The Visual Computer

Fast algorithms for computing the diameter of a finite planar set*

Binay K. Bhattacharya [1] and
Godfried T. Toussaint [2]

[1] School of Computing Science, Simon Fraser University, Burnaby, B.C., Canada, V5A1S6
[2] School of Computer Science, McGill University, Montreal, Quebec, Canada, H3A2K6

Three algorithms for computing the diameter of a finite planar set are proposed. Although all three algorithms have $O(n^2)$ worst-case running time, an expected-complexity analysis shows that, under reasonable probabilistic assumptions, all three algorithms have linear expected running time. Experimental results indicate that two of these algorithms perform very well for some distributions, and are competitive with an existing method. Finally, we exhibit situations where these exact algorithms out-perform a published approximate algorithm.

Key words: Diameter – Expected-complexity – Monte Carlo simulation – Approximate algorithm

* Research of the first author was supported by grant NSERC A 2422. Research of the second author was supported by grants NSERC A 9293, FCAC EQ-1678 and a Killam Senior Research Fellowship awarded by the Canada Council

1 Introduction

Let $S=\{p_1, p_2, \ldots, p_n\}$ be a set of n points in the plane. Let each point p_i be identified by the cartesian coordinates x_i and y_i. The diameter of the set S, denoted by DIAM(S), is defined by

$$\mathrm{DIAM}(S)=\max\{d(p_i, p_j)\}$$

where $d(p_i, p_j)$ is the euclidean distance between the points p_i and p_j.
In the clustering problem [15] it is often required to find the pair of points of a set which are farthest apart. This is precisely the diameter of the set. The diameter of a defect in an artillery shell is the crucial feature in the automatic inspection of radiographs of artillery shells [16]. In statistics, the bivariate range of a set is defined as the diameter of the set [10]. The bivariate range is used by applied ballisticians as a rapidly computed measure of the "precision" of small arms fire [10]. Finally, the diameter is used as a measure of the length of biological objects in morphology [22].
One obvious way of computing DIAM(S) is by the brute-force method. Here one first computes all the interpoint distances of the set S. Thus we have to compute $O(n^2)$ distances and therefore, the brute force method requires $O(n^2)$ running time.
Shamos [20] presented an algorithm to compute DIAM(S) based on the fact that the pair of points realizing DIAM(S) belong to the convex hull of S, i.e., DIAM(S) = DIAM($CH(S)$) where $CH(S)$ is the convex hull of S. There are many $O(n \log n)$ worst-case algorithms to compute the convex hull of S [1, 4, 23]. Under some mild conditions, the expected running time of the algorithm in [4] is $O(n)$. Once $CH(S)$ is computed, the diameter of the resulting convex polygon can be computed in linear time by first recognizing the "antipodal" pairs of points [5, 20]. Several linear time algorithms have been proposed to compute the diameter of a convex polygon [9, 14, 21]. However these algorithms have been shown to be incorrect [2, 6]. Shamos proposed another diameter algorithm which uses the "furthest point" Voronoi diagram of S [19]. However, it has been shown through a counterexample that this algorithm does not always work [8]. Snyder et al. [21] proposed a diameter algorithm which has $O(n^2)$ worst-case running time but, as claimed by the authors, has a fast expected-case running time. Firschein et al. [16] and Fischler [17] presented an $O(mn)$ algorithm to approximate the diameter of S where m is the number of directions considered. The maximum percentage error in estimating the length of the true diameter

is $100\left(1-\cos\left(\frac{90°}{m}\right)\right)\%$. The approximation gets better as m increases.

From the above discussion we see that the diameter of S can be computed by the algorithm in [20] in $O(n \log n)$ time in the worst case and in $O(n)$ time in the expected case under some mild conditions on the underlying probability distributions.

In this paper, the emphasis is on designing and comparing algorithms which have fast expected running times. In Sect. 2, we describe three new algorithms after developing the necessary background. The running times, both in the worst case and the expected case, are investigated. Section 3 contains a Monte Carlo simulation to empirically compare the performances of the proposed algorithms, along with the algorithms of Shamos (called here SHAMOS) [20] and Snyder et al. (called here SNYDER) [21].

2 Proposed algorithms

2.1 Definitions

A point $p_i=(x_i, y_i)$ *dominates* a point $p_j=(x_j, y_j)$ in (s_1, s_2) sense if $s_1 x_i > s_1 x_j$ and $s_2 y_i > s_2 y_j$ where $s_1, s_2 = +, -$. A point of S is *maximal* in (s_1, s_2) sense if and only if it is not dominated in (s_1, s_2) sense by any other point of S. Let $S_m^{(s_1,s_2)}$ be the *maximal subset* of S in (s_1, s_2) sense. Let

$$S_m = S_m^{(+,+)} \cup S_m^{(-,+)} \cup S_m^{(-,-)} \cup S_m^{(+,-)}.$$

It is not difficult to see that the convex hull points of S are contained in S_m.

Let S_e denote the extremal subset of S obtained by first finding the eight extreme points of S in the X, Y, $X+Y$ and $Y-X$ directions and deleting all points lying in the interior of the convex polygon resulting by connecting these extreme points in, say, counter-clockwise order. Again S_e contains the convex hull points of S.

We now show by citing counterexamples that S_m and S_e are not subsets of each other.

(a) S_m is not always contained in S_e

Let $p_1, p_2, \ldots, p_8$ be the eight extreme points of S (Fig. 1). Let us assume that all the remaining points of S lie inside the resulting convex polygon (Fig. 1). Therefore, $S_e=\{p_1, p_2, \ldots, p_8\}$. Let us consider a point $q \in S$ and $q \in S_e$. Clearly, q could be a maximal point if no point lies inside the quadrant shown in Fig. 1. Therefore, q may belong to S_m. Hence S_m is not always contained in S_e.

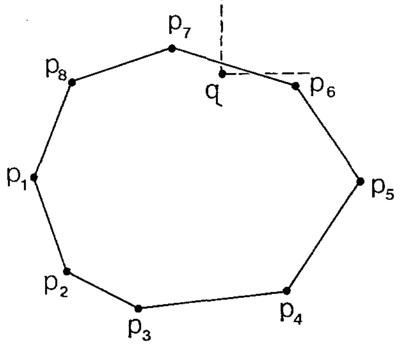

Fig. 1

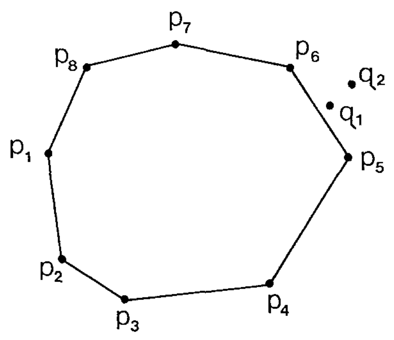

Fig. 2

(b) S_e is not always contained in S_m

Let $p_1, p_2, \ldots, p_8$ be the eight extreme points of S. Let q_1 and q_2 be two other points of S belonging to S_e (Fig. 2). Clearly, q_1 is dominated by the points of S in all the four senses. Therefore, $q_1 \notin S_m$. Hence S_m does not contain S_e.

2.2 *Algorithm* DIAM 1

We know that the pair of points that realize DIAM(S) belong to the set of vertices of the convex hull of S. Also it is observed that the convex hull vertices of S are contained in S_m. The algorithm DIAM 1, designed on the basis of these observations, is described as follows:

Step 1. Determine the maximal set S_m in S.

Step 2. Compute the distance between all pairs of points of S_m.

Step 3. Determine the maximum of the distances computed in step 2.

2.2.1 Analysis of DIAM 1

Each of the algorithms of Bentley et al. [3] and Kung et al. [18] to compute S_m has $O(n \log n)$ worst-case running time complexity. Step 1 is called the throw-away step. In this step all the non-maximal points are identified and these points are rejected from further consideration. Steps 2 and 3 of DIAM 1 are dependent on the cardinality of S_m. The size of S_m could be as high as $O(n)$ in the worst case. Therefore, $O(n^2)$ is the worst-case running time of DIAM 1.

In practice, it is rarely the case that all the points of S are maximal points. Let us assume that the data points are drawn independently at random from some underlying distribution f. Let N be the cardinality of S_m. The the complexity C of DIAM 1 is given by

$$C = C(S_m) + kN^2$$

where k is a positive constant and $C(S_m)$ denotes the complexity of determining the set S_m. The expected complexity of DIAM 1, denoted as $E\{C\}$, is given by

$$E\{C\} = E\{C(S_m)\} + kE\{N^2\}.$$

The algorithm in [3] to compute S_m has linear expected running time when $E\{N\}$ is $O(n^p)$, $p<1$. This is true whenever the density f can be written as the product of densities [4]. In other words, $E\{C(S_m)\}$ is $O(n)$ when the coordinates of the points of S are drawn from independent distributions. Under this assumption Devroye [11] has shown that $E\{N^2\}$ is also $O(n)$. Therefore, the expected running time of DIAM 1 is linear when the coordinates of the points of S are drawn from independent distributions.

2.3 *Algorithm* DIAM 2

We know that the vertices of the convex hull of S are contained in S_e. Therefore we can apply the exhaustive approach to the points of S_e, as in DIAM 1, to compute DIAM(S).

Step 1. Determine the extremal set S_e.

Step 2. Compute the distance between all pairs of points in S_e.

Step 3. Determine the maximum of the distances computed in Step 2.

2.3.1 Analysis of DIAM 2

The extremal set S_e can be obtained from S in a fairly simple manner in linear time. Like DIAM 1, Step 1 in DIAM 2 is called the throw-away step. The time required to implement steps 2 and 3 are dependent on the cardinality of S_e. If S_e contains N points, DIAM 2 takes $O(N^2)$ time to compute steps 2 and 3. Therefore, the worst case complexity of DIAM 2 is $O(n^2)$.

Concerning the expected complexity, Devroye and Toussaint [13] have shown that, when the points of S have a density f such that $0 < t \leq T < \infty$ where t and T are constants, for x in a rotated rectangle in 2-dimensional space and zero elsewhere, $E\{N^2\} \leq kn$, k a positive constant. Thus DIAM 2 runs in linear expected time when points of S are distributed uniformly in a rectangle. However, it is not linear for normally distributed points.

2.4 *Algorithm* DIAM 3

2.4.1 Theoretical basis

It is noticed from Shamos' algorithm [20], DIAM 1 and DIAM 2 that each algorithm removes quickly in their throw-away step a subset of points of S which lie in the "center of S". In the case of Shamos' algorithm, any point of S which is not a convex hull point of S can be considered to be lying in the center of the set. Similar arguments apply for nonmaximal points in DIAM 1 and non-extremal points in DIAM 2. In DIAM 3 we consider yet another approach to reduce the search space efficiently.

Let E be the set of (possibly eight) extreme points of S in the X, Y, $X+Y$ and $Y-X$ directions. The DIAM(S) is bounded below by DIAM(E). We enclose all the points of S by a rectangle with corners A_1, A_2, A_3, A_4, determined by the four extreme points in the X and Y direction. Clearly, DIAM(E) is greater than or equal to the length of the greatest side of the rectangle.

Corresponding to each corner A_i, there exists a region R_{i+2} such that for any point x in R_{i+2},

$d(A_i, x) \geq \text{DIAME}(E)$, $i=1, 2, 3, 4$ and $R_5 \equiv R_1$ and $R_6 \equiv R_2$. Let

$$R = \bigcup_{i=1}^{4} R_i,$$

Q = rectangle $A_1 A_2 A_3 A_4$, and

$T = Q - R$.

It is readily seen that for a pair of points p_i and p_j belonging to S with at least one of them, say p_i, lying in the interior of T, $d(p_i, p_j) < \text{DIAM}(E)$. Thus it follows that the pair of points p_i and p_j of S having $d(p_i, p_j) \geq \text{DIAM}(E)$ must belong to the set R. Let S_i be the set of points of S lying in the region R_i. Then, once DIAM(E) is known, we need to consider only S_i, $i=1, 2, 3, 4$ to compute DIAM(S). All the points of S lying in the interior of T are considered to be lying in the center of the set. Thus we have reduced the search space from S to $\bigcup_{i=1}^{4} S_i$. This search space can still be reduced further as shown in the following theorem.

Theorem 1. *The pair of points realizing* DIAM(S), *is such that one point belongs to* S_i *and the other belongs to* S_{i+2}, $i=1, 2$.

Proof. It is sufficient to show that all the pairs of points in $S_i \cup S_{i+1}$, $i=1, 2, 3, 4$; $S_5 \equiv S_1$ need not be considered. Due to symmetry, we need only consider the case $S_1 \cup S_2$.
Let $\bar{S}_{i,j} = S_i \cap S_j$. We can then partition S_1 and S_2 as follows:
$S_1 = \bar{S}_{1,4} \cup S'_1 \cup \bar{S}_{1,2}$, and
$S_2 = \bar{S}_{1,2} \cup S'_2 \cup \bar{S}_{2,3}$.
Therefore, (Fig. 3)
$S_1 \cup S_2 = \bar{S}_{1,4} \cup S'_1 \cup \bar{S}_{1,2} \cup S'_2 \cup \bar{S}_{2,3}$.

It is quite clear that some of $\bar{S}_{i,j}$ and S'_k could be empty.

(i) Since $\bar{S}_{1,4} = S_1 \cap S_4$ and $\bar{S}_{1,2} = S_1 \cap S_2$, it is not required here to compute $d(p, q)$ when $p \in \bar{S}_{1,4}$ and $q \in \bar{S}_{1,2}$ because this pair will be considered again in the sets S_2 and S_4. Similar arguments apply for the subsets $\bar{S}_{1,2}$ with $\bar{S}_{2,3}$ and $\bar{S}_{1,4}$ with $\bar{S}_{2,3}$.

(ii) We do not have to compute $d(p, q)$ when $p \in \bar{S}_{1,4}$ and $q \in S'_2$ since this pair will be considered again in the sets S_2 and S_4. Similarly, $\bar{S}_{2,3}$ and S'_1 need not be considered together.

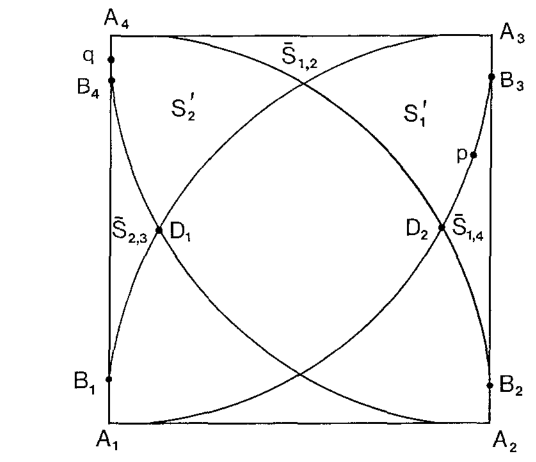

Fig. 3

(iii) It also follows from the construction that for any pair of points p and q, both belonging to $\bar{S}_{1,4} \cup S'_1$ or $S'_1 \cup \bar{S}_{1,2}$ or $\bar{S}_{1,2} \cup S'_2$ or $S'_2 \cup \bar{S}_{2,3}$, $d(p, q) < \text{DIAM}(E)$. Therefore, we can also neglect these cases.

(iv) The only remaining part to be proved is that for any $p \in S'_1$ and $q \in S'_2$, $d(p, q) < \text{DIAM}(E)$. In this case it is sufficient to show that for any point p in the line segment $A_3 B_3$ or on the arc $B_3 D_2$ and for any point q on the line segment $A_4 B_4$ or on the arc $B_4 D_1$, $d(p, q) < \text{DIAM}(E)$ (Fig. 3).

Without loss of generality, let the distance of p from the line $A_3 A_4$ be more than the distance of the point q from $A_3 A_4$. In this case, angle$(pqA_4) >$ angle(pA_4q) which implies that $d(p, q) < d(p, A_4)$ i.e. $d(p, q) < \text{DIAM}(E)$.

2.4.2 Steps of the algorithm DIAM3

The algorithm DIAM3 to compute the diameter of an arbitrary set S is described below:

Step 1. Compute the set E of extreme points of S in the X, Y, $X+Y$ and $Y-X$ directions.

Step 2. Enclose the set S by the rectangle defined by the extreme points in the X and Y directions.

Step 3. Compute DIAM(E) by the brute force method.

Step 4. Determine the points of S which belong to S_i, $I=1, 2, 3, 4$. (Throw-away step)

Step 5. Compute the distances between all pairs of points p and q where $p \in S_i$ and $q \in S_{i+2}$, $i=1, 2$.

Step 6. Select the maximum distance encountered in step 5 as the diameter of S.

2.4.3 Analysis of DIAM 3

Worst-case analysis. It is clear that steps 1 through 4 require $O(n)$ time in the worst case. The running time of Step 5 is dependent on the number of points in each of the sets S_i. If the points of S are such that they all lie on a circle, then all the points of S are involved in $\bigcup_{i=1}^{4} S_i$. In this case $O(n^2)$ operations are required to execute Step 5. Therefore, the worst-case complexity of DIAM 3 is $O(n^2)$.

Expected case analysis. It will be assumed that the underlying distribution of the points of S is uniform in a unit square. We now show that the expected running time of Step 5 in DIAM 3 is sublinear and therefore, the expected running time of DIAM 3 is $O(n)$ when the points are distributed uniformly in a unit square.

Let $A_1 A_2 A_3 A_4$ be the rectangle R which encloses the points of S generated uniformly in the unit square R': $A'_1 A'_2 A'_3 A'_4$. Clearly, R' encloses R totally. We already know that $\mathrm{DIAM}(E) \leq \mathrm{DIAM}(S)$. Let $R_i(R'_i)$ be the regions of $R(R')$ such that for any point q in $R_i(R'_i)$, $d(A_i, q) \geq \mathrm{DIAM}(E)$ $[d(A'_i, q) \geq \mathrm{DIAM}(E)]$. First we prove the following lemma.

Lemma 1. *When* $\mathrm{DIAM}(E) > \sqrt{\tfrac{5}{4}}$, $R_i \subseteq R'_i$, $i=1, 2, 3, 4$ *always.*

Proof. First R' is partitioned into four sub-squares (Fig. 4). When $\mathrm{DIAM}(E) > \sqrt{\tfrac{5}{4}}$, it is clear that the four corners of R must lie in the four sub-squares of R' (Fig. 4).

Let p be a point in R_1. It is now easy to show that when $\mathrm{DIAM}(E) > \sqrt{\tfrac{5}{4}}$, the point p must lie in the region shown in Fig. 4. In this case, $\mathrm{angle}(A'_1 A_1 p) \geq 90°$, thereby implying that $d(A'_1, p) \geq d(A_1, p) > \mathrm{DIAM}(E)$. Hence, if any point p is in R_1 it is also in R'_1. Therefore, $R_1 \subseteq R'_1$. Similar arguments apply for other regions also. Hence $R_i \subseteq R'_1$, $i=1, 2, 3, 4$ when $\mathrm{DIAM}(E) > \sqrt{\tfrac{5}{4}}$. □

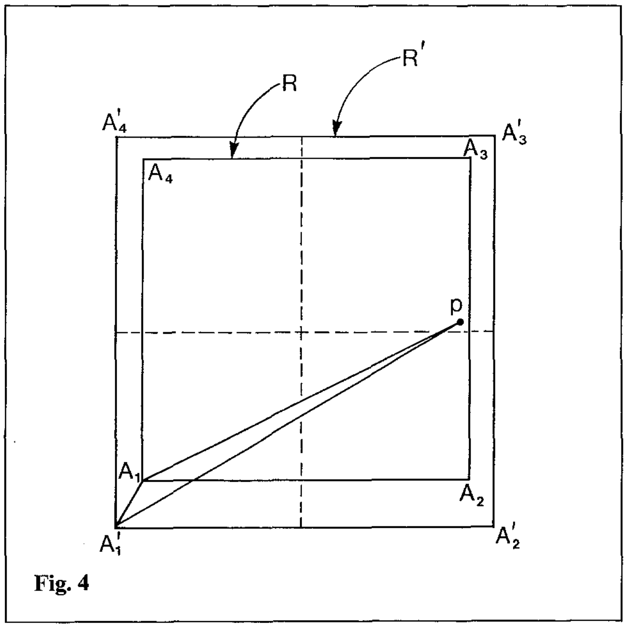

Fig. 4

Let us now consider the square R' [Fig. 5]. We construct the triangles $A'_i E'_i C'_i$, $i=1, 2, 3, 4$ with base at 45° and height $\delta < \tfrac{1}{2}[\sqrt{2} - \sqrt{\tfrac{5}{4}}]$. We now construct circles with centers at A'_i, $i=1, 2, 3, 4$ and radius $r = \sqrt{2} - 2\delta$. Clearly, $r > \sqrt{\tfrac{5}{4}}$. These circles determine points B'_{i+2} and D'_{i+2}, $i=1, 2, 3, 4$, (indices taken modulo 4) on the perimeter of the square R'. Let 2θ be the distance between B'_i and D'_i. From the triangle $A'_3 B'_1 A'_2$ (Fig. 5), we

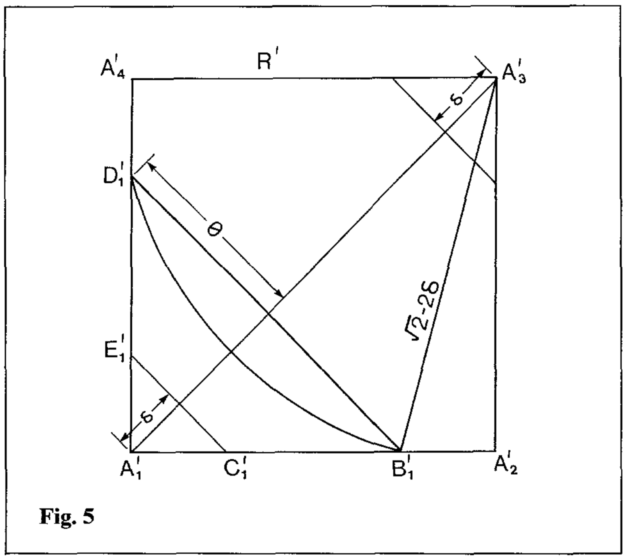

Fig. 5

see that

$$(\sqrt{2}-2\delta)^2=1+(1-\sqrt{2}\theta)^2$$

i.e. $\delta^2-\sqrt{2}\delta=\frac{\theta^2}{2}-\frac{\theta}{\sqrt{2}}$.

Furthermore, as δ approaches zero, θ also approaches zero (Fig. 5). Hence, for small δ, we have

$$\theta=(2+O(1))\delta. \qquad (1)$$

Let C and C' be the complexities of Step 5 when working with the rectangles R and R', respectively. When $\mathrm{DIAM}(E)>\sqrt{\frac{5}{4}}$, it is clear from Lemma 1 that $C<C'$.

Lemma 2. *If N is binomial (n, p), then* $E\{N^2\}\leq 2(1+n^2p^2)$.

Proof. We know that

$$E\{N^2\}=n^2p^2+np(1-p)\leq n^2p^2+np.$$

When $np\leq 1$, $E\{N^2\}\leq 2$ and when $np\geq 1$, $E\{N^2\}\leq 2n^2p^2$. Therefore,

$$E\{N^2\}\leq 2(1+n^2p^2). \quad \square \qquad (2)$$

Let N_0 be the total number of points of S in $\bigcup_{i=1}^{4}\Delta A'_i B'_i D'_i$. Let $Z=1$ if some of the triangles $A'_i C'_i E'_i$, $i=1,2,3,4$, are empty. Otherwise let $Z=0$. Clearly both N_0 and Z are random variables. If $Z=1$ then at least one small triangle $A'_i B'_i C'_i$ is empty and thus $\mathrm{DIAM}(E)$ could be smaller than or equal to $\sqrt{2}-2\delta$ and thus fewer points would be thrown away (in the worst case, maybe no points), resulting in an $O(n^2)$ complexity. If on the other hand, $Z=0$, then no small triangle is empty and therefore, $\mathrm{DIAM}(E)$ would be larger than $\sqrt{2}-2\delta$. Hence more points would be thrown away.

Theorem 2. $E\{C'\}\leq kn, \quad k>0$.

Proof. C', the complexity of executing Step 5 of DIAM 3, can be expressed as

$$C'\leq k_1 N_0^2+k_2 Z n^2 \qquad (3)$$

where k_1 and k_2 are nonnegative constants and let $\delta<\frac{1}{2}[\sqrt{2}-\sqrt{\frac{5}{4}}]$. The expected value of C', $E\{C'\}$, can be expressed as

$$E\{C'\}\leq k_1 E\{N_0^2\}+k_2 n^2 E\{Z\} \qquad (4)$$

Let p be the probability of a point falling in the triangle $A'_i B'_i D'_i$. Let q be the probability of a point falling in the triangle $A'_i C'_i E'_i$. Clearly, $p=\theta^2$ and $q=\delta^2$. Hence, using (1), we get $p=(4+O(1))\delta^2$. Thus (4) can be written as [using the Eq. (2)]

$$\begin{aligned}E\{C'\}&\leq 8k_1+8k_1 n^2p^2+4k_2 n^2(1-q)^n\\&=8k_1+8k_1 n^2(16+O(1))\delta^4+4k_2 n^2(1-\delta^2)^n\\&\leq 8k_1+8k_1 n^2(16+O(1))\delta^4+4k_2 n^2 e^{-n\delta^2}\\&\quad[\text{since } 1-x\leq e^{-x}].\end{aligned}$$

If we now take $\delta=\sqrt{\frac{2\log n}{n}}$, for large n, $\delta<\frac{1}{2}[\sqrt{2}-\sqrt{\frac{5}{4}}]$. Therefore, $E\{C'\}\leq k'\log^2 n$, $k'>0$. $\square$

3 Experimental results

A Monte Carlo simulation was carried out to compare the performances of DIAM 1, DIAM 2, DIAM 3 with the algorithms SNYDER [21] and SHAMOS [20].

We first discuss the implementation details of these algorithms which were coded in standard FORTRAN. Algorithms DIAM 2, DIAM 3 and SNYDER were implemented in the straight-forward manner. Implementing DIAM 1 and SHAMOS were nontrivial. The maximal set in DIAM 1 was computed using the standard divide-and-conquer approach [3]. The convex hull in SHAMOS was computed using the algorithm of Akl and Toussaint [1] as implemented in [7]. This algorithm has $O(n\log n)$ worst case running time [1] and $O(n)$ expected running time [13] under a reasonable probability model. The algorithm in [1] first computes S_e, the extremal set of S, and then the convex hull of S_e. Thus the throw-away step of DIAM 2 is identical to the throw-away step of the algorithm in [1]. The antipodal pairs of a convex polygon [19] were computed using the algorithm described in [5].

Space

The storage space requirement of each of DIAM 1, DIAM 2, DIAM 3, SNYDER and SHAMOS is shown in Table 1. Each algorithm requires $2n$ storage space for the input. The remaining storage space consists of work space. Thus DIAM 2, DIAM 3 and SNYDER may be the most desirable

Table 1. Storage space requirements of the algorithms in terms of n, the input size

DIAM1	DIAM2	DIAM3	SNYDER	SHAMOS
$5n$	$3n$	$3n$	$3n$	$5n$

algorithms to use when the storage space requirement is an important factor.

Time

All the algorithms were run on an IBM 3033 computer running MTS 5.0 using the FORTRAN G1 compiler. Several Monte Carlo simulations were carried out varying the number of points. The sample sizes considered for the number of data points were 100, 500, 1000, 3000, 5000, 7000, 10000, 12000 and 15000. Pseudo-random samples were generated for the following distributions:

(a) uniform in a unit square,

(b) normal with covariance matrix $\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}$, and

(c) normal with covariance matrix $\begin{pmatrix}0.5 & 0.375\\ 0.375 & 0.5\end{pmatrix}$.

Tables 2–4 show the results of the experiments. Each experiment was repeated 20 times. The running times are given in milliseconds.

It is seen from Table 2 that DIAM3 and SHAMOS are equaly efficient when the points are distributed uniformly in a unit square. However, DIAM3 has slightly larger standard deviation. Other algorithms, i.e. DIAM1, DIAM2 and SNYDER, have slower running times for all values of n. From Table 2 we can conclude the following:

(i) DIAM3 throws more points away than DIAM1, DIAM2 and SNYDER. This supports the theoretical claim that the expected running time of Step 5 of DIAM3 is $O(\log^2 n)$.

Table 2. Average running times (in milliseconds) to find the diameter of data sets generated uniformly in a unit square. Each experiment was repeated 20 times

Number of pts	DIAM1		DIAM2		DIAM3		SNYDER		SHAMOS	
	time	std. dev	time	std. dev	time	std. dev	time	std. dev	time	std. dev
100	6.3	0.10	3.3	0.12	3.0	0.00	3.6	0.11	4.1	0.08
500	28.6	0.16	15.1	0.51	13,5	0.15	16.9	0.28	15.3	0.17
1000	56.4	0.38	29.4	0.53	26.9	0.32	32.3	0.35	28.9	0.18
3000	166.9	1.24	85.3	1.73	82.5	0.77	98.9	0.91	82.0	0.66
5000	276.6	1.51	146.3	3.16	133.5	1.74	165.1	1.98	133.7	0.84
7000	392.7	1.58	210.7	4.49	187.8	2.29	239.3	4.10	188.0	1.36
10000	560.8	2.58	298.4	7.45	272.8	2.77	339.0	4.50	266.8	1.73
12000	672.8	3.78	360.9	11.18	321.7	4.04	403.4	5.93	318.5	1.67
15000	835.4	2.67	455.2	9.76	409.2	4.16	523.0	7.16	393.7	2.43

Table 3. Average running times (in milliseconds) to find the diameter of data sets generated normally with unit covariance matrix. Each experiment was repeated 20 times

Number of pts	DIAM1		DIAM2		DIAM3		SNYDER		SHAMOS	
	time	std. dev	time	std. dev	time	std. dev	time	std. dev	time	std. dev
100	6.2	0.11	2.5	0.11	2.9	0.11	3.1	0.07	3.3	0.10
500	28.2	0.19	10.8	0.15	14.9	0.34	15.1	0.24	12.4	0.13
1000	56.2	0.36	20.1	0.13	30.0	0.36	30.7	0.47	23.0	0.10
3000	167.5	1.16	58.9	0.35	95.7	3.11	94.7	1.32	65.9	0.27
5000	279.4	1.15	97.9	0.61	146.9	3.04	153.5	2.40	109.1	0.57
7000	391.5	1.98	135.1	0.37	222.3	8.07	217.3	2.99	150.6	0.24
10000	556.1	3.25	191.3	0.43	320.2	10.40	314.8	3.19	213.5	0.34
12000	673.0	2.57	230.1	0.88	396.1	17.20	372.2	5.74	256.0	0.55
15000	830.0	5.55	288.4	0.93	450.1	7.90	474.5	4.35	320.9	0.80

Table 4. Average running times (in milliseconds) to find the diameter of data sets generated normally where the variables are positively correlated. Each experiment was repeated 20 times

Number of pts	DIAM1		DIAM2		DIAM3		SNYDER		SHAMOS	
	time	std.dev	time	std.dev	time	std.dev	time	std.dev	time	std.dev
100	6.2	0.16	3.0	0.14	2.8	0.08	3.1	0.05	3.8	0.11
500	29.3	0.25	13.5	0.55	12.5	0.19	13.9	0.22	13.9	0.22
1000	55.4	0.41	25.8	1.26	24.3	0.31	28.3	0.29	26.0	0.54
3000	168.9	1.37	71.0	4.03	73.9	0.67	82.9	1.00	72.3	1.63
5000	280.4	1.93	122.8	4.86	117.5	1.75	136.9	1.74	122.9	2.38
7000	393.5	2.47	157.1	5.09	171.9	1.48	194.9	2.46	162.3	1.78
10000	558.9	4.53	228.4	4.56	247.0	1.54	282.5	4.39	236.3	2.80
12000	677.7	4.81	278.0	8.81	296.1	1.85	329.6	4.19	283.8	3.83
15000	837.1	6.88	370.2	19.95	365.2	3.78	411.7	5.24	357.9	5.32

(ii) DIAM2 is about 15% slower than SHAMOS. This suggests that the size of S_e is large because the difference of the running times of DIAM2 and SHAMOS to compute DIAM(S) is the same as the difference of the running times of the $O(n^2)$ brute force approach and $O(n \log n)$ approach of SHAMOS to compute DIAM(S_e) without any throw-away step.

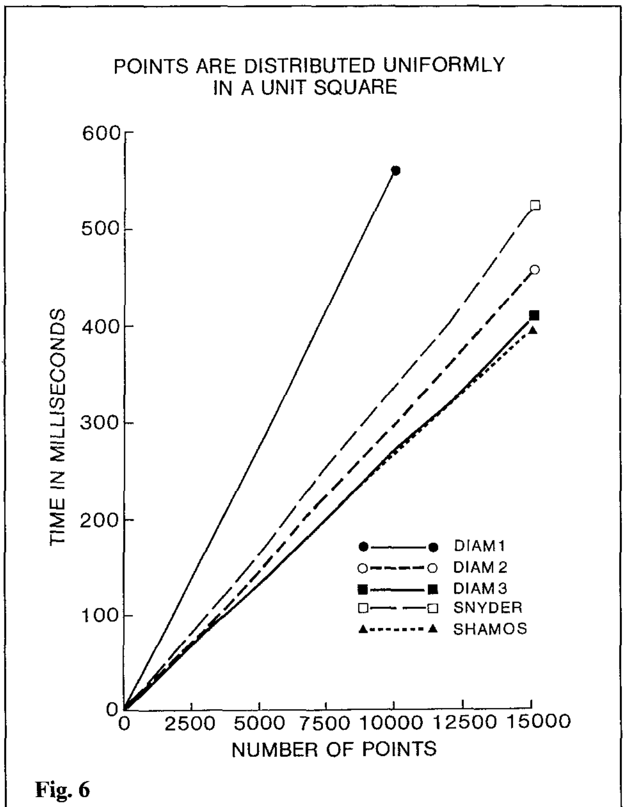

Fig. 6

All the five algorithms have average running times that are linear in n (Fig. 6). These results support the theoretical claim that the expected running time of each of DIAM1, DIAM2, DIAM3 and SHAMOS is linear in n when the points are distributed uniformly in a unit square.

Tables 3 and 4 give the running times in milliseconds of all the five algorithms when the points of S are distributed normally with covariance matrices $\begin{pmatrix}1.0 & 0.0\\ 0.0 & 1.0\end{pmatrix}$ and $\begin{pmatrix}0.5 & 0.375\\ 0.375 & 0.5\end{pmatrix}$ respectively. When the coordinates of the data points are uncorrelated, DIAM2 is the fastest. SHAMOS is about 10% slower than DIAM2 even though SHAMOS has lower worst-case running time complexity. DIAM3 is quite slow which indicates that not many points get thrown away during the throw-away step. DIAM1 and SNYDER are considerably slower. However, the picture changes significantly when the coordinates of the data points are correlated. In this case DIAM2, DIAM3 and SHAMOS have similar running times. This means that the points being rejected at the throw-away step are very sensitive to the direction in which the data points are oriented. If the general direction of data points is along any coordinate axis, DIAM2 will remove more points than DIAM3 in the throw-away step. However, the general direction of data points makes larger angles with both of the axes, DIAM3 removes considerably more points. The standard deviation of DIAM2 is quite large. This could be attributed to the fact that the size of S_e is very much dependent to the extreme values of the coordinates of the data points as mentioned in [13].

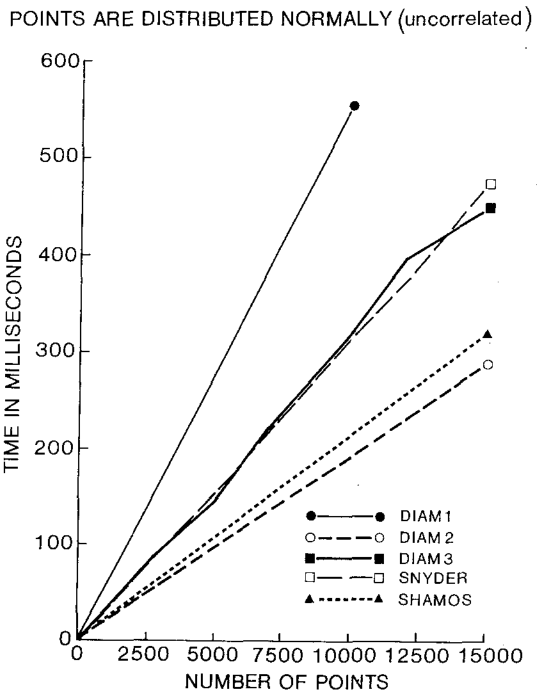

Fig. 7

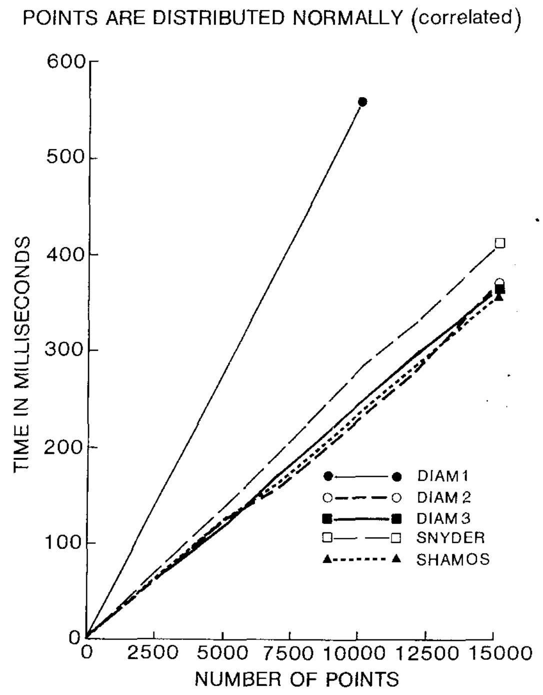

Fig. 8

Table 5. Average running times (in milliseconds) of the algorithm in [16, 17] to approximate the diameter of data sets generated uniformly in a unit square. Each experiment was repeated 20 times

Number of pts	Number of directions					
	4		8		16	
	time	std.dev	time	std.dev	time	std.dev
100	1.0	0.00	3.0	0.00	5.1	0.07
500	6.2	0.08	13.0	0.00	26.0	0.00
1000	13.0	0.00	25.8	0.09	51.0	0.00
3000	38.1	0.07	76.6	0.11	152.5	0.11
5000	64.0	0.00	128.5	0.11	257.4	0.75
7000	90.0	0.00	179.8	0.10	369.2	0.12
10000	128.1	0.05	256.5	0.11	512.6	0.16
12000	153.7	0.10	308.0	0.07	615.1	0.21
15000	192.0	0.00	384.2	0.09	769.1	0.23

Figures 7 and 8 show the average running times of theses algorithms when the data points are normally distributed. Each of these algorithms shows a running time linear in n.

We also implemented the algorithm in [16, 17] and average running times in milliseconds for $m=4$, 8 and 16 are shown in Table 5. We see from this table that even the *approximate* algorithm, for $m>8$, is slower than DIAM 3 or SHAMOS when the points are uniformly distributed. This is also true for data points distributed normally. Thus if one is interested in computing the approximate diameter with maximum percentage error within $100\left(1-\cos\left(\frac{90}{9}\right)\right)\% = 1.5\%$ of the true diameter, it is better, on an average, to use either DIAM 2, DIAM 3 or SHAMOS to compute the exact diameter.

4 Conclusion

We have proposed three algorithms DIAM 1, DIAM 2 and DIAM 3 to compute the diameter of a planar set of points. Each of these algorithms uses the "throw-away" principle [1] to reduce the search space considerably. Though all three algorithms have $O(n^2)$ running time in the worst case, analysis of the expected complexity shows that under a reasonable probability model, all of these algorithms have linear expected running time. DIAM 1, DIAM 2 and DIAM 3 are experimentally compared with SNYDER [21] and SHAMOS [20]. Experimental results indicate that DIAM 3 and SHAMOS perform very well when the points are distributed uniformly in a unit square. When the points are distributed normally, DIAM 2 works best for the set of data points whose coordinates are uncorrelated. However, when the coordinates of the data points are correlated, DIAM 3 has the similar average running time as that of SHAMOS. DIAM 2 and DIAM 3 require $3n$ storage space whereas SHAMOS requires $5n$ storage space. DIAM 2 and DIAM 3 are very easy to implement whereas implementing SHAMOS is nontrivial. It is also seen that DIAM 2 and DIAM 3 compare very well with the approximate algorithm of Firschein et al. [16] and Fischler [17] when one is interested in estimating the true diameter with maximum percentage error within 1.5% of the true diameter. In fact we have exhibited situations where the exact algorithm runs faster than an approximate algorithm. This suggests that care is needed when designing *approximate* algorithms to increase the speed of their execution.

Acknowledgement. The authors are indebted to Luc Devroye for the penetrating expected-time analysis of DIAM 3 [12].

References

1. Akl S, Toussaint GT (1978) A fast convex hull algorithm. Inf Proc Lett 7:219–222
2. Avis D, Toussaint GT, Bhattacharya BK (1982) On the multimodality of distances in convex polygons. Comput Math Appl 8:153–156
3. Bentley JL, Kung HT, Schkolnick M, Tompson CD (1978) On the average number of maxima in a set of vectors and applications. J ACM, 25:536–543
4. Bentley JL, Shamos MI (1978) Divide and conquer for linear expected time. Inf Proc Lett 7:87–91
5. Bhattacharya BK (1982) On the determination of all the antipodal pairs of a convex polygon. Internal manuscript
6. Bhattacharya BK, Toussaint GT (1982) A counterexample to a diameter algorithm for a convex polygon. IEEE Trans Pattern Anal Mach Intell PAMI 4:306–309
7. Bhattacharya BK, Toussaint GT (1983) Time-and-storage efficient implementation of an optimal planar convex hull algorithm. Image and Vision Computing 1:140–144
8. Bhattacharya BK, Toussaint GT (1985) On geometric algorithms that use the furthest-point Voronoi diagram. In: Toussaint GT (ed) Computational Geometry. North-Holland, pp 43–61
9. Brown KQ (1979) Geometric transformations for fast geometric algorithms. PhD Thesis, Carnegie-Mellon University
10. Cacoullos T, De Cicco H (1967) On the distribution of bivariate range. Technometrics 9:476–480
11. Devroye LP (1980) A note on finding convex hulls via maximal vectors. Inf Proc Lett 11:53–56
12. Devroye LP (1982) Private Communication
13. Devroye LP, Toussaint GT (1981) A note on linear expected time algorithms for finding convex hulls. Computing 26:361–366
14. Dobkin D, Snyder L (1979) On general method for maximizing and minimizing among certain geometric models. 20th Ann Symp Foundat Comput Sci, pp 9–17
15. Duda RO, Hart PE (1973) Pattern classification and scene analysis. John Wiley, New York
16. Firschein O, Eppler W, Fischler MA (1979) A fast defect measurement algorithm and its array processor mechanization. Proc IEEE Comput Soc Conf Pattern Recognition and Image Processing, Chicago, pp 109–113
17. Fischler MA (1980) Fast algorithms for maximal distance problems with applications to image analysis. Pattern Recognition 12:35–40
18. Kung HT, Luccio F, Preparata FP (1975) On finding the maxima of a set of vectors J ACM 22:469–476
19. Shamos MI (1978) Computational Geometry. PhD Thesis, Yale University
20. Shamos MI, Hoey D (1975) Closest point problems. Proc 16th Ann IEEE Symp Foundat Comput Sci, pp 151–162
21. Snyder WE, Allan Tang D (1980) Finding the extrema of a region. IEEE Trans Pattern Anal Mach Intell PAMI-2:266–269
22. Toussaint GT (1985) Computational geometry and morphology. Proc First Int Symp Sci Form, Tsukuba, Japan (November 1985)
23. Toussaint GT (1985) A historical note on convex hull finding algorithms. Pattern Recognition Lett 3:21–28