The Smallest Enclosing Circle Problem

Adarsh J and Shubham Sahai

Indian Institute of Technology, Kanpur

{*adarshaj,ssahai*} *@cse.iitk.ac.in*

November 18, 2013

Overview

Our mall is centerally located !!

In real world scenario we have heard the above line with respect to several facilities.

Our mall is centerally located !!

In real world scenario we have heard the above line with respect to several facilities.
For example,

- The mall.
- Schools, colleges and other educational institutions
- Hospitals.
- Housing Societies.

Our mall is centerally located !!

In real world scenario we have heard the above line with respect to several facilities.
For example,

- The mall.
- Schools, colleges and other educational institutions
- Hospitals.
- Housing Societies.

So, the "centrally located" terminology is apperently known to all of us.

Also, it gives us an intuitive idea that in some sense it is good for us in terms of "reduced distance" !!

Lets look at it more closely then !!

Motivation

Let us consider a simple scenario in which we want to build a charitable hospital in some rural area.

- The first task for the above project would be site selection.
- Obvious aim would be to minimize the distance between the hospital and residance of each person in the locality

One solution could be :

- Consider each residance of the locality as a point.
- Find the smallest circle that encloses all the points.
- If the hospital is located at the center of the circle, it minimizes the distance between the hospital and the farthest residance.
- Hence, solves the above problem.

Fromal Definition : Smallest Enclosing Circle

So we had established that, the term centrally located is analogous to defining a smallest enclosing circle in some real world setting, correspinding to the problem.

Smallest Enclosing Circle

Given a set of S of $2D$ points, find the circle C with the smallest radius such that all the points in S are contained in either C or its circumference.

Lets Circle around the Circle !!

Lets review some basic properties of circle and smallest enclosing circle before moving forward.

- **Center** is the point which is at the minimum distance from all the points on or within the circle (considered together).
- Given any three points, we can uniquely define a circle, with these points on circumference.
- Given any set of points P, the smallest enclosing disk (containing all the points) in unique.
- Given $N \geq 2$ points in the plane, the smallest circle that encloses them contains at least two of the points on its circumference.

Solution : Smallest Enclosing circle problem

A Naive Approach

A naive algorithm solves the problem in time $O(n^4)$ by testing the circles determined by all pairs and triples of points.

Running Time :

- The are $O(n^3)$ such circles
- Testing all points corresponding to each circle takes $O(n)$ time.
- The overall algorithm runs in $O(n^4)$ time.

Can we do better?

Fortunately, the answer is yes !!

- There are algorithm that make use of the convex hull, and has the running time ranging from $O(nh)$ to $O(h^3 n)$.
- There also exist several algorithms with running time ranging from $O(n^4)$ to $O(n log\ n)$.
- The best known algorithms are the Megiddo's Algorithm ($O(n)$ running time) and Welzl's Algorithm (expected $O(n)$ running time).

Further we will have a closer look at the Welzl's Randomized Algorithm for solving the problem.

Welzl's Algorithm

- Proposed by Emo Welzl in 1991.
- Solves the smallest enclosing disc problem, in expected $O(n)$ time.

Insight :

- The algorithm uses the power of randomized incremental construction.
- It also exploits the fact, that if the point is not present in the disc constructed so far, then it will be present on the boundary of the new disc.
- Basically, these are the known facts, and a generalized version of these facts are used in the algorithm.

Let's have a closer look at the algorithm !!

Basic Idea : Welzl's Algorithm

- Let md(P) denote the minimum enclosing disc enclosing the set of points "P"
- $md(P)$ for a given set P of n points, is calculated in an Randomized Incremental fashion.
- Let P be the set of n points and $D = md\{p_1, p_2, ...p_i\}$, for $1 \leq i \leq n$ points seen so far.
- Now, if $p_{i+1} \in D$, the we need not do anything, and now, $D = md\{p_1, p_2, ...p_i, p_{i+1}\}$, and we proceed to next point.
- Else, we use the fact that p_{i+1} will lie on the boundary of $D' = md\{p_1, p_2, ...p_i, p_{i+1}\}$.
- We can compute it by calling b_minidisk (A, p) , which calculates the smallest disk enclosing $A = \{p_1, p_2, ...p_i\}$ with $p = p_{i+1}$ on its boundary.

Welzl's Algorithm

Algorithm 1 : minidisk (P)

1: **if** $P = \phi$ **then**
2: $D \leftarrow \phi$
3: **else**
4: choose $p \in P$
5: $D \leftarrow minidisk(P - \{p\})$
6: **if** $p \notin D$ **then**
7: $D \leftarrow b\_minidisk(P - \{p\}, p)$
8: **end if**
9: **end if**
10: return D

Welzl's Algorithm

Algorithm 2 : b_minidisk (P, R)

1: **if** $P = \phi[or|R| = 3]$ **then**
2: $D = b\_minidisk(\phi, R)$
3: **else**
4: choose random $p \in P$
5: $D \leftarrow b\_minidisk(P - \{p\}, R)$
6: **if** [D defined and]$p \notin D$ **then**
7: $D \leftarrow b\_minidisk(P - \{p\}, R \cup \{p\})$
8: **end if**
9: **end if**
10: return D

Experimental Results

n	Running Time (μsec)
10	14
10^2	77
10^3	619
10^4	6156
10^5	83488
10^6	1051354
10^7	12889873

Table: Running time vs n

Distribution of running time

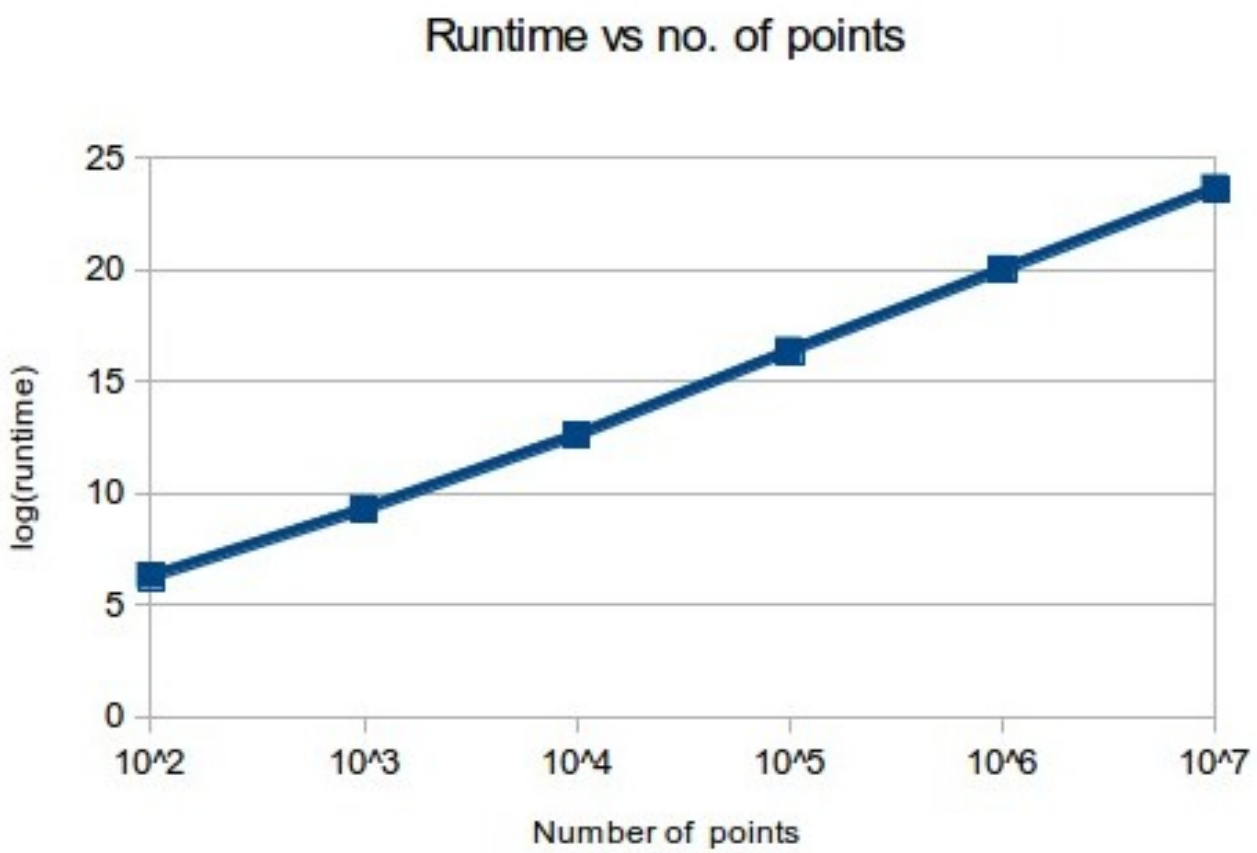
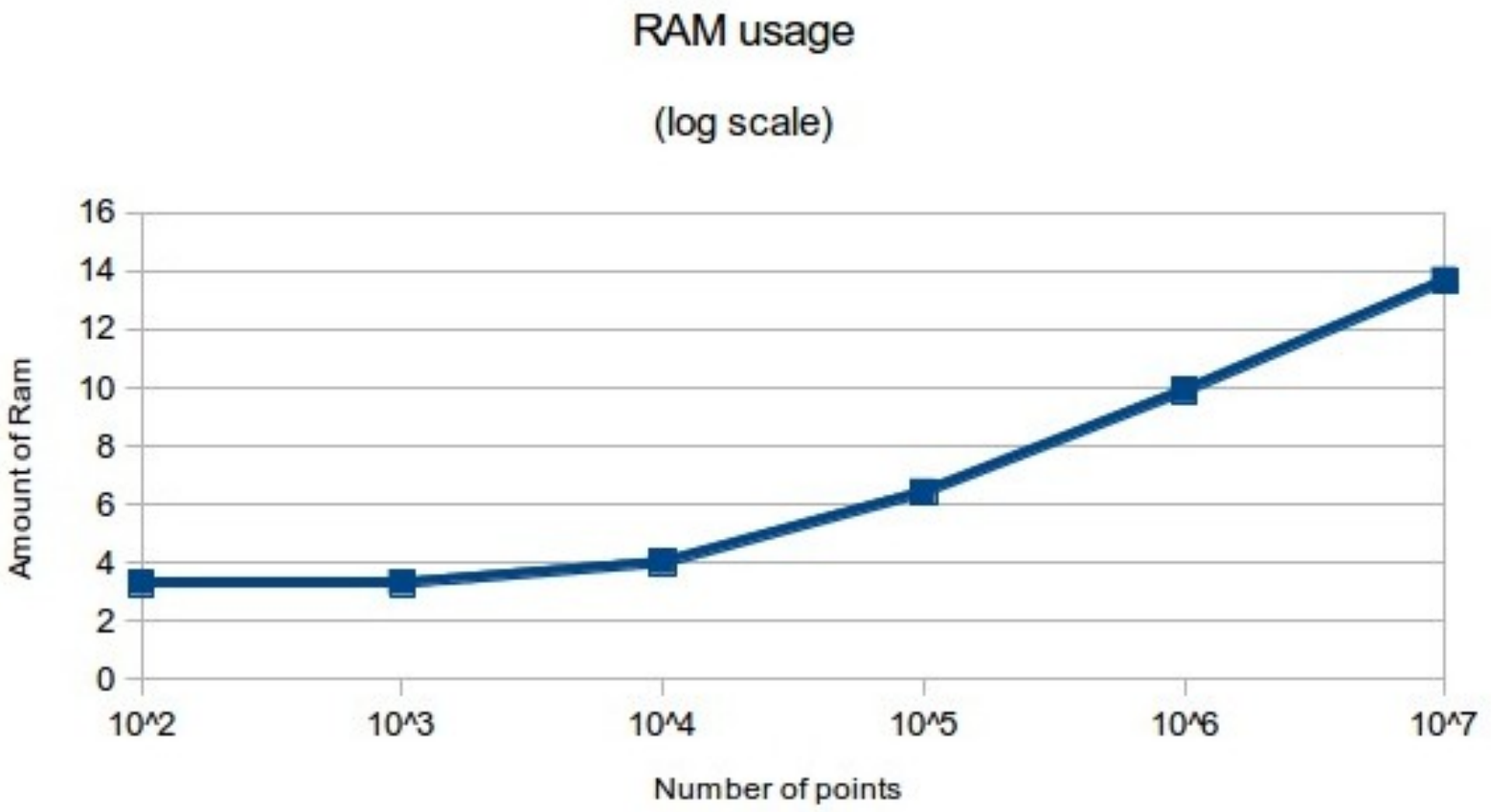


Table: RAM Utilization vs n

n	**RAM Utililization (mb)**
10	0.9911
10^2	0.9916
10^3	0.9940
10^4	1.63
10^5	8.59
10^6	95.78
10^7	1308

RAM Usage


RAM usage
(log scale)
Amount of Ram
16
14
12
10
8
6
4
2
0
10^2
10^3
10^4
10^5
10^6
10^7
Number of points

No. of cases	10	10^2	10^3	10^4	10^5	10^6
$<= 50\% \times avg$	85	61	28	13	4	4
$<= 75\% \times avg$	214	291	179	171	180	174
$<= 90\% \times avg$	159	229	197	198	200	194
$<= 95\% \times avg$	84	55	71	84	83	83
$<= avg$	67	44	70	73	69	84
$<= 105\% \times avg$	0	33	71	62	76	82
$<= 110\% \times avg$	74	40	55	63	62	78
$<= 125\% \times avg$	123	77	148	155	158	153
$<= 150\% \times avg$	141	57	121	137	126	109
$<= 200\% \times avg$	46	35	53	43	42	36

Table: Distribution of running time

Distribution of running time

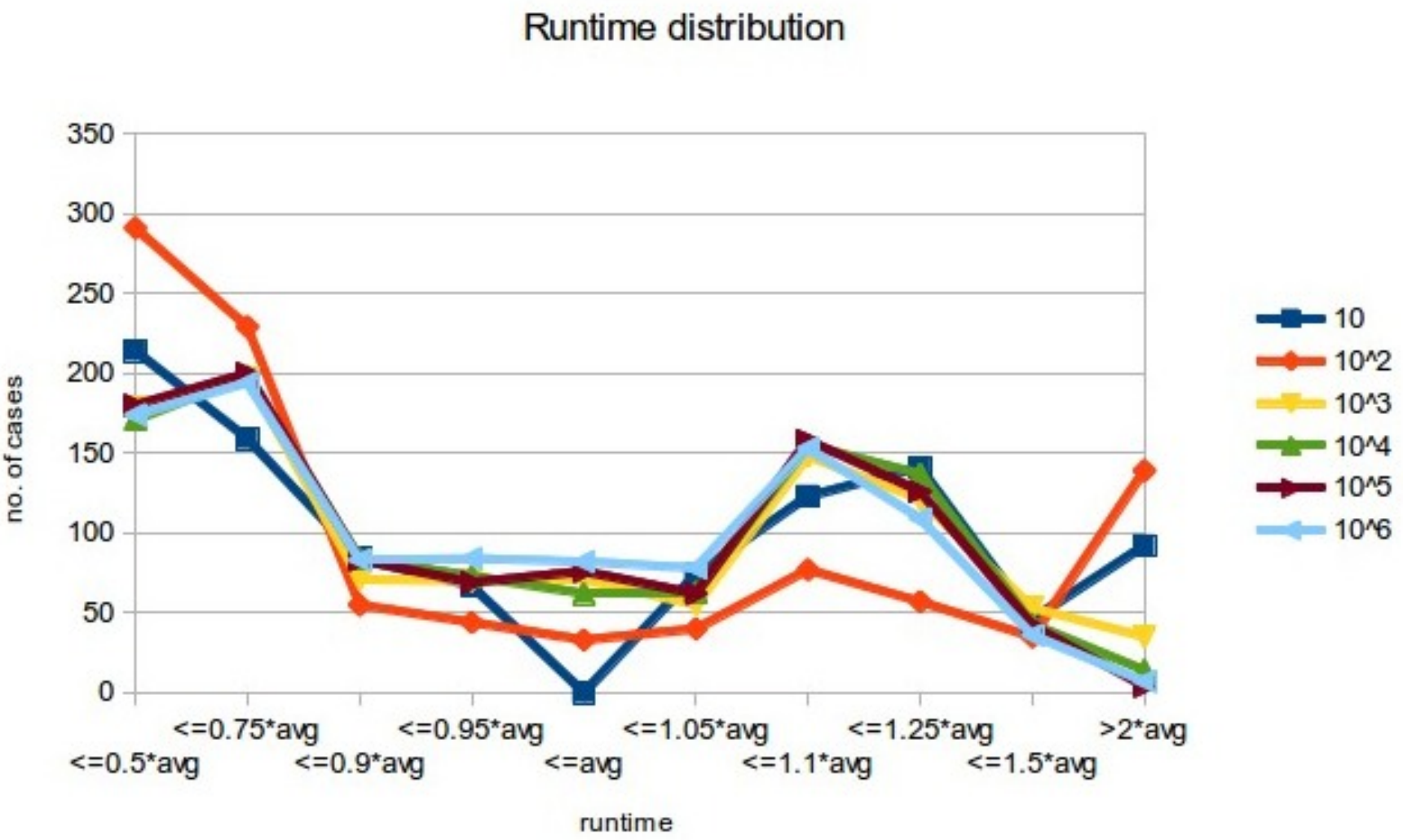
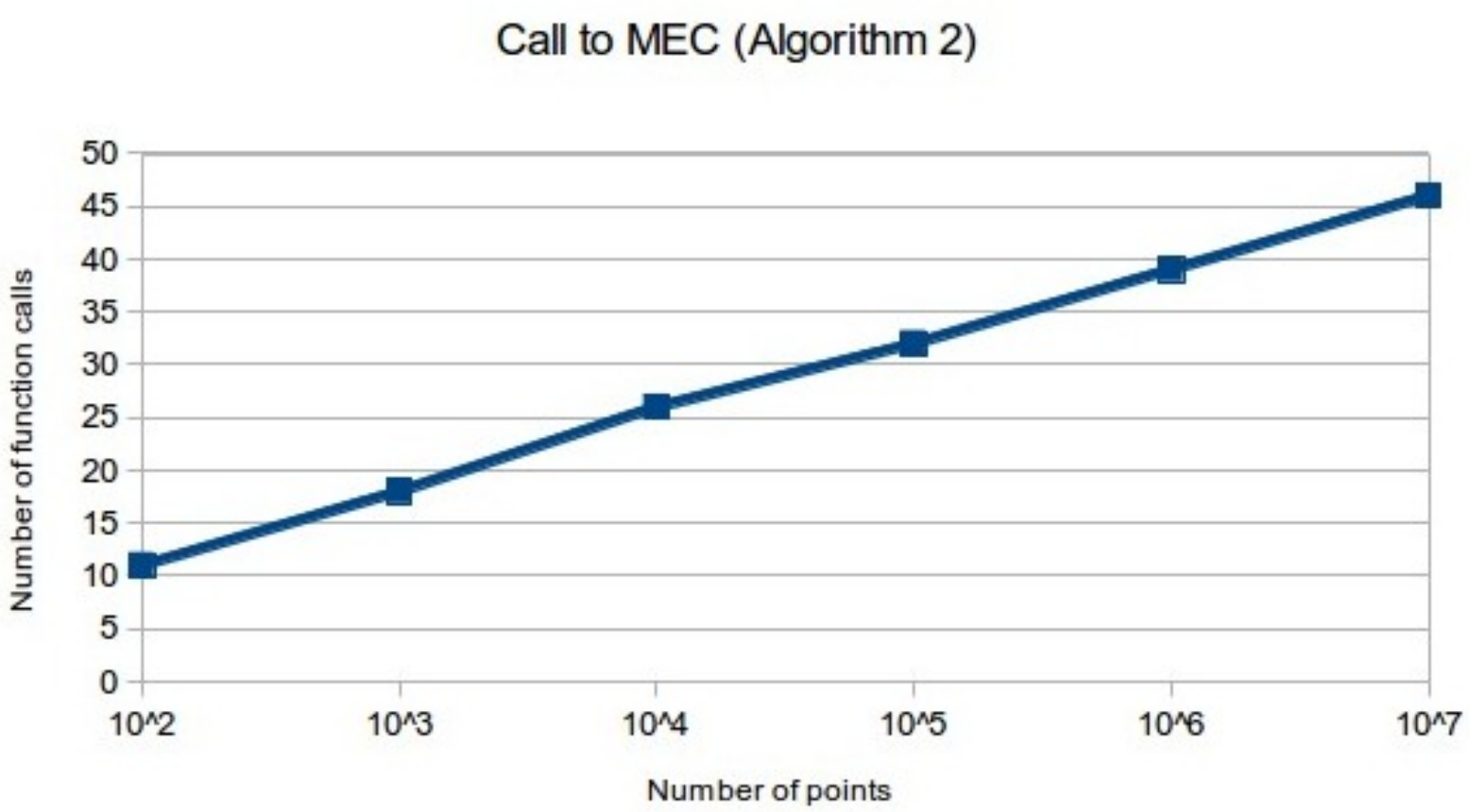


Table: Average number of calls to Algorithm 2 vs n

n	**Number of calls**
10	5
10^2	11
10^3	18
10^4	26
10^5	32
10^6	39
10^7	46

MEC calls vs n



The End