Applications of Parametric Searching in Geometric Optimization *

Pankaj K. Agarwal† Micha Sharir‡ Sivan Toledo§

Abstract

We present several applications in computational geometry of Megiddo's parametric searching technique. These applications include: (1) Finding the minimum Hausdorff distance under translation between two polygonal regions in the plane under the Euclidean metric; (2) Computing the biggest line segment that can be placed inside a simple polygon; (3) Computing the smallest width annulus that can contain a given set of points in the plane; (4) Solving the 1-segment center problem — given a set of points in the plane, find a placement for a given line segment (under translation and rotation) which minimizes the largest distance from the segment to the given points; (5) Given a set of n points in 3-space, finding the largest radius r such that if we place a ball of radius r around each point, no segment connecting a pair of points is intersected by a third ball. Besides obtaining efficient solutions to all these problems (which, in every case, either improve considerably previous solutions or are the first non-trivial solutions to these problems), our goal is to demonstrate the versatility of the parametric searching technique.

*Pankaj Agarwal has been supported by National Science Foundation Grant CCR-91-06514 Micha Sharir has been supported by Office of Naval Research Grant N00014-90-J-1284, by National Science Foundation Grant CCR-89-01484, and by grants from the U.S.–Israeli Binational Science Foundation, the G.I.F. — the German Israeli Foundation for Scientific Research and Development, and the Fund for Basic Research administered by the Israeli Academy of Sciences. Sivan Toledo has been supported by the Special Interdisciplinary Program at Tel-Aviv University.

†Department of Computer Science, Duke University

‡Courant Institute of Mathematical Sciences, New York University, and School of Mathematical Sciences, Tel Aviv University

§School of Mathematical Sciences, Tel Aviv University

0

1 Introduction

In this paper we present several applications in computational geometry of the *parametric searching* technique of Megiddo [32]. This technique, which we briefly review below, is a powerful and ingenious tool for solving efficiently a variety of optimization problems. Although it has been applied successfully to several problems in computational geometry [1, 3, 6, 18], its potential for problems in geometric optimization does not seem to be widely recognized as yet. Many problems of this kind, which could be easily attacked by the technique, are either solved by more complicated and more ad-hoc techniques, or are simply left unsolved. The purpose of this paper is to present efficient solutions, via the parametric searching technique, to several problems of this kind, with the by-product goal of publicizing the technique and making it more accessible to the computational geometry community.

The parametric searching technique can be described in the following general terms (which are not as general as possible, but suffice for our purposes). Suppose we have a decision problem $\mathcal{P}(d)$ that depends on a real parameter d, and is *monotone* in d, meaning that if $\mathcal{P}(d_0)$ is true for some d_0, then $\mathcal{P}(d)$ is true for all $d < d_0$. Our goal is to find the maximum d for which $\mathcal{P}(d)$ is true (or, if none exists, the supremum of all d for which $\mathcal{P}(d)$ is true). Suppose further that $\mathcal{P}(d)$ can be solved by a (sequential) algorithm $A_s(d)$ whose input is a set of data objects (independent of d) and d, and whose control flow is governed by comparisons, each of which amounts to testing the sign of some low degree polynomial in d. Megiddo's technique then runs A_s "generically" at the unknown maximum $d^\star$. Whenever A_s reaches a branching point that depends on some comparison with associated polynomial $p(d)$, it computes all its roots and runs A_s with the value of d equal to each of these roots. This yields an interval between two ad-

jacent roots, known to contain $d^\star$, and thus enables A_s to determine the sign of $p(d^\star)$, thus resolving the comparison and allowing the generic execution to proceed. As the algorithm proceeds, the interval known to contain $d^\star$ keeps shrinking as a result of resolving further comparisons, and at the end either the interval becomes a singleton, which is thus the desired $d^\star$, or else $d^\star$ can be shown to be equal to its upper endpoint.

The cost of the procedure just described is generally too high, because the number of times A_s in invoked within the generic execution is proportional to the number of comparisons in the generic A_s. To speed up the execution, Megiddo proposes to replace the generic algorithm by a parallel algorithm A_p. If A_p uses P processors and runs in T_p parallel steps, then each parallel step involves at most P *independent* comparisons. We can then compute roots of all polynomials associated with these comparisons, and perform a binary search to locate d^* among them using A_s at each binary step. If A_s has running time T_s, then the cost of simulating a parallel step of A_p is $P + T_s \log P$, for a total of $PT_p + T_pT_s \log P$. In most cases the second term dominates the running time. (Since the parallel algorithm is simulated sequentially, we can allow the comparison model of Valiant [38], which measures parallelism only in terms of comparisons being made, and ignores all other operations. This observation simplifies the technique considerably.)

This brief overview of parametric searching does not cover all aspects of the technique. Among the issues we left out are a trick due to Cole [17], which in certain cases improves the running time of the procedure by a logarithmic factor, a variant due to Matoušek [31] and others which replaces in certain applications the parallel generic algorithm by a randomized (sequential) one, leading to simplified solutions, and a variant due to Frederickson and Johnson [20, 21], where the optimal solution $d^\star$ is an element of an implicitly given matrix, whose elements satisfy certain monotonicity properties. There are various other extensions of the technique. In particular, Megiddo's subsequent linear-time algorithm for linear programming [34] can be regarded as an optimized variant of the parametric searching technique.

Since its design, about 8 years ago, the parametric searching technique have been successfully applied to a variety of optimization problems. In computational geometry it has been applied to the slope selection problem [18], computing the center of a set of points in 2 and 3 dimensions [35], selecting distances in the plane [1], certain 2-center problems for planar point sets [3], range searching and ray shooting [2], and extremal polygon containment problems [6]. This is still a relatively small crop, given the large body of literature on geometric optimization problems.

In this paper we demonstrate the power of the technique by applying it to solve a variety of additional geometric optimization problems. Loosely speaking, the recipe for such an application is first to solve the *fixed-size problem* (i.e. the decision problem $\mathcal{P}(d)$) by an efficient sequential algorithm and an efficient parallel one. Then the application of parametric searching is almost routine and yields efficient solution to the related optimization problem.

The problems that we solve in this paper are (see also the subsequent sections for additional discussion of the results and comparison with previous work):

- Computing the biggest line segment that can be placed inside a simple n-gon. We present a randomized algorithm with expected running time $O(n^{8/5+\epsilon})$, for any $\epsilon > 0$, considerably improving the previous algorithm of [15] whose running time is $O(n^{1.999878})$.[1]
- Computing the smallest-width annulus that contains a given set of n points in the plane. We give a randomized algorithm with expected running time $O(n^{8/5+\epsilon})$, for any $\epsilon > 0$, considerably improving the quadratic-time algorithm of [19].
- Finding the minimum Hausdorff distance under translation between two polygonal regions in the plane under the Euclidean metric. This is a hard instance of a general pattern matching problem. It was left untreated in [26], and solved by a brute-force inefficient method in [8]. We solve it in time $O((mn)^2 \log^3(mn))$, where m and n are the number of edges of the given polygons. This is about 3 orders of magnitude faster than the algorithm of [8].
- Solving the 1-segment center problem — given a set of n points in the plane, find a placement for a given line segment (under translation and rotation) which minimizes the largest distance from the segment to the given points. We present an algorithm for this problem whose time complexity is $O(n^2\alpha(n) \log^3 n)$. It improves the previous algorithm of Imai et al. [29] by roughly two orders of magnitude.

[1] Throughout this paper, ϵ denotes an arbitrarily small positive constant. The multiplicative constants in the asymptotic bounds may depend on ϵ.

- Solving the problem of complete mutual visibility among spheres — given a set of n points in 3-space, find the largest radius r so that if we place a ball of radius r around each point, no segment connecting a pair of points is intersected by a third ball. This problem arises in the context of optical interconnections between processors in 3-space. We present an $O(n^2 \log^5 n)$ algorithm for this problem, which, as far as we know, is the first nontrivial solution.

Although the common theme of our solutions is the application of parametric searching, the bulk of the technical contribution of this paper is in the solutions of the corresponding fixed-size problems, which are by no means easy. They require the application of a variety of sophisticated geometric techniques, such as range searching, point location among algebraic varieties, computing Minkowski sums, Davenport-Schinzel sequences and related polygon placement techniques, and output-sensitive hidden surface removal in 3-space. We also remark that the challenge is not only in solving these fixed-size problems efficiently by a sequential algorithm, but also to design efficient parallel algorithms (in Valiant's model) for these problems.

The paper is organized as follows. In Section 2 we present a solution to the biggest stick problem and the minimum-width annulus problem. Section 3 studies the problem of computing the minimum Hausdorff distance between two polygons. Section 4 considers the 1-segment center problem, and Section 5 solves the 3-D complete mutual visibility problem. For lack of space, we omit in this conference version some of the details, and refer the readers to the full version of the paper [6].

2 The Biggest Stick Problem

In this section we obtain an improved solution to a problem posed by M. McKenna in 1986: Given a simple polygon P with n edges, find the "biggest stick" (i.e. largest line segment) that can be placed inside P (i.e. be disjoint from the exterior of P). It is easy to design an algorithm for solving this problem in time $O(n^2)$, and the goal is to obtain subquadratic solutions. Chazelle and Sharir [15] have given such a subquadratic solution. It runs in time $O(n^{1.999878})$ and is based on Collins' cylindrical algebraic decomposition technique. In this section we give a considerably improved solution, whose running time is $O(n^{8/5+\epsilon})$, for any $\epsilon > 0$. (We note that if the endpoints of the stick are constrained to lie at vertices of P then a faster solution is known [7].)

Our solution is based on the following approach, also used by the previous algorithms mentioned above. We find a chord e that partitions P into two subpolygons, P_1, P_2, such that each contains roughly half the number of vertices. We recursively find the biggest stick in P_1 and in P_2. Then we compute the biggest stick within P which crosses e, and the final answer is the largest of these three candidate sticks.

To compute the biggest stick that crosses e we proceed as follows. Suppose without loss of generality that e is vertical and lies on the y axis. By using a standard duality transformation, we can map lines crossing e to points, and thus obtain two planar maps, M_1, M_2 in the dual plane, where each face of either map M_i is the locus of points dual to lines emanating from e and hitting first (the interior of) some fixed edge of P_i. It is easy to show that each M_i is a convex subdivision having $O(n)$ faces, edges and vertices, and that it can be computed in $O(n \log n)$ time; see [14].

Now let us fix a length $d > 0$, and consider the subproblem of determining whether a line segment of length d can be placed inside P so that it also intersects e. It is easy to show that if such a placement exists then there also exists a placement in which the segment either passes through two vertices of P or has one endpoint on e. Searching for sticks of length d that satisfy the latter condition can easily be done by examining each of the maps M_i separately. In the former case, if the two vertices through which the segment passes lie in the same subpolygon, say P_1, then the line containing the segment is dual to a vertex of M_1, and locating that vertex in M_2 immediately tells us whether such a placement exists. Thus we locate each vertex of M_1 in M_2 and vice versa, at an overall cost of $O(n \log n)$, thereby handling all the critical placements of this kind.

The hard case is when the two vertices through which the segment passes lie in different subpolygons. The corresponding line is then dual to a point of intersection between an edge of M_1 and an edge of M_2. The number of such intersections can be $\Theta(n^2)$ in the worst case, so we cannot afford to compute all of them explicitly. Instead, we preprocess the edges of one of the maps, say M_1, for efficient range searching queries of a particular kind (detailed below), and then query the resulting structure with range queries derived from the edges of M_2. These queries collectively determine whether there exists a critical placement of the segment with the required properties, thereby solving the fixed-size subproblem.

In more detail, every edge g of M_i corresponds to a pair (v, a), where v is a vertex and a is an edge of

P_i; the points of g are dual to lines ℓ which cross e, pass through v and hit a behind (or at) v, so that the portion of ℓ between its intersections with e and a, excluding the point v, lies in the interior of P_i; see Figure 1.

We regard each edge $g \in M_1$ as the xy-projection of an arc γ in 3-space, so that for each $x \in g$ the height of γ above x is equal to d minus the length of the portion of ℓ, the line dual to x, between its intersections with e and with a. In this manner, the edges of M_1 are mapped into a collection $\mathcal{G}$ of arcs in 3-space.

Next, we map each edge $h \in M_2$, associated with the pair (w, b), to an arc η in 3-space, so that the xy-projection of η is h and the height of η above any $x \in h$ is the length of the portion of the line dual to x between its intersections with e and with b. Let x be the intersection of h with an edge g of M_1. Then it is easily verified that the line ℓ dual to x contains a placement of a line segment with length d crossing e and lying inside P if and only if the arc η passes (at x) above the arc γ corresponding to g.

The problem has therefore been reduced to the following. Given a collection $\mathcal{G}$ of n arcs in 3-space, corresponding to the edges of M_1 in the manner described above, and a second collection $\mathcal{H}$ of n arcs, corresponding to the edges of M_2, determine whether there exists a pair of arcs, $\gamma \in \mathcal{G}$, $\eta \in \mathcal{H}$, such that η passes above γ.

We solve this problem in two stages. First, following the technique of [13], we construct a 'hereditary segment tree' on the x-projections of the edges of M_1 and M_2. In particular, we construct a segment tree $\mathcal{T}$ on the interval decomposition of the x-axis induced by the x-coordinates of the vertices M_1, M_2. The i^{th} leaf of $\mathcal{T}$ stores the i^{th} interval from the left. Each node $v \in \mathcal{T}$ is associated with an interval δ_v, the union of the intervals associated with the leaves of the subtree rooted at v. Let $I_v = \delta_v \times [-\infty, +\infty]$. An edge of M_1, M_2 is associated with a node v if it completely crosses I_v, but does not cross $I_{p(v)}$, where $p(v)$ is the parent of v. Let $M_1(v)$ denote the set of edges in M_1 associated with v, and let $M_1^*(v) = \bigcup_w M_1(w)$, where w is a descendeant of v (including v itself). For the sake of simplicity, we assume that the edges of $M_1(v), M_1^*(v)$ are clipped within I_v. Similarly, define $M_2(v)$ and $M_2^*(v)$. By construction

$$\begin{aligned}\sum_{v\in\mathcal{T}}(|M_1(v)| + |M_2(v)|) &\leq \sum_{v\in T}(|M_1^*(v)| + |M_2^*(v)|)\\ &= O(n\log n)\,.\end{aligned}$$

One can construct a family of $O(n_v)$ canonical subsets $M_{1,1}(v), M_{1,2}(v), \ldots$ of $M_1(v)$ such that the set of segments in $M_1(v)$ intersecting a segment of $M_2^*(v)$ can be represented as a union of $O(\log n_v)$ pairwise disjoint canonical subsets. Let $M_{2,i}^*(v) \subseteq M_2^*(v)$ be the set of segments h for which $M_{1,i}(v)$ is used to represent the set of segments in $M_1(v)$ intersected by h. Similarly define $M_{2,i}(v), M_{1,i}^*(v)$. It has been shown in [12, 3] that the intersection points of segments in $M_{1,i}(v), M_{2,i}^*(v)$, and in $M_{2,i}(v), M_{1,i}^*(v)$ contain all intersection points of M_1, M_2.

Consider one of the subsets, say $M_{1,i}(v), M_{2,i}^*(v)$. Since every pair of segments in $M_{1,i}(v), M_{2,i}^*(v)$ intersects, we can extend the segments to full lines. Let $\mathcal{G}_0$ and $\mathcal{H}_0$ denote the sets of arcs (actually curves) corresponding to the lines in $M_{1,i}(v)$ and $M_{2,i}^*(v)$, respectively; let $|\mathcal{G}_0| = n_0$, $|\mathcal{H}_0| = m_0$. We want to determine whether any arc of $\mathcal{G}_0$ lies above any arc of $\mathcal{H}_0$. Recall that each segment is associated with a vertex v and an edge e. So, each arc $\gamma \in \mathcal{G}_0$ can be defined by four real parameters — the coordinates v_1, v_2 of the corresponding vertex v and the coefficients a_1, a_2 of the line containing the corresponding edge a (the preceding filtering segment tree technique allows us to ignore the endpoints of a and to regard it as a full line). For each $\eta \in \mathcal{H}_0$ we can express the difference in height between η and γ, at the point of intersection between their xy-projections, as a function $F_\eta(\gamma) = F_\eta(v_1, v_2, a_1, a_2)$ of the four parameters defining γ, Our goal is thus to determine whether there exists $\gamma \in \mathcal{G}_0$ such that

$$\max_{\eta\in\mathcal{H}_0} F_\eta(\gamma) \geq 0\,. \tag{1}$$

Let Γ denote the set of surfaces $\{F_\eta(v_1, v_2, a_1, a_2) = 0 \mid \eta \in \mathcal{H}_0\}$ in $\mathbb{R}^4$. Since there is a unique point $\xi(v_1, v_2, a_1)$ on the surface $\xi \in \Gamma$ for every (v_1, v_2, a_1), (1) satisfies for an arc $\gamma \in \mathcal{G}_0$ if the corresponding point does not lie in the lower envelope of Γ. Chazelle et al. [12] have presented an $O(m^{5+\epsilon})$ size data structure for answering $O(\log m)$ point location queries among a collection of m algebraic surfaces (of fixed degree) in $\mathbb{R}^4$. Their structure relies on a scheme that triangulates the arrangement of m surfaces into roughly $O(m^5)$ constant size cells. Since we want to determine whether a point lies in the lower envelope of Γ, it suffices to preprocess only the lower envelope for point location queries, which in turn implies that it suffices to triangulate only the lower envelope of Γ. We show that the lower envelope of Γ can be triangulated in roughly $O(m^4)$ cells. Due to lack of space we briefly sketch the main idea; see the full version for details [6].

[1]: Micha, Am I right??

Let ξ be a surface of Γ. We project the intersection of ξ with other surfaces to hyperplane $a_2 = 0$, and tri-

angulate the resulting set of m surfaces into roughly $O(m^3)$ cells using the algorithm of [12]. Next, for each cell τ of the triangulation, if $\tau' = \{(x, \xi(x)) \mid x \in c\}$ appears on the lower envelope of Γ, we add the cell $\psi_c = \{(x, z) \mid x \in c, z \leq \xi(c)\}$ to the triangulation of the lower envelope of Γ. Repeating the above step for all surfaces, we obtain a triangulation of the lower envelope of Γ os size roughly $O(m^4)$. Using this triangulation scheme and folllowing the same technique as in [12], we can preprocess Γ in time $O(m^{4+\epsilon})$ into a data structure, so that one can determine in time $O(\log m)$ whether $\max_{\zeta \in \mathcal{H}_0} F_\zeta(\gamma) \geq 0$ for an arc $\gamma \in \mathcal{G}_0$. Thus the total time spent is $O(m^{4+\epsilon} + n \log m)$. By flipping the roles of $\mathcal{G}_0$ and $\mathcal{H}_0$, we can obtain another solution whose running time is $O(n_0^{4+\epsilon} + m_0 \log n_0)$.

To obtain a further improved solution, we use the technique of random sampling [16]: we choose a random sample of r surfaces F_η, for some constant parameter r, and triangulate their lower envelope into roughly $O(r^{4+\epsilon})$ cells, as described above. With high probability each cell intersects $O(\frac{m_0}{r} \log r)$ surfaces. This leads to a collection of subproblems, where the problem associated with cell c_i involves the surfaces that cross the cell and the points of $\mathcal{G}_0$ that fall in the cell. By solving each subproblem recursively, using the preceding technique to bootstrap the recursion, one can show that the expected running time of this step is

[2]: Use random sampling or deterministic algorithms.

$$O(m_0^{4/5+\epsilon} n_0^{4/5+\epsilon} + (m_0 + n_0)^{1+\epsilon})$$

for any $\epsilon > 0$. We omit the details of the analysis; similar analyses can be found in [4].

We next sum this bound over all subproblems. Since we have $\sum m_0 = \sum n_0 = O(n \log^2 n)$, we readily conclude that the overall running time of the algorithm is $O(n^{8/5+\epsilon'})$ for another, still arbitrarily small, $\epsilon' > 0$. That is, we have:

Theorem 2.1 *Given a simple polygon P with n edges and a length $d > 0$, one can determine whether a line segment of length d can be placed inside P, in time $O(n^{8/5+\epsilon})$.*

Next we apply parametric searching to turn the preceding algorithm into one that computes the biggest stick within P. For this we need an efficient parallel version of the above procedure. This is easy to do in Valiant's model — the divide-and-conquer process on P is parallelizable. Note that the actual decomposition of P can be done sequentially since it does not depend on d. Similarly, the maps M_1 and M_2, at each recursive step, can also be prepared outside the generic algorithm, as well as the hereditary segment tree decomposition. The step involving point location in 4-space amidst our surfaces F_η, which does depend on d, is easy to parallelize because the random sampling decomposition breaks down the problem into many independent subproblems, which can all be processed in parallel. Omitting some of the details, we can obtain a parallel algorithm that has overall parallel depth $O(\log^2 n)$, and uses $O(n^{8/5+\epsilon})$ processors. Plugging all of this into the parametric searching paradigm, we easily conclude:

Theorem 2.2 *Given a simple polygon P with n edges, one can compute the maximum length of a line segment that can be placed inside P, in time $O(n^{8/5+\epsilon})$.*

3 The Minimum Width Annulus

Next we consider the problem of approximating a planar point set by a circle. That is, suppose we are given a set S of n points in the plane and we want to approximate S by a circle. One way of obtaining such an approximation is to compute two concentric circles C_1 and C_2 of radii $r_1 < r_2$ such that all points of S lie in the exterior of C_1 and in the interior of C_2, and such that $r_2 - r_1$ is minimized. In other words, compute an *annulus* of minimum width that contains all points of S. An $O(n^2)$ algorithm was proposed by Ebara et al. [19]. We present an algorithm whose running time is $O(n^{8/5+\epsilon})$. As it turns out, this application is a variant of the technique used above for the biggest stick problem.

Specifically, let $\mathrm{Vor}_c(S), \mathrm{Vor}_f(S)$ be the closest and the farthest point Voronoi diagrams of S, respectively. For a point $\xi \in \mathbb{R}^2$ lying in the Voronoi cell $\mathrm{Vor}_c(p_i)$ ($p_i \in S$) of $\mathrm{Vor}_c(S)$, let $F_c(\xi)$ denote the distance between ξ and p_i. Analogously, define $F_f(\xi)$ for $\mathrm{Vor}_f(S)$. Given a point ξ in the plane, the width of the thinnest annulus centered at ξ, which covers S, is $F_f(\xi) - F_c(\xi)$. Thus, our goal is to compute

$$\min_{\xi \in \mathbb{R}^2} F_f(\xi) - F_c(\xi)\,.$$

As in the case of the biggest stick problem, it suffices to describe an algorithm that, for a given parameter W, can determine whether

$$\min_{\xi \in \mathbb{R}^2} F_f(\xi) - F_c(\xi) \leq W\,. \tag{2}$$

It has been shown in [19] that the desired minimum is attained either at a vertex of one Voronoi diagram or at an intersection of two diagram edges.

By preprocessing $\mathrm{Vor}_c(S), \mathrm{Vor}_f(S)$ for efficient planar point location queries, we can test in $O(n \log n)$ time whether any vertex of the two diagrams satisfies (2). So, the hard part is testing the intersection points of edges of the two diagrams.

We map each edge $g \in \mathrm{Vor}_c(S)$ to an arc γ in $\mathbb{R}^3$, whose xy-projection is g and whose height at a point $x \in g$ is $F_c(x)$. Let $\mathcal{G}$ denote the resulting set of edges. Similarly, we can regard each edge $h \in \mathrm{Vor}_f(S)$ as the xy-projection of an arc η in $\mathbb{R}^3$ whose height at $x \in h$ is $F_f(x) - W$. Let $\mathcal{H}$ denote the resulting set of arcs in $\mathbb{R}^3$. The problem now reduces to determining whether there exist two arcs $\gamma \in \mathcal{G}$, $\eta \in \mathcal{H}$ such that γ passes above η. Each edge of the two diagrams is a portion of a perpendicular bisector of two points in S, so each arc in $\mathcal{G}, \mathcal{H}$ can be defined by four parameters. Following the same approach as in the previous section, we can answer the last question in time $O(n^{8/5+\epsilon})$. We thus obtain

Theorem 3.1 *Given a set S of n points in the plane, one can determine, in time $O(n^{8/5+\epsilon})$, a minimum width annulus that contains all points of S.*

Remark 3.2: An annulus of minimum area can be computed in linear time using Megiddo's linear programming algorithm.

4 Minimum Hausdorff Distance Under Translation Between Polygonal Shapes

In this section, we consider the following problem: "Let $\mathcal{P}$ be a collection of m objects in the plane and $\mathcal{Q}$ another collection of n objects in the plane. We wish to compute a translation t of $\mathcal{Q}$ which minimizes the *Hausdorff distance* between $\mathcal{P}$ and $\mathcal{Q} \oplus t$, the translated copy of $\mathcal{Q}$." The Hausdorff distance between two sets A and B of objects is defined as

$$H(A, B) = \max\{h(A, B), h(B, A)\},$$

where

$$h(A, B) = \max_{p \in \cup A} \min_{q \in \cup B} d(p, q)$$

(we assume that the objects of A and B are all compact sets, so the minima and maxima appearing in this formula are all well defined). Here $d(\cdot, \cdot)$ denotes the Euclidean distance between two points. For a set $\pi \subseteq \mathbb{R}^2$ and a vector t let $\pi \oplus t = \{p + t \mid p \in \pi\}$ be the *Minkowski sum* of π and t, and for a set A of objects let $A \oplus t = \{\pi \oplus t \mid \pi \in A\}$. We want to compute

$$D(\mathcal{P}, \mathcal{Q}) = \min_{t \in \mathbb{R}^2} H(\mathcal{P}, \mathcal{Q} \oplus t) = \min_{t \in \mathbb{R}^2} H(\mathcal{P} \oplus t, \mathcal{Q}).$$

See Figure 2 for an illustration of the problem for the case where $\mathcal{P}$ and $\mathcal{Q}$ are simple polygons..

The value of $D(\mathcal{P}, \mathcal{Q})$ gives a measure of the resemblance between $\mathcal{P}$ and $\mathcal{Q}$, so its (efficient) computation has applications in pattern recognition, computer vision, etc. Huttenlocher and Kedem [25] showed that if $\mathcal{P}$ and $\mathcal{Q}$ are sets of points, then $D(\mathcal{P}, \mathcal{Q})$ can be computed in $O((mn)^2 \alpha(mn))$ time, where $\alpha(\cdot)$ is the inverse Ackermann function. This bound has been recently improved to $O(mn(m+n)\alpha(mn) \log mn)$ by Huttenlocher et al. [26]. They also showed that if the distance between two points is measured in the L_1 or L_∞ metrics, the distance $D(\mathcal{P}, \mathcal{Q})$, for sets $\mathcal{P}$, $\mathcal{Q}$ each consisting of non-intersecting segments, can be computed in time $O((mn)^2 \log mn)$. However, their algorithm does not extend to the more useful case of the Euclidean metric. For this case, Alt et al. [8] presented an algorithm with the rather high time complexity $O((mn)^3(m+n)\log(m+n))$.

In this section we show that if $\mathcal{P}$ and $\mathcal{Q}$ are sets each consisting of non-intersecting segments, then $D(\mathcal{P}, \mathcal{Q})$ can be computed in time $O((mn)^2 \log^3(mn))$. We first solve the fixed-size problem, which, given a parameter $\delta > 0$, determines whether $D(\mathcal{P}, \mathcal{Q}) \leq \delta$. We then convert this procedure, using the parametric search technique, into another algorithm that computes the value of $D(\mathcal{P}, \mathcal{Q})$.

We are thus given two sets $\mathcal{P}$, $\mathcal{Q}$, each consisting of non-intersecting segments, and a parameter $\delta > 0$, and we wish to determine whether $D(\mathcal{P}, \mathcal{Q}) \leq \delta$. Without loss of generality, we can assume that $\mathcal{P}$ is fixed and we seek a translation of $\mathcal{Q}$ which brings it within distance δ of $\mathcal{P}$. A placement of $\mathcal{Q}$ can be defined by the position of some fixed reference point $O_\mathcal{Q}$ rigidly attached to $\mathcal{Q}$. We assume that the original set $\mathcal{Q}$ is placed so that $O_\mathcal{Q}$ lies at the origin.

Let B_δ denote a disk of radius δ around the origin. For a segment e, let $e_\delta = e \oplus B_\delta$ be the Minkowski sum of e and B_δ. e_δ has the shape of a *racetrack* — a rectangle of width 2δ with two semicircles of radius δ attached to its sides. Let $\mathcal{P}_\delta = \bigcup_{e \in \mathcal{P}} e_\delta$ (see Figure 3). Since the relative interiors of the segments in $\mathcal{P}$ do not intersect each other, the boundaries of e_δ, e'_δ, for $e, e' \in \mathcal{P}$, intersect in at most two points (assuming general position of e, e' [28]; in any case, it is easy to show that the intersection of the boundaries of e_δ, e'_δ consists of at most two connected components). Therefore, by the result of [28], $\mathcal{P}_\delta$ has only $O(m)$

edges (and, symmetrically, $\mathcal{Q}_\delta$ has $O(n)$ edges); here each edge of $\mathcal{P}_\delta$ or of $\mathcal{Q}_\delta$ is either a straight segment or a circular arc.

For a set $A \subseteq \mathbb{R}^2$, let A^c denote the complement $\mathbb{R}^2 - A$. Let $K_{\mathcal{QP}} = \mathcal{P}_\delta^c \ominus \mathcal{Q}$ be the Minkowski difference of $\mathcal{P}_\delta^c$ and $\mathcal{Q}$.

Lemma 4.1 *$K_{\mathcal{QP}}^c$ is the set of translations t of $\mathcal{Q}$ for which $h(\mathcal{Q} \oplus t, \mathcal{P}) \leq \delta$.*

Proof: Let $O_\mathcal{Q} = t$ be a placement of $\mathcal{Q}$ for which $h(\mathcal{Q} \oplus t, \mathcal{P}) \leq \delta$. Then for every point $\zeta \in \mathcal{Q}$, there is a point $\xi \in \mathcal{P}$, such that $d(\zeta + t, \xi) \leq \delta$. In other words, $h(\mathcal{Q} \oplus t, \mathcal{P}) \leq \delta$ if and only if $(\mathcal{Q} \oplus t) \cap \mathcal{P}_\delta^c = \emptyset$. That is, there is no point $q \in \mathcal{Q}$ and a point p in the interior of $\mathcal{P}_\delta^c$ with $q + t = p$ or $t = q - p$. Hence, $t \in K_\Gamma^c$. □

Each edge of $K_{\mathcal{QP}}^c$ is contained in an arc of the form $z - q$, where z is an edge of $\mathcal{P}_\delta$ and q is an endpoint of a segment of $\mathcal{Q}$, or z is a vertex of $\mathcal{P}_\delta$ and q is a segment of $\mathcal{Q}$, or z is a point on a circular arc of $\mathcal{P}_\delta$ whose tangent is parallel to the segment q of $\mathcal{Q}$. Since $K_{\mathcal{QP}}^c$ is defined by $O(pq)$ segments and circular arcs, its combinatorial complexity is $O((pq)^2)$.

In order to define the set of translations t for which $h(\mathcal{P}, \mathcal{Q} \oplus t) \leq \delta$, we flip the roles $\mathcal{P}$ and $\mathcal{Q}$, i.e., we fix $\mathcal{Q}$ and define the set of placements t of $\mathcal{P}$ for which $h(\mathcal{P} \oplus t, \mathcal{Q}) \leq \delta$. By the preceding lemma, this set is $K_{\mathcal{PQ}}^c$, where $K_{\mathcal{PQ}} = \mathcal{Q}_\delta^c \ominus \mathcal{P}$. It now follows that

Lemma 4.2 *$D(\mathcal{P}, \mathcal{Q}) \leq \delta$ if and only if $K_{\mathcal{QP}}^c \cap (-K_{\mathcal{PQ}}^c)$ is not empty.*

In view of the above discussion, an algorithm for determining whether $D(\mathcal{P}, \mathcal{Q}) \leq \delta$ can be summarized as follows:

1. Compute $\mathcal{P}_\delta$ and $\mathcal{Q}_\delta$.
2. Compute $K_{\mathcal{QP}} = \mathcal{P}_\delta^c \ominus \mathcal{Q}$ and $K_{\mathcal{PQ}} = \mathcal{Q}_\delta^c \ominus \mathcal{P}$.
3. Determine whether $K_{\mathcal{QP}}^c$ and $-(K_{\mathcal{PQ}}^c)$ have nonempty intersection.

As for the time complexity of the algorithm, Step 1 can be accomplished in time $O((m+n)\log^2(m+n))$ using the algorithm of [28]. The set $K_{\mathcal{QP}}$ (and similarly $K_{\mathcal{PQ}}$) can be computed in time $O((mn)^2 \log mn)$ by constructing the entire arrangement of the arcs defining the edges of these sets. Finally, an intersection between $K_{\mathcal{QP}}^c$ and $-K_{\mathcal{PQ}}^c$ can be detected in $O((mn)^2 \log mn)$ time by a sweep-line algorithm [10]. Hence, we can conclude

Theorem 4.3 *Given a collection $\mathcal{P}$ of m non-intersecting segments and another collection $\mathcal{Q}$ of n non-intersecting segments in the plane, one can determine whether $D(\mathcal{P}, \mathcal{Q}) \leq \delta$ in time $O((mn)^2 \log(mn))$.*

Next, in order to apply parametric searching, we need an efficient parallel version of the algorithm. It is well known that the arrangement of a collection of t arcs in the plane can be computed in $O(\log t)$ time using $O(t^2)$ processors. Therefore the arrangement of $\{e \oplus B_\delta \mid e \in \mathcal{P}\}$ can be computed in $O(\log m)$ time with $O(m^2)$ processors. After having computed the arrangement, we determine for each face f the arrangement, the number of racetracks e_δ that contain f. Let c_f denote this quantity, then for two adjacent faces f, f' $|c_f - c_{f'}| = 1$. We compute the dual graph of the arrangement, and then a spanning tree of the dual graph, which we convert to an Eulerian path. Once we have an Eulerian path, we can compute c_f using any parallel prefix algorithm. See [1, 37] for details. The total time spent in computing c_f is $O(\log m)$ using $O(m^2)$ processors. It is easily seen that an edge γ of the arrangement is in $\mathcal{P}_\delta$ if $c_f = 0$ for one of the faces adjacent to γ. P_δ thus can be computed in $O(\log m)$ time using $O(m^2)$ processors. Similarly, one can compute $\mathcal{Q}_\delta^c$ in $O(\log n)$ time using $O(n^2)$ processors, and $K_{\mathcal{QP}}, K_{\mathcal{PQ}}$ in $O(\log mn)$ time using $O((mn)^2)$ processors. Finally, an intersection between $K_{\mathcal{QP}}^c$ and $-K_{\mathcal{PQ}}^c$ can be detected in $O(\log mn)$ time with $O((mn)^2)$ processors using the algorithm of Atallah et al. [9]. Applying the parametric search technique to the resulting algorithm, we can thus conclude

Theorem 4.4 *Given a collection $\mathcal{P}$ of m non-intersecting segments and another collection $\mathcal{Q}$ of n non-intersecting segments in the plane, one can determine the minimum Hausdorff distance between $\mathcal{P}$ and $\mathcal{Q}$, allowing translation, in time $O((mn)^2 \log^3(mn))$.*

5 Computing a Segment Center

The problem considered in this section is: "Given a set S of n points in the plane, and a segment e of length 1, find a placement of e (allowing translations and rotations) which minimizes the maximum (Euclidean) distance from e to the points of S."

The problem was posed by D.T. Lee a few years ago. It generalizes the well known notions of a point center (which is the center of the smallest enclosing disk of S [33]) and of a line-center. Finding the point center and the line center of a set S is easy;

the segment-center problem appears to be more difficult. Using parametric searching, we present in this section an algorithm for solving this problem which runs in close to quadratic time. This is a significant improvement over the previous algorithm by Imai et al. [29], whose time complexity is $O(n^4 \log n)$.

For lack of space we omit most of the details, and sketch the main ideas of the algorithm. First we solve the fixed-size problem: Given S and e as above and a real parameter $d > 0$, determine whether there exists a placement of e so that all points of S lie within distance d from e. To solve this problem, we define $e_d = e \oplus B_d$ to be an expansion of e by distance d (see also the previous section); e_d has the shape of a racetrack. Instead of moving e about, we fix e and move s rigidly. It is easily seen that the problem reduces to that of determining whether $P \equiv conv(S)$ can be placed (by translations and rotations) inside e_d; see Figure 4.

The problem has thus been reduced to a *polygon containment* problem, such as those studied in [6, 11, 30], with the twist that the environment e_d in which P has to be placed is not polygonal. Nevertheless, techniques similar to those used in [30, 36], which compute all *critical free placements* of P, can be developed.

Without loss of generality we assume that e_d is placed such that e lies on the x-axis and its enpoints are at $(0, 0)$ and $(1, 0)$.

If P can be placed inside e_d, we can translate it in the positive x-direction until one of its vertices touches the right semicircle C_R on the boundary of e_d. It therefore suffices to seek placements of P inside e_d at which one of the vertices of P touches C_R.

We fix a vertex p_i of P and constrain it to touch C_R, thereby leaving P with only two degrees of freedom — translation of p_i along C_R and rotation. Let O be some fixed point in P such that p_i is the rightmost vertex of P when the segment p_iO is horizontal. Each constrained placement of P can be parametrized by the two parameters (y, θ), where y is the y-coordinate of p_i and θ is the angle between the x-axis and the segment p_iO; θ is called the *orientation* of P. The subproblem we wish to solve is to determine whether there exists a constrained placement of P fully inside e_d.

Now fix a contact point $v = (x, y)$ of p_i with C_R, and rotate P around v. The other vertices of P trace concentric circles about v. It is easy to verify that, for each vertex p_j, there is a (possibly empty) unique angular interval $I_j = I_j(y)$, within which p_j is inside e_d. Hence there exists a placement of P inside e_d with p_i touching C_R at v if and only if the intersection $I = I(y)$ of all the angular intervals I_j is nonempty; since each of these intervals is less than π, as is easily checked, the intersection I is a single (possibly empty) angular interval. For each vertex p_j of P, the endpoints of the interval $I_j(y)$ are functions of y, and we denote the clockwise (resp. counter-clockwise) endpoint by $CW_{p_i,p_j}(y)$ (resp. $CCW_{p_i,p_j}(y)$.) Note that these functions may only be partially defined.

Define the following upper and lower envelopes:

$$U_{p_i}(y) = \max_j \{CW_{p_i,p_j}(y)\}, \quad (3)$$

$$L_{p_i}(y) = \min_j \{CCW_{p_i,p_j}(y)\}, \quad (4)$$

where we adopt the convention that whenever $CW_{p_i,p_j}(y)$ and $CCW_{p_i,p_j}(y)$ are undefined their values are set to $\frac{3\pi}{2}$ and $\frac{\pi}{2}$, respectively. Our conventions ensure that $U_{p_i}(y) > L_{p_i}(y)$ whenever some CW_{p_i,p_j} or $CCW_{p_i,p_j}(y)$ is not defined.

Lemma 5.1 *The graph of U_{p_i}, L_{p_i} have $O(\lambda_4(n))$ breakpoints.*

See the full version for a proof of the above lemma. In view of the above lemma, U_{p_i}, L_{p_i} can be computed by a sequential algorithm in in $O(n\lambda_3(n) \log n)$ time [24], or by a parallel algorithm in time $O(\log n)$ using $O(\lambda_4(n))$ processors. It has been shown in [23, 5] that $\lambda_3(n) = \Theta(n\alpha(n))$ and $\lambda_4(n) = \Theta(n2^{\alpha(n)})$. After having computed U_{p_i}, L_{p_i}, a 'good' point can also be determined within the same time bound. Hence, the fixed size problem can be solved in time $O(n^2\alpha(n) \log n)$ time, or in $O(\log n)$ parallel time using $O(n^2 2^{\alpha(n)})$ processors. Plugging all this into the parametric searching paradigm, we obtain:

Theorem 5.2 *Given a set S on n points in the plane, and a line segment e, we can compute a center location for e, which minimizes the maximum distance from e to the points of S, in time $O(n^2\alpha(n) \log^3 n)$.*

6 Complete Mutual Visibility Among Spheres

Let $\mathcal{S} = \{S_1, \ldots, S_n\}$ be a given set of n spheres in $\mathbb{R}^3$, all with the same radius. Let c_i denote the center of S_i, for $i = 1, \ldots, n$. We say that two spheres S_i and S_j are *mutually visible* if the line segment connecting c_i with c_j does not intersect any other sphere. We say that $\mathcal{S}$ is *completely mutually visible* if every pair of spheres in $\mathcal{S}$ is mutually visible.

The problem studied in this section is: "Given a set P of n points in $\mathbb{R}^3$, we wish to determine the largest possible common radius r such that the set of n spheres of radius r, centered at the given points, is completely mutually visible."

This problem arises in the context of parallel computations using optical interconnections. The spheres model the individual processors, and mutual visibility between a pair of spheres models the ability of the two processors to communicate by an optical link. In our problem the locations of the (centers of the) processors is predetermined, and we want to determine how large can the processors be if every pair is to be able to communicate optically.

In order to employ the parametric searching technique, we solve the following fixed-size decision problem: "Given a set of n spheres with the same radius r, determine if it is completely mutually visible".

We use the following simple scheme. For each of the spheres S_i, we compute the *visibility map* of all the other spheres, as viewed from its center c_i. We then check that all the centers are visible from c_i. This is true for each of the n visibility maps, if and only if the set of spheres is completely mutually visible.

The visibility maps can in general have rather high (up to quadratic) combinatorial complexity, but we establish the following interesting property (whose proof is omitted):

Lemma 6.1 *Let $\mathcal{S}$ be a collection of n congruent (nonintersecting) spheres in 3-space, and let p be a point outside the spheres. If the centers of all spheres are visible from p then the complexity of the visibility map of the spheres, as seen from p, is $O(n)$.*

The idea is now to compute the visibility map of the spheres from each center c_i, using the output-sensitive hidden surface removal algorithm of [27]. The algorithm runs in time $O((n+k)\log^2 n)$, where k is the combinatorial complexity of the visibility map. If the algorithm runs for too long, we stop it and conclude, in view of the preceding lemma, that not all centers are visible from c_i. Otherwise the algorithm terminates and then we check whether all centers are indeed visible. This takes, over all visible maps, a total time of $O(n^2 \log^2 n)$.

In order to apply the parametric search technique, we need a parallel implementation of the hidden surface removal algorith. Based on the sequential algorithm of [27], we can develop a parallel algorithm whose running time is $O(\log^2 n)$ using $O(n \log n + k)$ processors; see [6] for details. We compute visibility maps from all centers in parallel. We allocate $O(n)$ processors to each center. If any of the procedures require more than the allocated processors, we stop it and conclude that all centers are not mutually visible. If all the visibility maps have linear size, we preprocess each of them for point location queries, and then locate the centers in each of them. This can be done in $O(\log^2 n)$ using $O(n^2 \log n)$ processors. Omitting all the details, which can be found in the full version, we state the final result:

Theorem 6.2 *Given a set of n points in $\mathbb{R}^3$, we can find the largest possible common radius r, so that the system of spheres with radius r about the given points are completely mutually visible, in time $O(n^2 \log^5 n)$.*

References

[1] P.K. Agarwal, B. Aronov, M. Sharir and S. Suri, Selecting distances in the plane, *Proc. 6th ACM Symp. on Computational Geometry,* 1990, 321–331. (Also to appear in *Algorithmica.*)

[2] P.K. Agarwal and J. Matoušek, Ray shooting and parametric search, Tech. Rept. CS-1991-21, Dept. Computer Science, Duke University, 1991.

[3] P.K. Agarwal and M. Sharir, Planar geometric location problems, Tech. Rept. 90-58, DIMACS, Rutgers University, August 1990. (Also to appear in *Algorithmica.*)

[4] P.K. Agarwal and M. Sharir, Counting circular arc intersections, *Proc. 7th ACM Symp. on Computational Geometry,* 10–20.

[5] P.K. Agarwal, M. Sharir and P. Shor, Sharp upper and lower bounds for the length of general Davenport-Schinzel sequences, *J. Combin. Theory, Ser. A* 52 (1989), 228–274.

[6] P.K. Agarwal, M. Sharir and S. Toledo, Applications of parametric searching in geometric optimization, manuscript, 1991.

[7] A. Aggarwal and S. Suri, The biggest diagonal in a simple polygon, *Information Processing Letters* 13–18.

[8] H. Alt, B. Behrends and J. Blomer, Approximate matching of polygonal shapes, *Proc. 7th ACM Symp. on Computational Geometry*, 1991, 186–193.

[9] M. Atallah, R. Cole and M. Goodrich, Cascading divide-and-conquer: A technique for designing parallel algorithms, *SIAM J. Computing* 18 (1989), 499–532.

[10] J.L. Bentley and T. Ottmann, Algorithms for reporting and counting geometric intersections, *IEEE Trans. on Computers* C–28 (1979), 643–647.

[11] B. Chazelle, The polygon containment problem, in *Advances in Computing Research, Vol. I: Computational Geometry,* (F.P. Preparata, Ed.), JAI Press, Greenwich, Connecticut (1983), pp. 1–33.

[12] B. Chazelle, H. Edelsbrunner, L. Guibas and M. Sharir, A singly exponential stratification scheme for real semi–algebraic varieties and its applications, to appear in *Theoretical Computer Science.*

[13] B. Chazelle, H. Edelsbrunner, L. Guibas and M. Sharir, Algorithms for bichromatic line segment problems and polyhedral terrains, submitted to *Algorithmica.*

[14] B. Chazelle and L. Guibas, Visibility and intersection problems in plane geometry, *Discrete Comput. Geom.* 4 (1989), 551–589.

[15] B. Chazelle and M. Sharir, An algorithm for generalized point location and its applications, *J. Symbolic Computation* 10 (1990), pp. 281–309.

[16] K. Clarkson and P. Shor, Applications of random sampling in computational geometry, II, *Discrete Comput. Geom.* 4 (1989), 387–422.

[17] R. Cole, Slowing down sorting networks to obtain faster sorting algorithms, *J. ACM* 31 (1984), 200–208.

[18] R. Cole, J. Salowe, W. Steiger and E. Szemerédi, Optimal slope selection, *SIAM J. Computing* 18 (1989), 792–810.

[19] H. Ebara, N. Fukuyama, H. Nakano and Y. Nakanishi, Roundness algorithms using the Voronoi diagrams, *First Canadian Conf. Computational Geometry*, 1989.

[20] G. Frederickson, Optimal algorithms for tree partitioning, *Proc. 2nd ACM-SIAM Symp. on Discrete Algorithms*, 1991, 168–177.

[21] G. Frederickson and D. Johnson, Finding the kth shortest paths and p-centers by generating and searching good data structures, *J. Algorithms* 4 (1983), 61–80.

[22] M. Goodrich, Approximation algorithms to design parallel algorithms that may ignore processor allocation, *Proc. 32nd Annual IEEE Symp. Foundations Computer Science*, 1991, pp. 711–722.

[23] S. Hart and M. Sharir, Nonlinearity of Davenport-Schinzel sequences and of generalized path compression schemes, *Combinatorica* 6 (1986), 151–177.

[24] J. Hershberger, Finding the upper envelope of n line segments in $O(n \log n)$ time, *Inf. Proc. Lett.* 33 (1989), 169–174.

[25] D. Huttenlocher and K. Kedem, Efficiently computing the Hausdorff distance for point sets under translation, *Proc. 6th ACM Symp. on Computational Geometry*, 1990, 340–349.

[26] D. Huttenlocher, K. Kedem and M. Sharir, The upper envelope of Voronoi surfaces and its applications, *Proc. 7th ACM Symp. on Computational Geometry*, 1991, 194–203.

[27] M.J. Katz, M.H. Overmars and M. Sharir, Efficient hidden surface removal for objects with small union size, *Proc. 7th ACM Symp. on Computational Geometry*, 1991, 31–40.

[28] K. Kedem, R. Livne, J. Pach and M. Sharir, On the union of Jordan regions and collision-free translational motion amidst polygonal obstacles, *Discrete Comput. Geom.* 1 (1986), 59–71.

[29] H. Imai, D.T. Lee, and C. Yang, 1-Segment covering problem, *1st Canadian Conf. Computational Geometry*, 1989.

[30] D. Leven and M. Sharir, On the number of critical free contacts of a convex polygonal object moving in two-dimensional polygonal space, *Discrete Comput. Geom.* 2 (1987), 255–270.

[31] J. Matoušek, Randomized optimal algorithm for slope selection, manuscript, 1991.

[32] N. Megiddo, Applying parallel computation algorithms in the design of serial algorithms, *J. ACM* 30 (1983), 852–865.

[33] N. Megiddo, Linear-time algorithms for linear programming in $\mathbb{R}^3$ and related problems, *SIAM J. Computing*12 (1985), 720–732.

[34] N. Megiddo, Linear programming in linear time when the dimension is fixed, *J. ACM* 31 (1984), 114–127.

[35] N. Naor and M. Sharir, Computing the center of a point set in three dimensions, *Proc. 2nd Canadian Conf. on Computational Geometry* (1990), pp. 10–13.

[36] M. Sharir and S. Toledo, Extremal polygon containment problems, manuscript, 1991. (See also S. Toledo, Extremal polygon containment problems, *Proc. 7th ACM Symp. on Computational Geometry,* 176–185.)

[37] R. Tarjan and U. Vishkin, An efficient parallel biconnectivity algorithm, *SIAM J. Comp.* 14 (1985), 862-874.

[38] L. Valiant, Parallelism in comparison problems, *SIAM J. Computing* 4 (1975), 348–355.