Scheme*

In this subsection we present an improved approximation algorithm with approximation factor of $1 + \frac{2}{k+1}$. However, this bound might not be tight. The examples that achieve the approximation bound $1 + \frac{2}{k+1}$ given in the previous subsection, have a slightly smaller approximation bound when executed by the algorithm in this subsection.

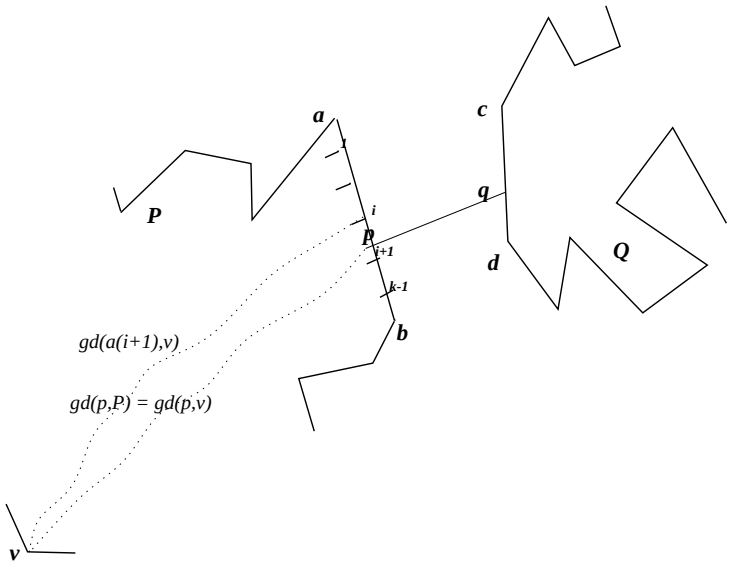

Fig. 8. General case: $k-1$ artificial vertices. Point p is in the interval (a_i, a_{i+1})

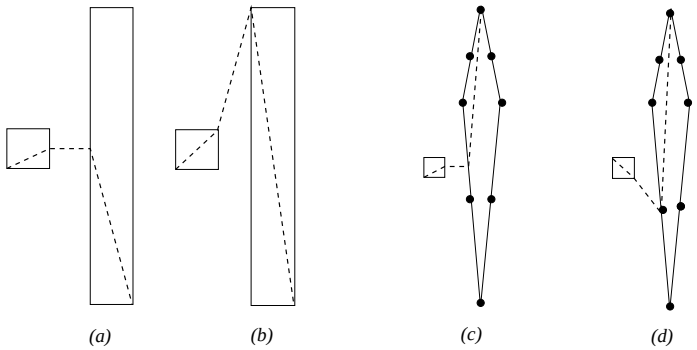

Fig. 9. Worst case example for the case when $k = 1$. (a) Optimal solution, and (b) Solution generated by AV-OVB with $k = 1$. Worst case example for the case when $k = 2$. (c) Optimal solution, and (b) Solution generated by AV-OVB with $k = 2$.

The main idea behind our algorithm is similar to the one in the previous subsection. The main difference is the length of the intervals. In the previous subsection, the intervals had uniform lengths. But the intervals close to the center of the line segment (a, b) have the largest approximation error when the optimal bridge includes a point inside the interval, whereas the ones closer to the end points have a smaller approximation error. The reason for this is that the bound for x (i.e., the distance to the geodesic-furthest point in the polygon) increases as we move from the center of the line segment (a, b) towards the extreme points, a and b. This is because the maximum distance from any point p to a or b increases as we move p from the center of the line to the extreme points, a and b.

Our strategy is to define the interval(s) closest to the center smaller than the ones in the previous subsection so as to decrease the approximation error when the optimal bridge has an endpoint inside the interval. Then we need to compensate for the remaining intervals, i.e, make the intervals larger, so that the total number of artificial points remain equal. Following the same lines for the analysis given in the previous subsection, it is sufficient to establish that for each of the k intervals, $d(a_{i-1}, a_i) \leq 2x/(k+1)$ where $x = gd(p, P)$ with p being the endpoint of the bridge on $\delta(P)$ and p located in the interval (a_{i-1}, a_i).

Let us consider first the case when the number of intervals k is even. The number of points is $m = k - 1$. Let $d = d(a, b)$. Let y be the point in the center of the line from a to b. Define $\delta_1 = 1/(k+1)$, $\delta_2 = 1/k$, $\delta_3 = 1/(k-1)$, and for $4 \leq i \leq (m-1)/2$ let $\delta_i = 1/k$. There is a point located at y, and for $1 \leq i \leq (m-1)/2$ there is a point at $y + \sum_{j=1}^{i} \delta_j * d$ and at $y - \sum_{j=1}^{i} \delta_j * d$. We shall refer to the algorithm AV_OVB that uses this Non-Uniform distanced artificial vertices as procedure NAV_OVB.

Lemma 1. *For each interval I as defined above, $d(I) \leq 2x/(k+1)$, where $x = gd(p, P)$ with p being the endpoint of the bridge on $\delta(P)$ and p located in the interval I.*

Proof. Since the intervals are symmetric, we only consider the intervals between the points located at $y - \sum_{j=1}^{i} \delta_j * d$ for $1 \leq i \leq (m-1)/2$ and a. There are four types of intervals I.

Case 1: Interval I from y to $y - \delta_1$.

Since vertex b is located at a distance $\geq d/2$ from every point in the interval, if the optimal bridge has an endpoint in this interval, we know that $x \geq d/2$ (or equivalently $d \leq 2x$). By definition the interval has length $d/(k+1)$. Substituting $d \leq 2x$ in the previous expression we know that the length of the interval is at most $2x/(k+1)$.

Case 2: Interval I from $y - \delta_1$ to $y - \delta_1 - \delta_2$.

Since vertex b is located at a distance $\geq d/2 + d/(k+1)$ from every point in the interval, we know that $x \geq d/2 + d/(k+1)$ (or equivalently, $d \leq 2(k+1)x/(k+3)$). By definition the interval has length d/k. Substituting $d \leq 2(k+1)x/(k+3)$ in the previous expression we know that the length of the interval is at most $2(k+$

$1)x/(k(k+3))$ which is at most $2x/(k+1)$ since $k \geq 1$.

Case 3: Interval I from $y - \delta_1 - \delta_2$ to $y - \delta_1 - \delta_2 - \delta_3$.
Since vertex b is located at a distance $\geq d/2 + d/(k+1) + d/k$ from every point in the interval, we know that Clearly $x \geq d/2 + d/(k+1) + d/k \geq d/2 + 2d/(k+1)$ (or equivalently, $d \leq 2(k+1)x/(k+5)$). By definition the interval has length $d/(k-1)$. Substituting $d \leq 2(k+1)x/(k+5)$ in the previous expression we know that the length of the interval is at most $2(k+1)x/((k+5)(k-1))$ which is at most $2x/(k+1)$ since $k \geq 2$.

Case 4: Interval I from $y - \sum_{j=1}^{i-1} \delta_j$ to $y - \sum_{j=1}^{i} \delta_j$, for $i \geq 3$.
The proof in this case is as in Case 2 since we know that $x > d/2 + d/(k+1)$. This completes the proof of the lemma. □

The proof when k is odd is similar, and for brevity we do not include it here. The overall approximation bound for both cases is established in the following theorem.

Theorem 5. *Algorithm* `NAV_OVB(P,Q,k)` *generates a bridge such that* $\hat{f}/f^* \leq 1 + \frac{2}{k+1}$ *in* $O(kn \log kn)$.

Proof. The proof follows from Theorem 4 and Lemma 1, and the discussion just before this theorem. □

4. Polynomial Time Approximation Scheme

In this section we present a polynomial time approximation scheme that given any constant $\epsilon > 0$ generates a bridge such that $\hat{f}/f^* \leq 1 + \epsilon$. The idea is similar to the one in the previous section but instead of having three different types of intervals, the length of the intervals vary, and are such that they generate an error bounded by ϵ when an optimal bridge has a point inside the interval. The interval(s) closest to the center of the line from a to b are the smallest. The length of the interval increases as they are located farther from the center. The reason why we selected three different types of intervals in the previous section, was because defining the length of these intervals to generate the same error for a given value of k is complex. The idea in this section is to allow the user to specify the maximum error which can be tolerated, and then we compute the number of intervals needed and hence the number of points to be introduced along each line.

Lets consider first the case when k is even. Let $d = d(a, b)$. Let y be the point in the center of the line segment from a to b. We will define the values $\delta_1, \delta_2, ...$ in terms of ϵ. There is a point at y as well as at $y + \sum_{j=1}^{i} \delta_j * d$ and at $y - \sum_{j=1}^{i} \delta_j * d$, for $i = 1, 2, \ldots$, until we pass points a and b. We define $\delta_i = \frac{\epsilon(1+\epsilon)^{(i-1)}}{2}$ for $i \geq 0$. We shall refer to the algorithm AV_OVB that uses artificial vertices defined in terms of ϵ as procedure AVe_OVB.

Lemma 2. *The length of each of the interval* $\delta_i * d$ *is at most* ϵx, *where* $x = gd(p, P)$ *with* p *being the endpoint of the bridge on* $\delta(P)$ *and* p *located in the current interval.*

Proof. Consider the ith interval from the center. Clearly $x \geq d/2 + \sum_{j=1}^{i-1} \delta_j d$ (or equivalently, $d \leq 2x/(1 + 2\sum_{j=1}^{i-1} \delta_j)$.

The length of the ith interval is $\delta_i d$. Substituting the above expression we know that the length of the interval is at most $2\delta_i x/(1 + 2\sum_{j=1}^{i-1} \delta_j)$.

Inductively one can establish that $1 + 2\sum_{j=1}^{i} \delta_j = (1 + \epsilon)^i$ because

$$1 + 2\sum_{j=1}^{i} \delta_j = 1 + 2\sum_{j=1}^{i-1} \delta_j + 2\delta_i = (1+\epsilon)^{i-1} + 2 * (\epsilon/2)(1+\epsilon)^{i-1} = (1+\epsilon)^i$$

Substituting $1+2\sum_{j=1}^{i} \delta_j = (1+\epsilon)^i$ as well as the definition for δ_i in $2\delta_i x/(1+2\sum_{j=1}^{i-1} \delta_j)$, we know that the length of the interval is at most $2\delta_i x/(1+2\sum_{j=1}^{i-1} \delta_j) = \epsilon x$.

This completes our proof of the lemma. □

Theorem 6. *Algorithm* `AVe_OVB(P,Q,k)` *generates a bridge such that* $\hat{f}/f^* \leq 1+\epsilon$ *in* $O(kn \log kn)$*, where* k*, the number of intervals in each line, is even. The time complexity of the algorithm is* $O(kn \log kn)$*, where* $k = 2 * \lceil \frac{1}{\log(1+\epsilon)} \rceil$.

Proof. The proof follows from Theorem 4 and Lemma 2, and the discussion just before this theorem.

Now, since $\epsilon > 0$, we know $\sum_{j=1}^{m} \delta_j$ is $S_m = ((1+\epsilon)^m - 1)/2$. The least value of m such that $S_m \geq 1/2$ indicates that to achieve the approximation error ϵ one needs $2(m-1)+1 = 2m-1$ points, with the intervals having length defined by $\delta_i * d$ for $1 \leq i \leq m-1$. Therefore, $m = \lceil 1/\log(1+\epsilon) \rceil$ and the total number of intervals $k = 2m = 2\lceil 1/\log(1+\epsilon) \rceil$. □

For the case when we introduce an odd number of intervals the situation is slightly different. For this case $\delta_i = \frac{\epsilon}{2+\epsilon}(1+\epsilon)^{i-1}$ and the artificial points are located at $y + 0.5\delta_1 d + \sum_{j=2}^{i} \delta_j * d$, and $y - 0.5\delta_1 d - \sum_{j=2}^{i} \delta_j * d$ for $i = 1, 2, ...$ until we pass points a and b. Remember that y is the center point in line ab.

For this case we can prove a lemma identical to Lemma 2 and establish the following theorem.

Theorem 7. *Algorithm* `AVe_OVB(P,Q,k)` *generates a bridge such that* $\hat{f}/f^* \leq 1+\epsilon$ *in* $O(kn \log kn)$*, where* k*, the number of intervals in each line, is odd. The time complexity of the algorithm is* $O(kn \log kn)$*, where* $k = 2 * \lceil 1 + \log(2+\epsilon)/\log(1+\epsilon) \rceil$ *+ 1.*

Proof. The proof is similar to Theorem 6, but follows the definitions in the above discussion. □

For some values of ϵ it is better to choose the number of intervals odd and for other cases it is better to choose it even. The following table lists, for a set of ϵ

values the least number of points needed to achieve the approximation bound ϵ as well as the δ_i values.

Table 1. Number of points needed to achieve the approximation bound ϵ

ϵ	$k-1$	δ_1	δ_2	δ_3	δ_4	δ_5	δ_6	δ_7	δ_8
0.6667	2	0.2500	0.4167	-	-	-	-	-	-
0.5000	3	0.2500	0.3750	-	-	-	-	-	-
0.4000	4	0.1667	0.2333	0.3267	-	-	-	-	-
0.3333	4	0.1429	0.1905	0.2540	-	-	-	-	-
0.2857	5	0.1429	0.1837	0.2362	-	-	-	-	-
0.2500	6	0.1111	0.1389	0.1736	0.2170	-	-	-	-
0.2222	6	0.1000	0.1222	0.1494	0.1826	-	-	-	-
0.2000	7	0.1000	0.1200	0.1440	0.1728	-	-	-	-
0.1818	8	0.0833	0.0985	0.1164	0.1376	0.1626	-	-	-
0.1667	9	0.0833	0.0972	0.1134	0.1323	0.1544	-	-	-
0.1538	9	0.0769	0.0888	0.1024	0.1182	0.1363	-	-	-
0.1429	10	0.0667	0.0762	0.0871	0.0995	0.1137	0.1300	-	-
0.1333	11	0.0667	0.0756	0.0856	0.0970	0.1200	0.1247	-	-
0.1250	11	0.0625	0.0703	0.0791	0.0890	0.1001	0.1126	-	-
0.1176	12	0.0556	0.0621	0.0694	0.0776	0.0867	0.0969	0.1083	-
0.1111	13	0.0556	0.0617	0.0686	0.0762	0.0847	0.0941	0.1045	-
0.1053	13	0.0526	0.0582	0.0643	0.0711	0.0785	0.0868	0.0959	-
0.1000	14	0.0476	0.0524	0.0576	0.0633	0.0697	0.0767	0.0844	0.0928
0.0952	15	0.0476	0.0522	0.0571	0.0626	0.0685	0.0751	0.0822	0.0900
0.0909	15	0.04545	0.0496	0.0541	0.0590	0.0644	0.0702	0.0766	0.0836

5. Concluding Remarks

In this paper we have presented an exact algorithm and approximation schemes for the optimal bridge problem, all of which are based on an efficient algorithm for the optimal *vertex* bridge problem. We did not report other investigations with a few other variations of the basic approximation schemes presented here. One of them is based on the idea of introducing a number of artificial points per line segment depending on the length of the segment. In general, this strategy allows the same approximation bound while reducing the overall time complexity bound.

Our approximation algorithms are amenable to efficient implementations, and the only external data structures and algorithms required are those for the *additively-weighted* nearest-point Voronoi diagram, and Suri's algorithm[17] for computing the geodesic-furthest neighbors of all vertices of a simple polygon. A geodesic-furthest-point Voronoi diagram[13], which is required for the exact algorithm, can also be used instead of Suri's algorithm[17].

Our approximation schemes can also be applied to the ag-bridge problem. To do this we need to compute $gd(p, q)$ for every pair of points belonging to different polygons. If $T_{gdsp}(n)$ represents the time required to build the tree of *geodesic shortest paths* from a point to a given set of $O(n)$ other points in presence of polygonal obsta-

cles in the plane, we can achieve the same approximation bounds for the ag-bridge problem in $O(k^2 \cdot nT_{gdsp}(n))$ time since the shortest paths tree needs to be computed for every vertex of the polygon P (or Q). The algorithm by Hershberger and Suri[18] builds the geodesic shortest paths tree of a point in $O(n \log n)$ time. This gives us an approximation of the ag-bridge problem in $O(k^2 n^2 \log n)$ time. Since the number of *obstacles* is fixed to 2 (the polygons P and Q) in our problem, the algorithm by Kapoor, et al.[19] can be used to build the tree in $O(n)$ time, thus producing an approximately optimal ag-bridge in $O(k^2 n^2)$ time. We should point out that results "similar" to the ones in Theorems 2.1 and 2.2 can be established for the ag-bridge problem. However, one cannot use the additively weighted nearest point Voronoi diagram to compute $gd(p, Q)$ because one uses $gd(p, q)$ rather than $d(p, q)$. But, if we replace this step by a brute force approach the whole algorithm would now take $O(n^4)$ time to solve the optimal ag-bridge problem between two polygons. An efficient data structure supporting additively-weighted *geodesic* nearest-point queries can significantly improve the time complexity. But unfortunately such data structure does not currently exist.

Another interesting generalization of our optimal bridge problem similar to the one introduced by Kim and Sin[4]. Given a set of points S of points in P and a set T of points in Q, find a bridge (p, q) such that

$$max_{s \in S}\{gd(p, s)\} + d(p, q) + max_{t \in T}\{gd(q, t)\},$$

is minimized. Our techniques can also be adapted to this problem to produce results similar to the ones reported in this paper.

With respect to the well-studied *visible* bridge problem, we have shown that a visible bridge has weight at most twice that of an optimal bridge. Furthermore, we've constructed problem instances for which this bound is the best possible.

Some of the interesting open problems suggested by our work include design of approximation schemes that do not use an optimal vertex bridge algorithm. Also, it would be challenging to improve the $O(n^2 \log n)$ bound for the exact solution of the problem. Another obvious open question is developing a faster algorithm for the *exact* solution of the ag-bridge problem, since the algorithm (mentioned above) uses brute-force in a critical step. As stated above, the improvement can also be achieved by an efficient data structure for additively-weighted-geodesic-nearest-point queries. The time complexity of the above procedure for approximating the ag-bridge problem is also pretty high, and admits a scope for improvement.

References

1. L. Cai, Y. Xu, and B. Zhu. Computing the optimal bridge between two convex polygons. *Information Processing Letters*, 69, 1999 127 – 130.
2. B. Bhattacharya and R. Benkoczi. On computing the optimal bridge between two convex polygons. *Information Processing Letters*, 79(5), 2001 215 – 221.
3. X. Tan. On optimal bridges between two convex regions. *Information Processing Letters*, 76, 2000 163 – 168.

4. S. K. Kim and C. S. Shin. Computing the optimal bridge between two polygons. *Theory of Computing Systems*, 34(4), 2001 337 – 352.
5. T. Tokuyama. Efficient algorithm for the minimum diameter bridge problem. *Lect. Notes Comput. Sci.*, 2098, 2001 362 – 369.
6. X. Tan. Finding an optimal bridge between two polygons. *International Journal of Computational Geometry and Applications*, 12(3), 2002 249 – 262.
7. D. P. Wang. An optimal algorithm for constructing an optimal bridge between two simple rectilinear polygons. *Information Processing Letters*, 79, 2001 229–236.
8. N. M. Amato. Determining the separation of simple polygons. *International Journal of Computational Geometry and Applications*, 4(4), 1994 457 – 474.
9. H.-K. Ahn, O. Cheong, and C.-S. Shin. Building bridges between convex regions. *Computational Geometry: Theory and Applications*, 25(1/2), 2003 161 – 170.
10. C. H. Papadimitriou. An algorithm for shortest path motion in three dimensions. *Information Processing Letters*, 20, 1985 259 – 263.
11. J. Choi, J. Sellen, and C. K. Yap. Approximate euclidean shortest path in 3-space. In *Proc. 10th annual symposium on Computational geometry*, , 1994 41 – 48.
12. J. Choi, J. Sellen, and C. K. Yap. Approximate euclidean shortest path in 3-space. *International Journal of Computational Geometry and Applications*, 7(4), 1997 271 – 295.
13. B. Aronov, S. Fortune, and G. Wilfong. The furthest-site geodesic Voronoi diagram. *Discrete Comput. Geom.*, 9, 1993 217 – 255.
14. B. Chazelle and L. J. Guibas. Visibility and intersecting problems in plane geometry. *Disc. & Comp. Geometry*, 4((6)), 1989 551 – 581.
15. F. Aurenhammer. Voronoi diagrams: A survey of a fundamental geometric data structures. *ACM Comput. Surveys*, 23(3), 1991, 343 – 405.
16. F. Aurenhammer and H. Imai. Geometric relations among Voronoi diagrams. In *Geom. Dedicata 27*, 1988, 65 – 75.
17. S. Suri. Computing geodesic furthest neighbors in simple polygons. *J. of Comp. and Sys. Sciences*, 39, 1989 220 – 235.
18. J. Hershberger and S. Suri. An optimal algorithm for euclidean shortest paths in the plane. *SIAM J. Comput.*, 28, 1999 2215 – 2256.
19. S. Kapoor, S. N. Maheshwari, and J. S. B. Mitchell. An efficient algorithm for Euclidean shortest paths among polygonal obstacles in the plane. *Discrete Comput. Geom.*, 18, 1997 377–383.