A Practical Approach to Approximating the Diameter of Point-set in Low Dimensions

Kumar Gaurav Bijay* Antoine Vigneron†

Abstract

The problem of computing the diameter of point-sets presents itself in a variety of fields like databases, data-mining and vision. A naïve algorithm takes time O(n^2) which is impractical for the large point-sets encountered in these fields, and hence there is a need for faster algorithms, albeit approximate. We present new ideas to efficiently approximate the diameter of a point-set in low dimensions. The new algorithm has a worst-case running time of $O(n + \sqrt{n_\epsilon}\frac{1}{\epsilon^{d/2}})$ - where $n_\epsilon \leq \frac{1}{\epsilon^{d-1}}$ and is faster for soft inputs where the number of potential diametrical pairs is small.

1 Definition

Given a set S of n-points in d-dimensional space, the diameter of the set is the maximum distance between any two points in the set, that is,

$$diameter(S) = \max_{x,y \in S} \|x - y\|$$

2 Approximate Diameters

Let Δ be the actual diameter of the given point-set S.An approximate diameter Δ_ϵ is called an ϵ-*approximation* of Δ if

$$\Delta_\epsilon \leq \Delta \leq (1+\epsilon)\Delta_\epsilon$$

2.1 Constant-factor Approximates

Getting a constant-factor approximation in linear time is an easy problem. For example, the following algorithm returns a 2-factor approximation:

1. pick a point x $\in$ S-the given point-set.
2. Find the point y $\in$ S that maximizes $\|x - y\|$ by brute-force.
3. Δ_2=$\|x - y\|$

2.2 ϵ - Approximates

2.2.1 Basic Idea

Though there have been other attempts, the idea[2] that has been the most successful comprises of the following two steps:

- **Step I : Snapping points**
  Consider an ϵ-grid over the d-dimensional space. Round each point to its nearest grid-point. This step takes O(n) time. As the grid has ϵ as unit-size, the error in the diameter will be O(ϵ). Also, the maximum number of points after this step can be O($\frac{1}{\epsilon^d}$).
- **Step II : Projection**
  Consider x and y as the pair of points in the given point-set at diametrical distance. Let direction l be such that angle($x - y$,l) $\leq \sqrt{\epsilon}$. Let x' and y' be the orthogonal projections of x and y on l. Then,

$$\begin{aligned} \cos(\sqrt{\epsilon}) &\geq 1 - \epsilon/2 \;\; - \; Taylor\; Expansion \\ \Rightarrow \|x' - y'\| &\leq \|x - y\| \leq \|x' - y'\| \frac{1}{1 - \epsilon/2} \\ \Rightarrow \|x - y\| &\leq \|x' - y'\|(1+\epsilon) \end{aligned}$$

Thus,$\|x' - y'\|$ is an O(ϵ) - approximation of $\|x - y\|$. The problem is to get this direction l. So, choose many direction-vectors such that for any line L there exists a direction-vector v with angle(L,v) $\leq \sqrt{\epsilon}$. Such a set of direction-vectors with cardinality O($\frac{1}{\epsilon^{(d-1)/2}}$) can be found - for example, by constructing a grid on a unit-sphere. As the number of points have been reduced by step I to O($\frac{1}{\epsilon^d}$), the overall time required is O(n + $\frac{1}{\epsilon^d}\frac{1}{\epsilon^{(d-1)/2}}$) i.e. O(n + $\frac{1}{\epsilon^{(3d-1)/2}}$). Note that had brute-force been applied after step I, the time required would have been O(n + $\frac{1}{\epsilon^{2d}}$). For analysis sake, we shall assume that $\Delta \leq 1$. Note that this can be achieved by computing a minimum axis-parallel bounding box in $O(n)$ time and scaling it.

3 Improvements

There have been improvements to the running-time of the above technique; attempts have also been made to use a different approach than projection. Pioneering work has been done by Chan[3][2], Har-Peled[6], Agrawal[1], Malandain and Boissonnat[7] and others. The essential first step in most algorithms is the same. It is the second step where the approaches differ. A recent development in this direction is a theorem due to Chan[3] which computes the diameter of the grid-points in time O(n+$\frac{1}{\epsilon^{d-3/2}}$). We present a modified version of Chan's algorithm that has a worst-case time bound of O($\sqrt{n_\epsilon}(\frac{1}{\epsilon^{d/2}})$) where n_ϵ is the number points obtained after projection to the ϵ-grid. We claim that the performance will be still better for *soft* inputs, where the number of potential diametrical pairs is small.

*Department of Computer Science and Engineering, Indian Institute of Technology, Bombay, gauravk@cse.iitb.ac.in

†School of Computing, National University Of Singapore, antoine@comp.nus.edu.sg

3.1 Chan's Algorithm

We will assume that O(n) time has been taken to snap the points to an ϵ-grid from now on. We refer the reader to Chan's algorithm [3]. The projection vectors can be taken to be coming from a $\sqrt{\epsilon}$-grid on a hypercube. The algorithm breaks the problem into maximizing the projection on each one of the 2d (d-1) dimensional faces of the hypercube. It uses a subroutine that gets the maximizing points with fixed first co-ordinates and then finds the maximum among these. Its running time is O($n + E^{d-2}F$) where E=O($\frac{1}{\epsilon}$) and F=O($\frac{1}{\sqrt{\epsilon}}$). This algorithm is optimal only if n=$\Omega(\frac{1}{\epsilon^{d-3/2}})$.
Note: To get O(1) largest distances between the pairs of points instead of just the diameter, we can keep O(1) points at each stage of the algorithm and the running time will be unaffected.

3.1.1 Modified Chan's Algorithm

A weakness of the above method is that no matter how the points are distributed in space, the time taken is the same. We now present the *Modified Chan's Algorithm*.

Claim: When $n = O(E^k)$, Chan's algorithm can be modified to run in time O(n+$E^k F^{d-k-1}$)=O($E^k F^{d-k-1}$).

Proof : We prove the above claim by modifying the algorithm to use the following *New Subroutine* to compute for each $x \in [F]^{d-1}$, a point $p \in P \subseteq [E]^{d-1} \times \mathbb{R}$ that maximizes $p_1x_1 + \cdots + p_{d-1}x_{d-1} + p_d$.

New Subroutine

1: **for all** $i_1, \ldots, i_k \in [E]$ **do**
2: **for all** $x_{k+1}, \ldots, x_{d-1} \in [F]$ **do**
3: $r[i_1, \ldots, i_k, x_{k+1}, \ldots, x_{d-1}]$ = a point p ∈ P with $p_1 = i_1, \ldots, p_k = i_k$ that maximizes $p_{k+1}x_{k+1} + \cdots + p_{d-1}x_{d-1} + p_d$
4: **end for**
5: **end for**
6: **for all** $x_{k+1}, \ldots, x_{d-1} \in [F]$ **do**
7: **for all** $x_1, \ldots, x_k \in [F]$ **do**
8: $q[x_1, x_2, \ldots, x_{d-1}]$ = a point p ∈ $\{r[i_1, \ldots, i_k, x_{k+1}, \ldots, x_{d-1}] | i_1, \ldots, i_k \in [E]\}$ that maximizes $p_1x_1 + p_2x_2 + \cdots + p_{d-1}x_{d-1} + p_d$
9: **end for**
10: **end for**

This idea of projection to a carefully chosen sub-space is important. As in Chan's original algorithm, we have in lines 3 and 8 problems similar to the one we are trying to solve. However, Chan's idea was to use the same algorithm recursively. Instead, we propose to solve the reduced problem in line 3 by brute-force while the problem in line 8 shall be solved using original Chan's algorithm.

Timing Analysis:
Let $T_{orig}(d, n)$ be the time needed by the original Chan's algorithm when running on a sample of size n in d-dimensions; likewise let $T_{mod}(d, n)$,$T_{brute}(d, n)$ be the time required by modified Chan's algorithm and brute-force algorithm respectively.

The time needed for modified Chan's algorithm can be expressed as:

$$\begin{aligned}
T_{mod}(d,n) &= \sum_{i=1}^{E^k} T_{brute}(d-k, n_i) + \\
&\quad F^{d-k-1} T_{orig}(k+1, E^k) + \\
&\quad O(F^{d-k-1}E^k) \\
\Rightarrow T_{mod}(d,n) &= \sum_{i=1}^{E^k} n_i F^{d-k-1} + F^{d-k-1}E^k \\
&\quad \left(as\ T_{orig}(k+1, E^k) = O(E^k)\right) \\
&= F^{d-k-1} \sum_{i=1}^{E^k} n_i + F^{d-k-1}E^k \\
&= F^{d-k-1}n + F^{d-k-1}E^k \\
&= F^{d-k-1}E^k \quad - as\ n = O(E^k) \\
\Rightarrow T_{mod}(d,n) &= F^{d-k-1}E^k
\end{aligned}$$

For our problem, $n = O(E^k)$ and $F = \sqrt{E} = \frac{1}{\sqrt{\epsilon}}$. Therefore, $T_{mod}(d, n) = O(\sqrt{n}\frac{1}{\epsilon^{d/2}})$.

4 The new idea

The new algorithm is fairly intuitive. We do not attempt to change step I of the Basic Idea. It is the second step which looks inefficient. The basic question we ask is - Can we find quickly, the possible candidates for diametrical directions ? If there is a way out, we need not project on all the directions. This notion shall now be discussed.

- **Step A:**
  Snap the points to a $\sqrt{\epsilon}$-grid and run the diameter algorithm to get potential diametrical-pairs.
  We now prove a simple theorem.

  Theorem 1 *Let Δ be the diameter in ϵ-grid (between points p and q). Let the point-set be snapped to a $\sqrt{\epsilon}$-grid, p and q going to p' and q' respectively . Let Δ' be the diameter in $\sqrt{\epsilon}$-grid (between points x' and y'). Then, $\|p' - q'\| \geq \Delta' - 2\sqrt{d\epsilon}$.*

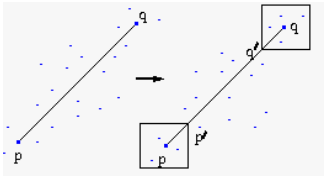

Figure 1: Snapping to $\sqrt{\epsilon}$-grid

Proof. Observe that when a and b are snapped to a' and b' on the $\sqrt{\epsilon}$-grid, $\|a' - b'\| - \sqrt{d\epsilon} \leq \|a - b\| \leq \|a' -$

$b'\| + \sqrt{d\epsilon}$. Consider figure 1. Here,

$$\begin{aligned} \|p' - q'\| &\geq \Delta - \sqrt{d\epsilon} \\ Also,\ \Delta &\geq \Delta' - \sqrt{d\epsilon} \\ \Rightarrow \|p' - q'\| &\geq \Delta' - 2\sqrt{d\epsilon} \end{aligned}$$

□

We have seen that O(1) diameters can be maintained using Chan's and similarly modified Chan's algorithm. So, if we snap the points to a $\sqrt{\epsilon}$-grid and run modified Chan's algorithm, the time required would be $\approx$ O($\sqrt{n_{proj}}\frac{1}{\epsilon^{d/4}}$) *i.e.* $O(\frac{1}{\epsilon^{d/2}})$ as $n_{proj} = O(\frac{1}{\epsilon^{d/2}})$. The above theorem then ensures that the actual diameter was reduced to one of these obtained diametrical-pairs.

- **Step B:**
  Get the points from the ϵ-grid that were snapped to the obtained diametrical-pairs from step A (henceforth this region is called as a cell). Project these points on the obtained diametrical pairs and return the maximum.

  Step B requires a good thought as we now face a dilemma here. If we have case(A) like situation, as shown in figure 2, it is good as we can simply project on the direction of the diameter however if we have case(B) like situation, naïve projection would be too costly.

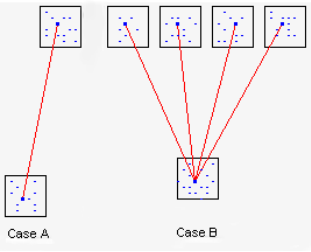

Figure 2: Step B dilemma!

4.1 A Simple Strategy

To solve the dilemma in step B, we consider the following lemma.

Lemma 2 *If 2 diametrical pairs through some point A are at an angle $\leq \epsilon^{1/4}$, then projecting the points around A from the ϵ-grid to just one of them is sufficient. (This effectively implies that we only need to consider diameters whose angular-separation is $\Omega(\epsilon^{1/4})$.)*

Proof. Consider the figure 3 shown. AB and AC are two diameters having angular-separation as $\epsilon^{1/4}$. Let X be a point at distance α along AB. We claim that the projection along AC suffices, projection on AB is not required. This is because the error incurred, e is:

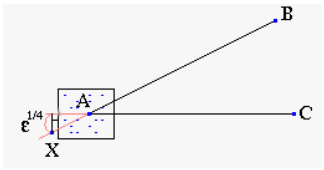

Figure 3: Determining diameters to be considered

$$\begin{aligned} e &\leq \alpha - \alpha\cos(\epsilon^{1/4}) \\ &\leq \alpha - \alpha(1 - \frac{\sqrt{\epsilon}}{2}) \\ &\leq \frac{\alpha}{2}\sqrt{\epsilon} \\ &\leq \frac{\omega\sqrt{\epsilon}}{2}\sqrt{\epsilon} \quad - as\ \alpha = O(\sqrt{(\epsilon)}) \\ &\leq O(\epsilon) \quad - as\ \omega = O(1) \end{aligned}$$

□

We now adopt a simple greedy strategy 1 i.e. for each cell, we try to use the best-available method - modified Chan's or simple projection - whichever is cheaper.

Algorithm 1 Subroutine for projection of points

```
for all i = 1 ... E^{d/2} do
    if E^{d/2} ≥ #(d_i)√n_i then
        simply project points in cell i (brute force)
    else
        project points in cell i by modified Chan's algorithm
    end if
end for
```

Here, $E = 1/\sqrt{\epsilon}$ and $\#(d_i)$ is the number of diametrical pairs corresponding to cell i.

The following theorem [6] constraints the number of near-diametrical pairs.

Theorem 3 *Let diameter be scaled to 1 and ab and yz be two diametrical pairs returned after step A of the new algorithm. Then both of the following statements cannot simultaneously hold true:*

- *All of* $\|a - y\|, \|a - z\|, \|b - y\|, \|b - z\| \geq 3\epsilon^{1/4}$
- *Angle(a-b,y-z)* $\leq \epsilon^{1/4}/3$

where $\epsilon < 4/9$.

5 Bad Cases - Nearly Spherical Distribution

Perfectly Spherical: Lets first look at a perfectly spherical distribution.

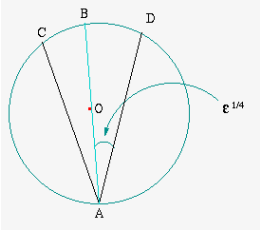

Figure 4: Perfectly spherical case - GOOD

In this case, for each diameter AB,

$$\begin{aligned} \theta &\approx \arccos(\frac{1-\sqrt{\epsilon}}{1}) \\ \Rightarrow \theta &\approx \epsilon^{1/4} \end{aligned}$$

This however by lemma 2 is a good-case for us. For each ball, we need to consider O(1) diameters only and hence we require O(n+$\frac{1}{\epsilon^{d/2}}$) time.

Nearly Spherical: We now take up the case of nearly spherical distributions. All points in region AB are potential diam-

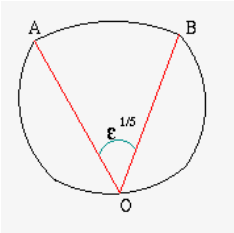

Figure 5: Nearly spherical case - BAD

eters here and angle AOB is $\Omega(\epsilon^{1/4})$, say $\epsilon^{1/5}$. In this case, the running-time tends towards its upper bound. It must be noted that the number of potential diametrical pairs is very large here.

6 Conclusion

We have thus presented a practical algorithm based on Chan's algorithm [3] that gives good performance for *soft* inputs. As the distribution assumes a more spherical shape, that is, more diametrical pairs, the running time suffers but has an upper bound of O(n+$\sqrt{n_\epsilon}\frac{1}{\epsilon^{d/2}}$).

7 Acknowledgments

The authors would like to thank Professor Abhiram Ranade of Indian Institute of Technology, Bombay for fruitful discussions during the course of this work.

References

[1] P. K. Agarwal, J. Matousek, and S. Suri. Farthest neighbors, maximum spanning trees and related problems in higher dimensions. In *Computational Geometry: Theory and Applications*, volume 1(4), pages 189–201, 1992.

[2] T. M. Chan. Approximating the diameter, width, smallest enclosing cylinder, and minimum-width annulus. *Int. J. Comput. Geometry Appl.*, 12(1-2):67–85, 2002.

[3] T. M. Chan. Faster core-set constructions and data stream algorithms in fixed dimensions. In *Annual Symposium on Computational Geometry*, pages 152–159, 2004.

[4] T. H. Cormen, C. Stein, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. The MIT Press, 2001.

[5] K. Fischer and B. Gartner. The smallest enclosing ball of balls: combinatorial structure and algorithms. In *Annual Symposium on Computational Geometry*, pages 292–301, 2003.

[6] S. Har-Peled. A practical approach for computing the diameter of a point set. In *Annual Symposium on Computational Geometry*, pages 177–186, 2001.

[7] G. Malandain and J.-D. Boissonnat. Computing the diameter of a point set. In *DGCI*, volume 2301, pages 197–208, 2002.

[8] D. Mount. 754 lecture notes, 2002. `http://www.cs.umd.edu/~mount/754/Lects/754lects.pdf`.