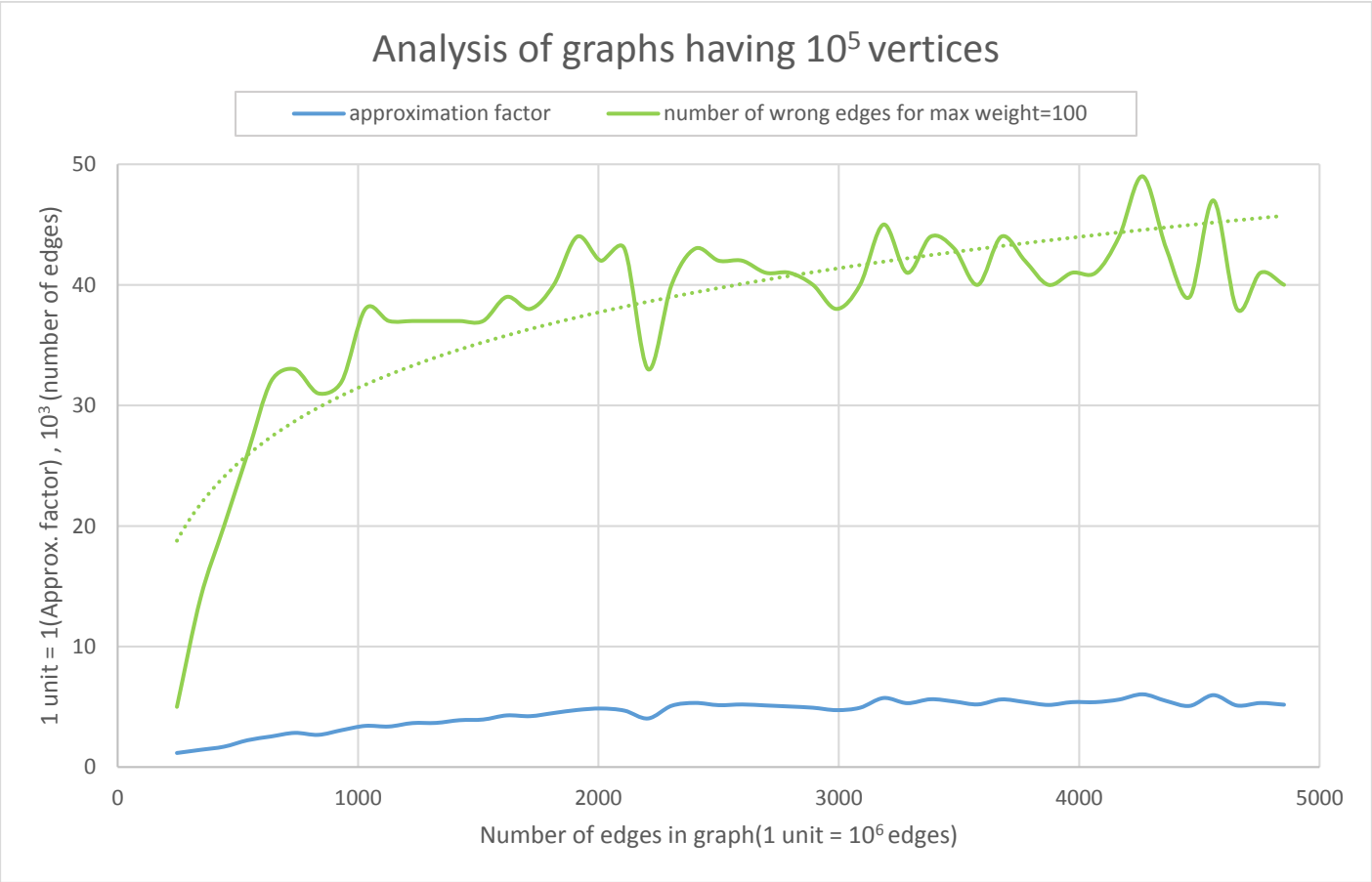
Complete Results after implementation

Algorithm1: Minimum spanning tree in Semi-streaming model

Approximation factor (in terms of number of wrong edges) is large because minimum weight can be 1 and maximum weight can be 100.

Also shown the error. Error means how many wrong edges were chosen during execution.



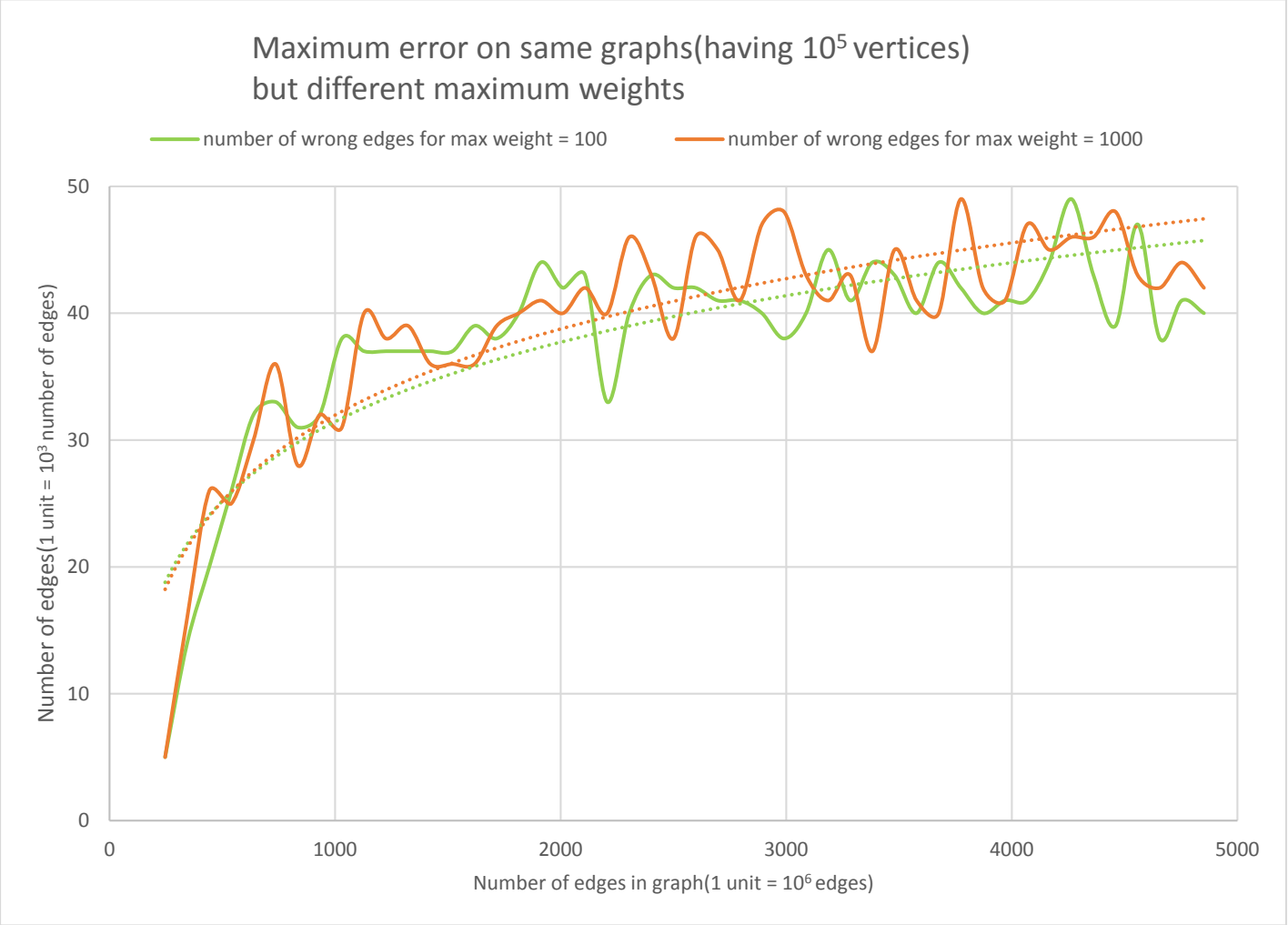
Observations:

Error is high in graphs having large number of edges.

Approximation factor is high.

But here we see that approximation factor is not good criteria to analyze the results as approximation factor increases drastically if maximum weight is very large and minimum weight is very small for a random graph.

So maximum error (maximum error means no of wrong edges added in worst case) is analyzed and shown below



Observations:

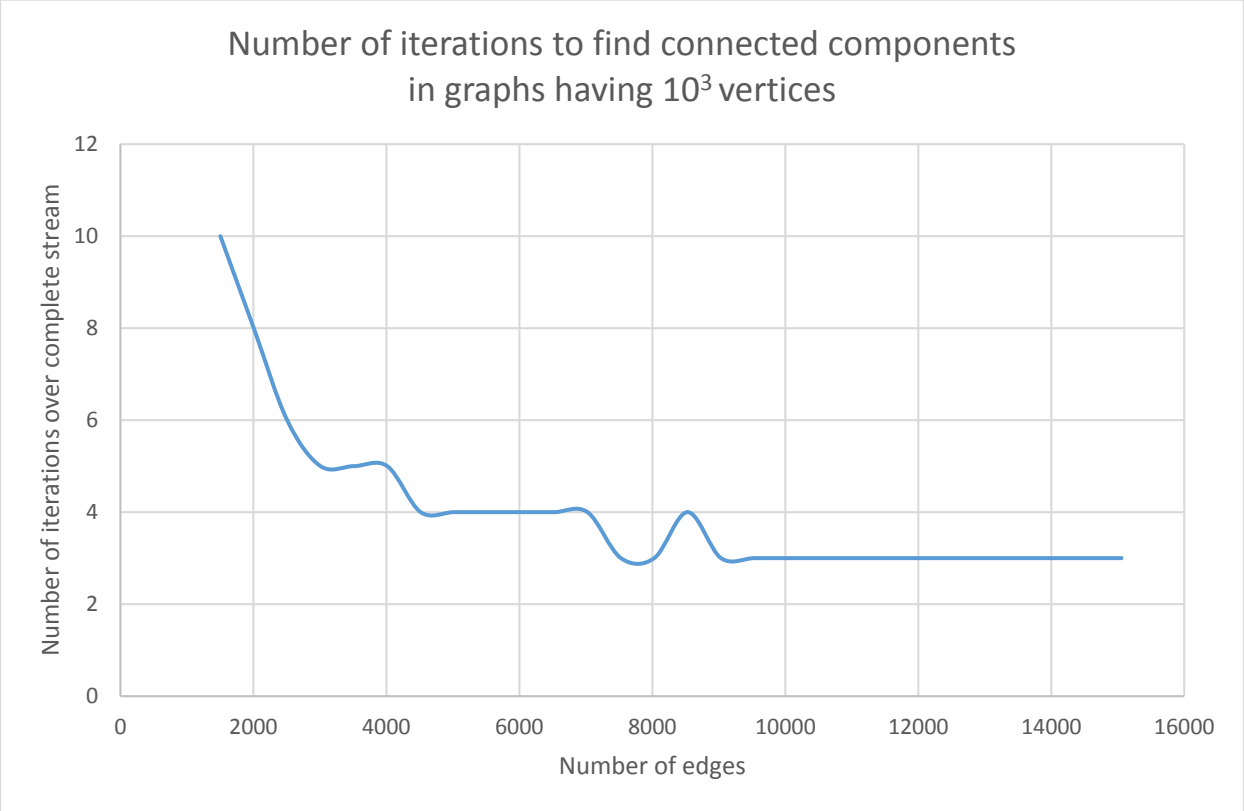
With large difference between maximum and minimum weights in random graphs, error follows the same behavior.

Algorithm2: Minimum spanning tree in W-stream model

Solving connected components can be done in W-stream model. It is an open problem, how to use connected component in W-stream to solve minimum spanning tree.

Algorithm3: Minimum spanning tree in Stream-sort model (Using connected components)

Connected components is calculated in O(log n) iterations on entire input stream using randomization. It is an important bound in this model. Finding connected components will be subroutine later in this algorithm.

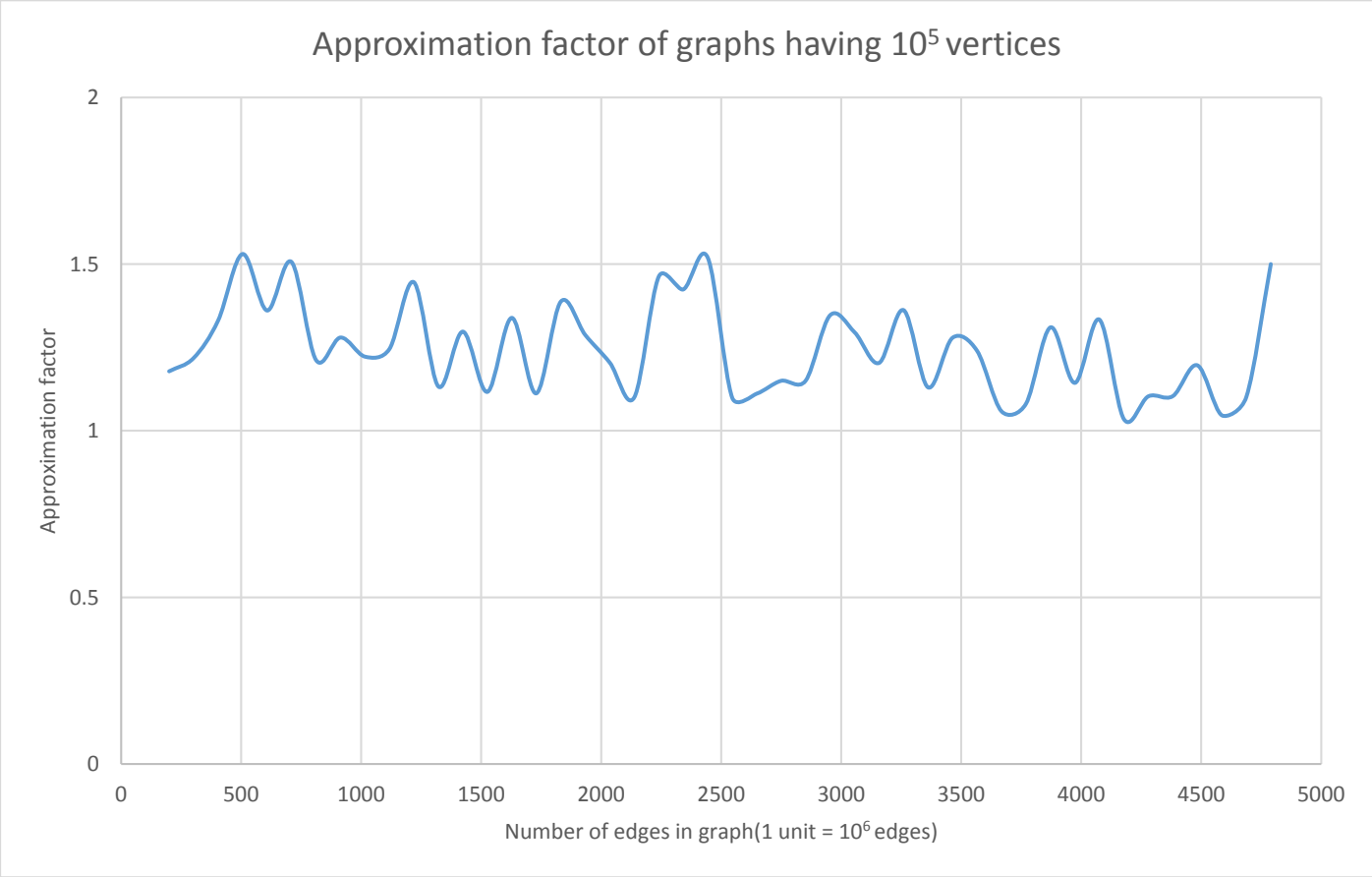


Number of iterations are not shown for both planer graphs (as it would be O(n)) and complete graphs(as it would be O(1)).

Here $n = 10^3$ and $\log_2(10^3)=9.96$

Minimum Spanning Tree algorithm here uses sorting of stream (External sorting was used) which is, in a way, of similar to Kruskal's algorithm.

Difference between maximum and minimum weight was kept low and we got low approximation factor.



Observations:

Better approximation is found.

Sorting the stream reduces the error and makes better bound.

Algorithm4: Streaming algorithm to find Minimum Enclosing Ball in higher dimensions.

It has better approximation factor than last semester's Timothy M. Chan Minimum Enclosing Ball algorithm.

Works faster even in higher dimensions.

Has better space complexity bounds (helps in case of large number of points in high dimensions).

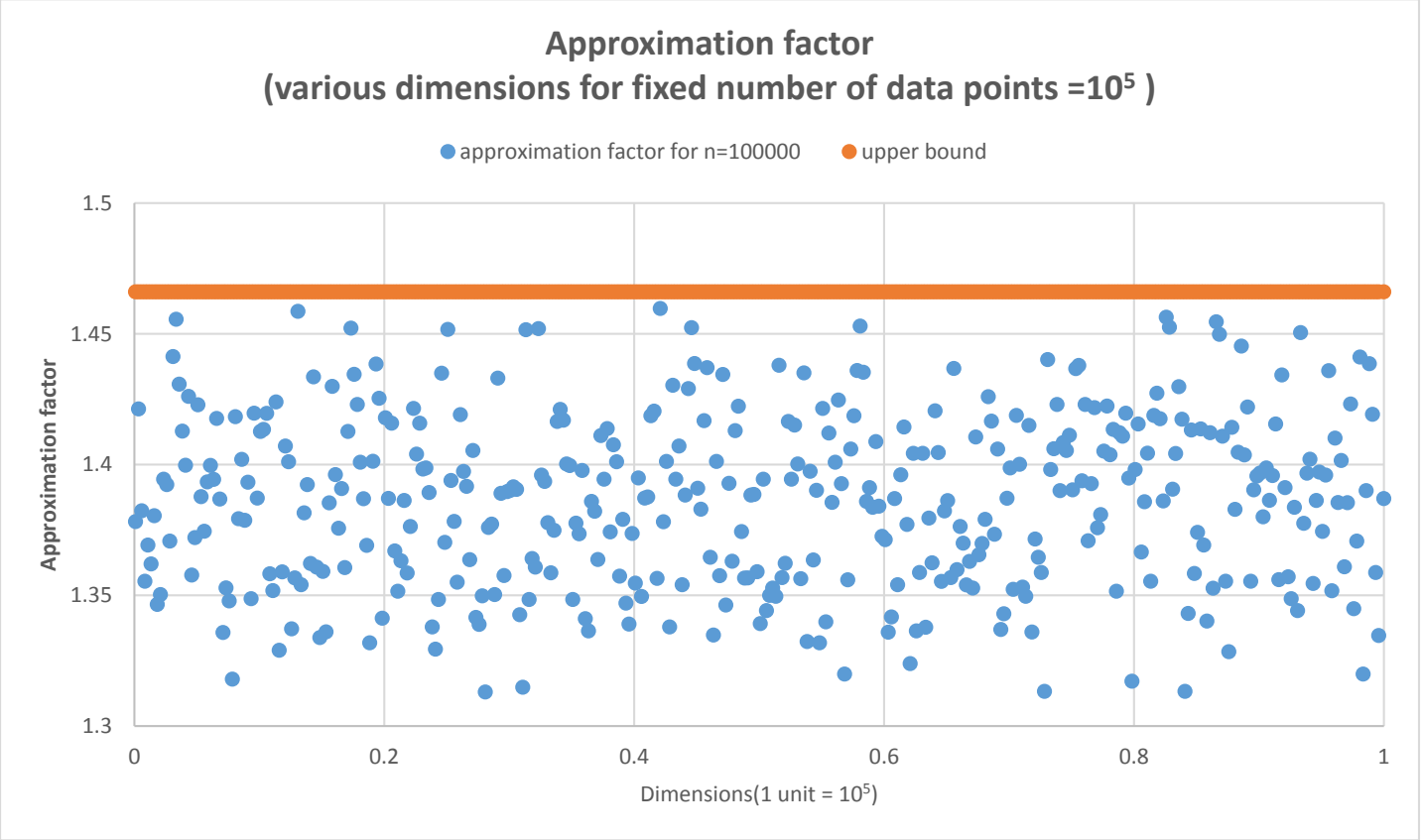
It uses two different algorithms, first is finding epsilon-expansions and second is finding minimum enclosing ball from set of balls.

It has theoretical upper bound on approximation factor which is equal to "$(1+\sqrt{3})/2 + \epsilon$" where $0< \epsilon < 1$

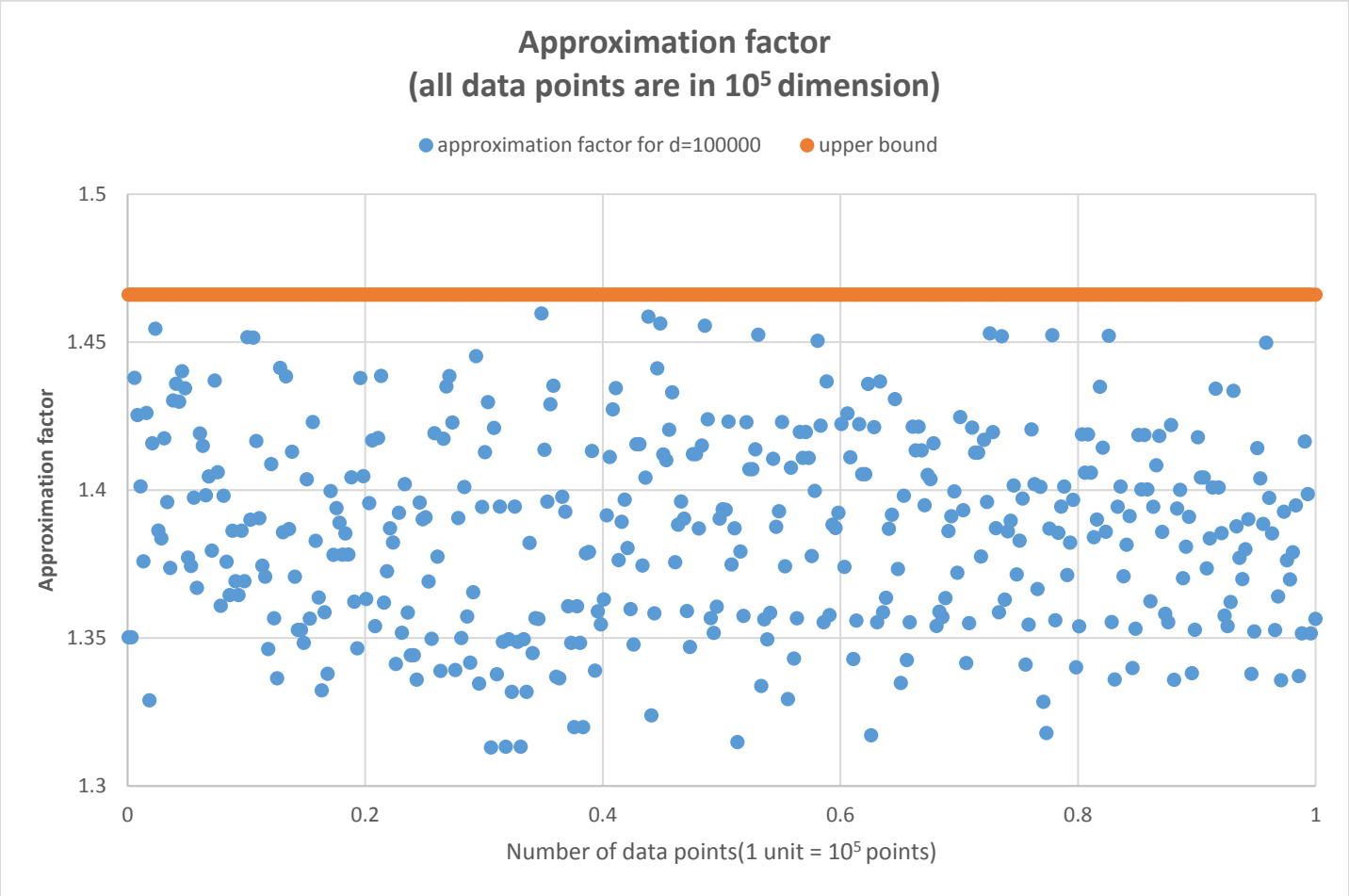
First- Results: bound on approximation factor = $(1+\sqrt{3})/2 + \epsilon$

$\epsilon = 0.1$ $(1+\sqrt{3})/2 + \epsilon = 1.466$

To check the effect of dimensions on data points, dimensions varies from 10 to 10^5 for fixed 10^5 data points.

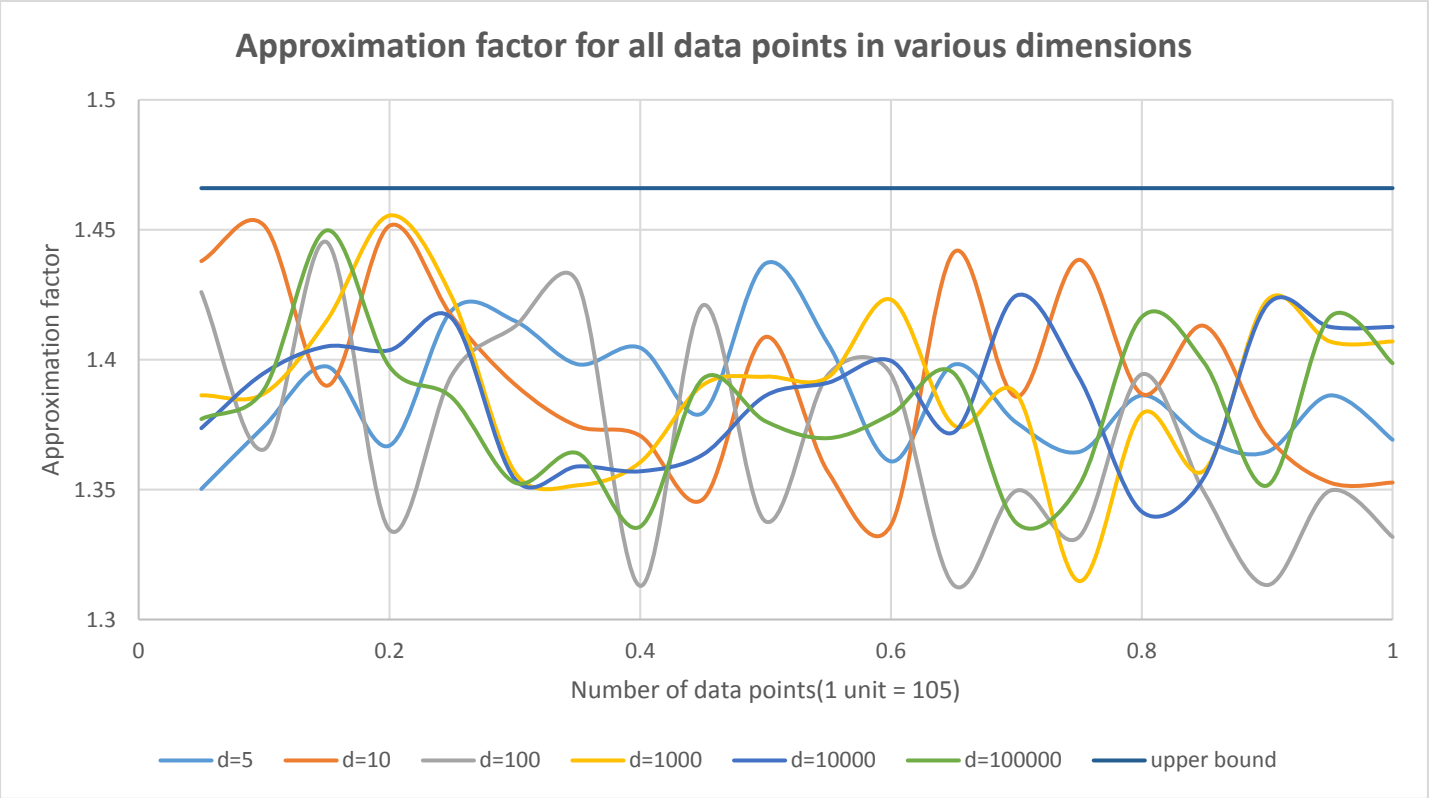


To check the effect of number of data points, no. of data points varies from 10 to 10^5 in fixed 10^5 dimension.



To check the effect of data points and dimensions in one plot

(here all the results are not used so that the plot looks clear)



Observations:

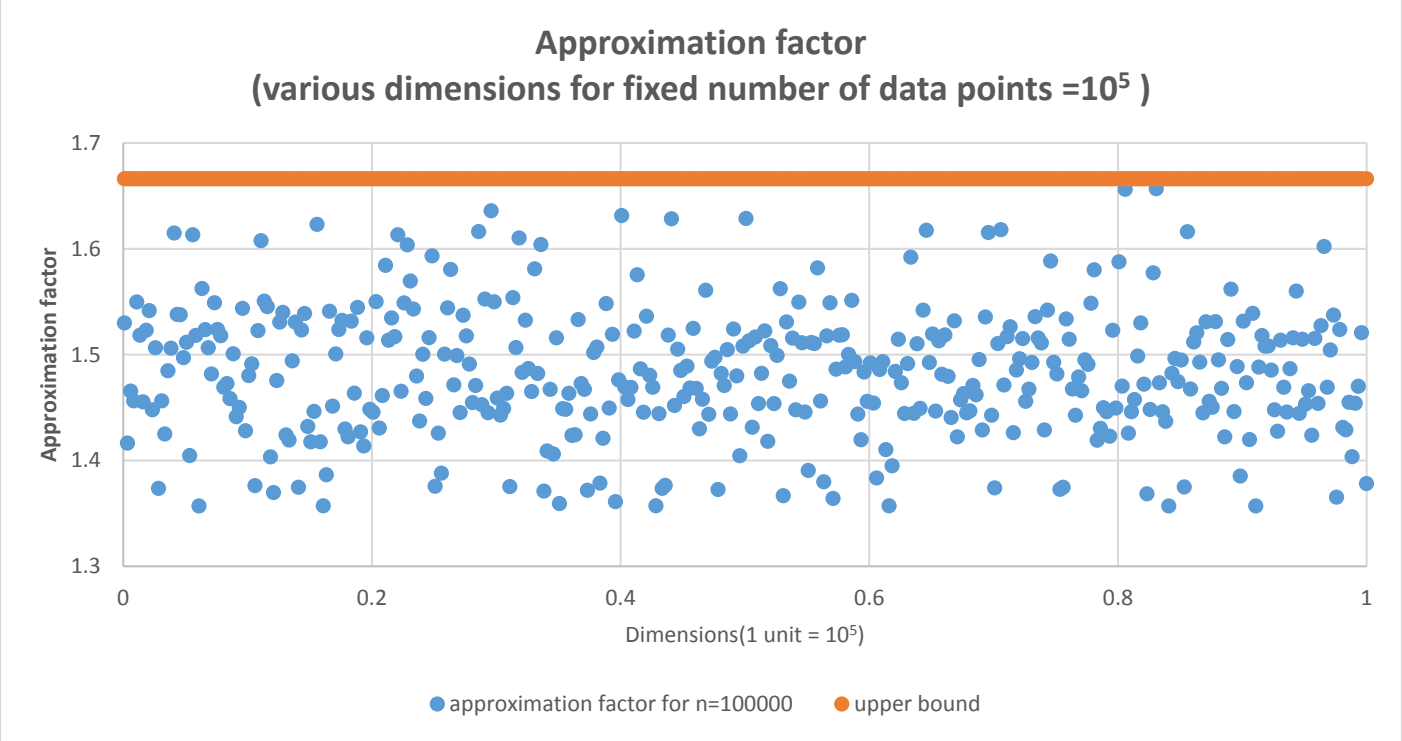
No instance crossed the upper bound.

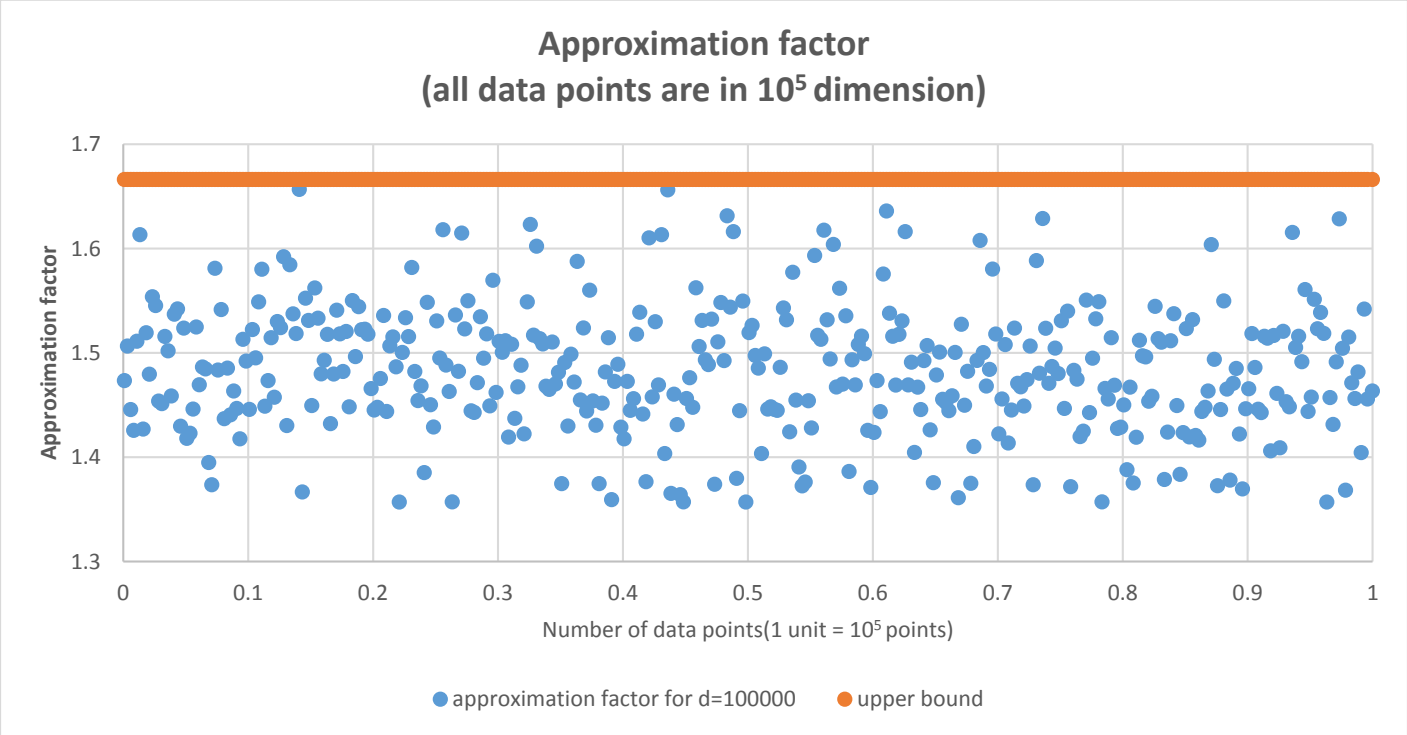
None went below a certain point

Does above algorithm has some lower bound, say 1.3 or 1.2? (Not proven/analyzed in the original paper but results, after implementing it, strongly suggest this)

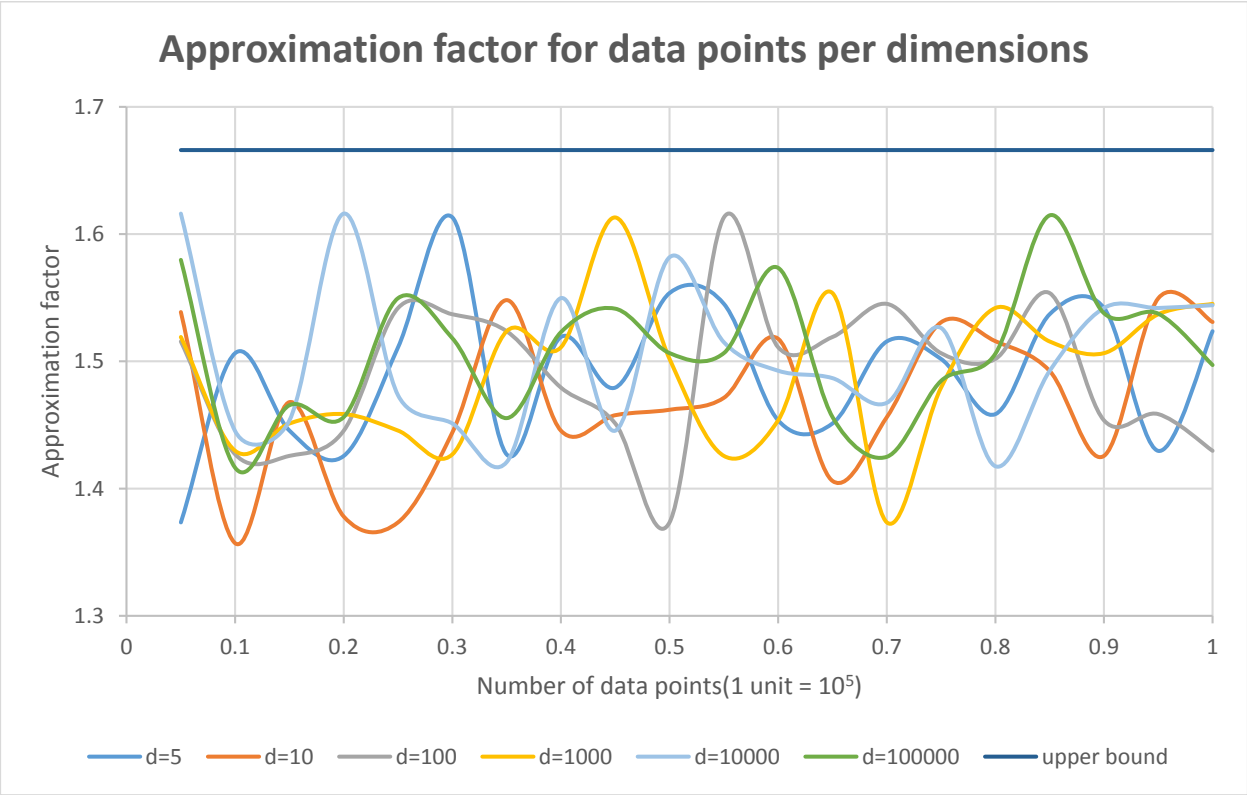
Similar results were found for different values of epsilon

$\epsilon = 0.3$ $(1+\sqrt{3})/2 + \epsilon = 1.666$





To check the effect of data points and dimensions in one plot (here all the results are not used)

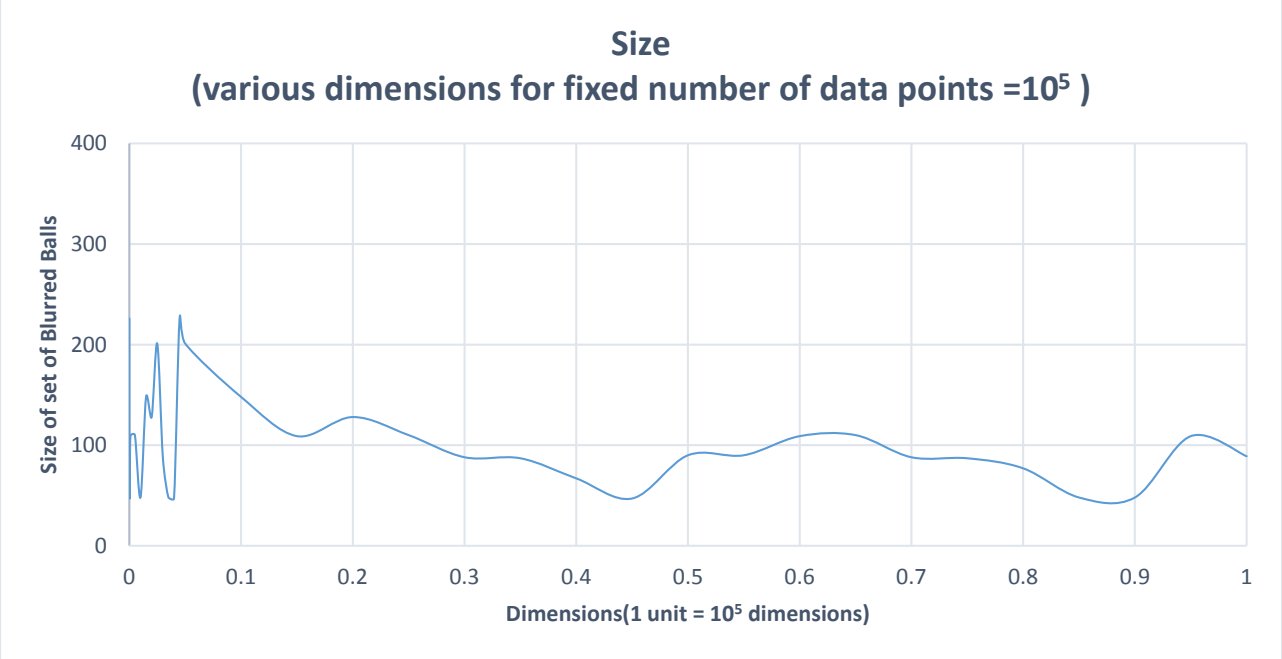


So does above algorithm has some lower bound?

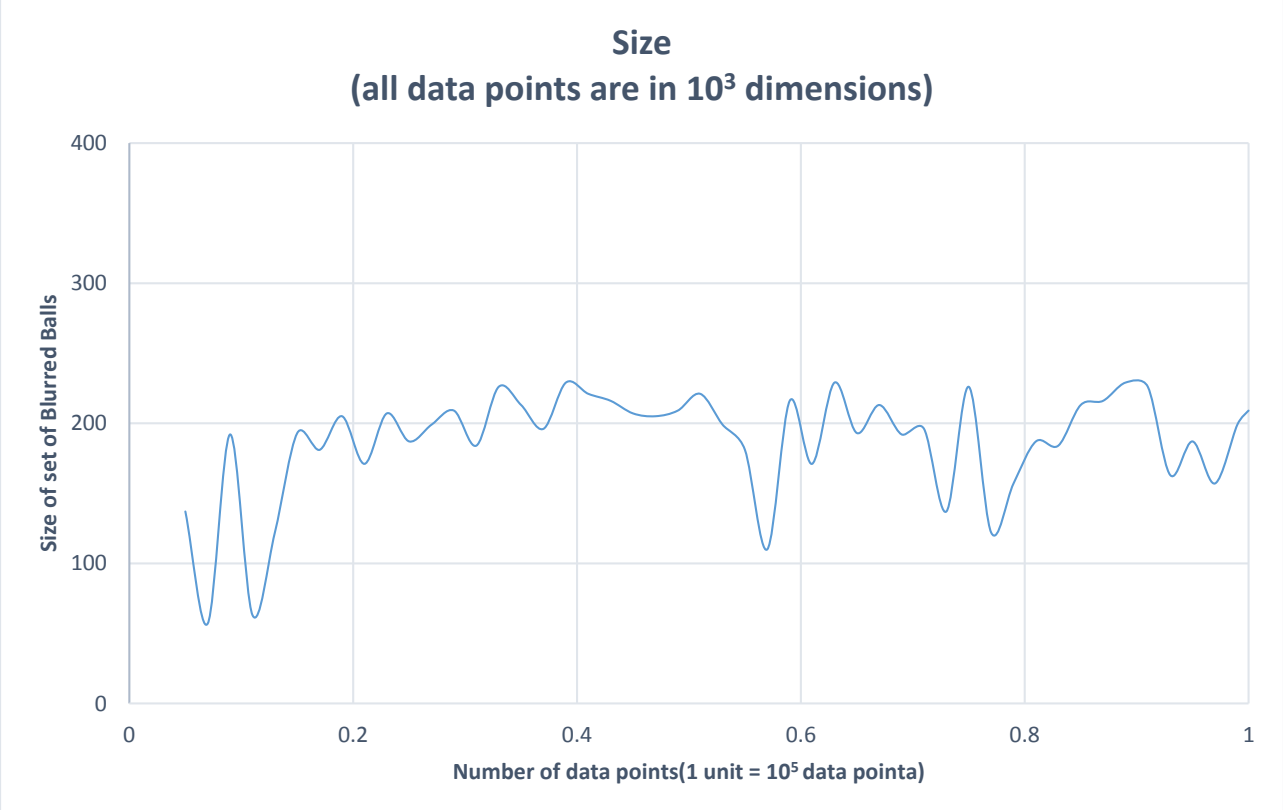
Second- Results: bound on size = $O((1/\epsilon)^2 \log(1/\epsilon))$

$\epsilon = 0.1$ $((1/\epsilon)^2 \log_2(1/\epsilon) = 332.192$

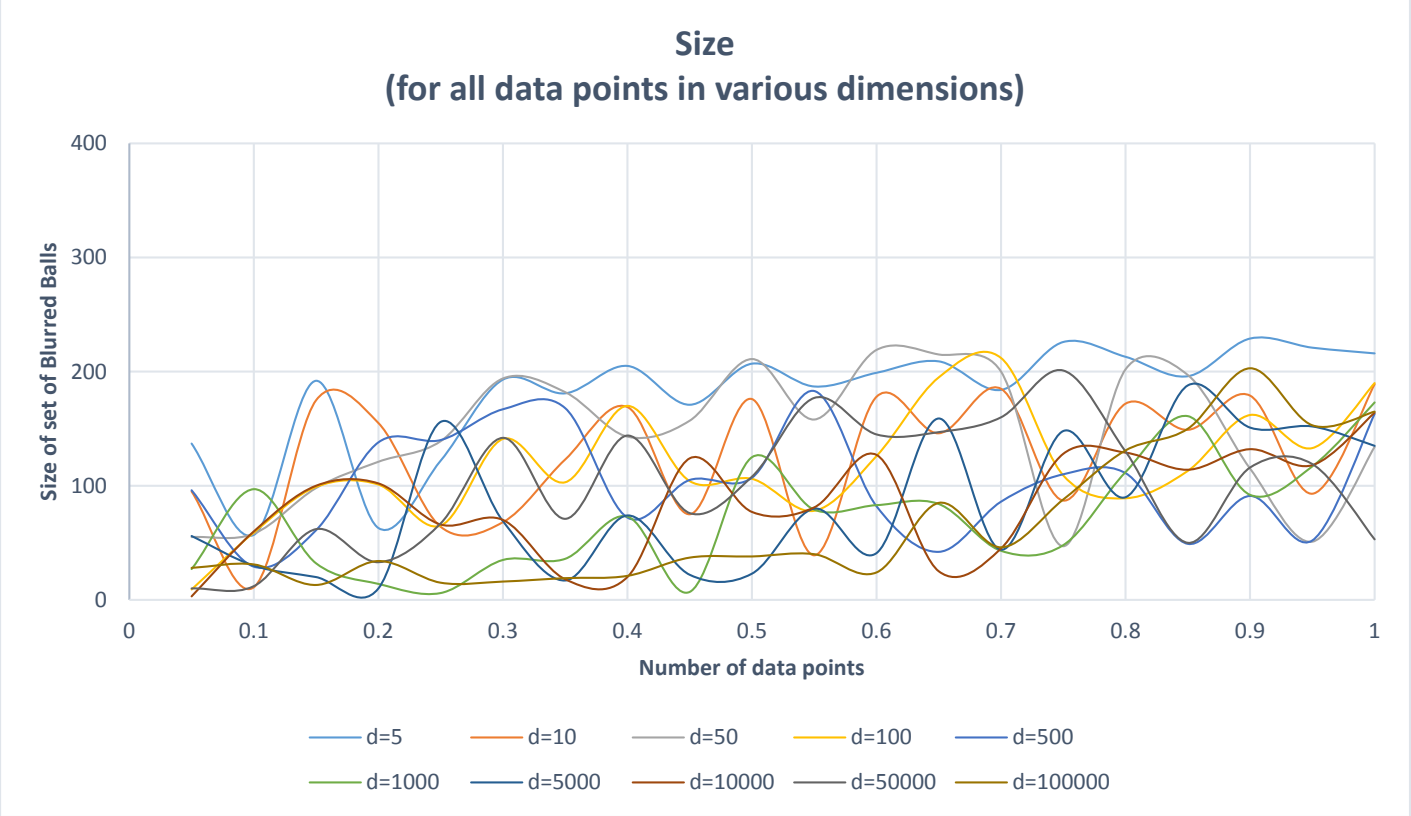
Effect of dimensions



Effect of number of data points



Effect of data points and dimensions



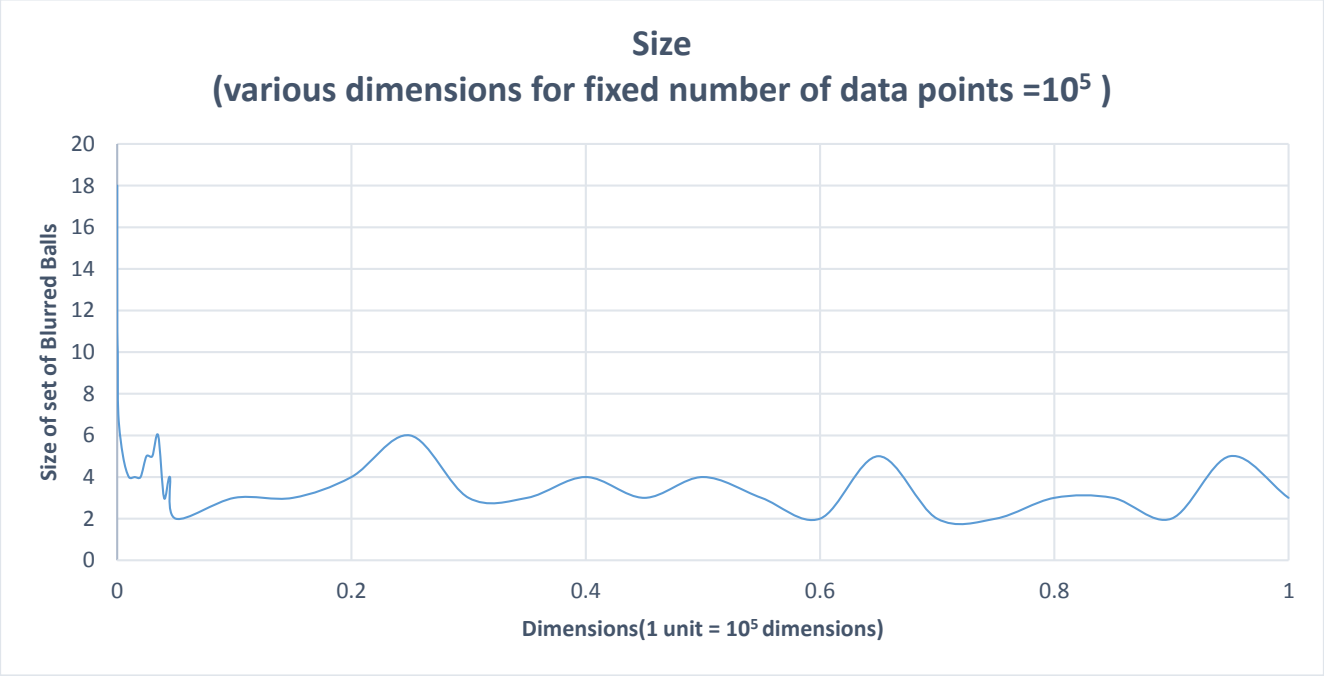
Observations:

At large no of data points in low dimensions, we get bigger size.

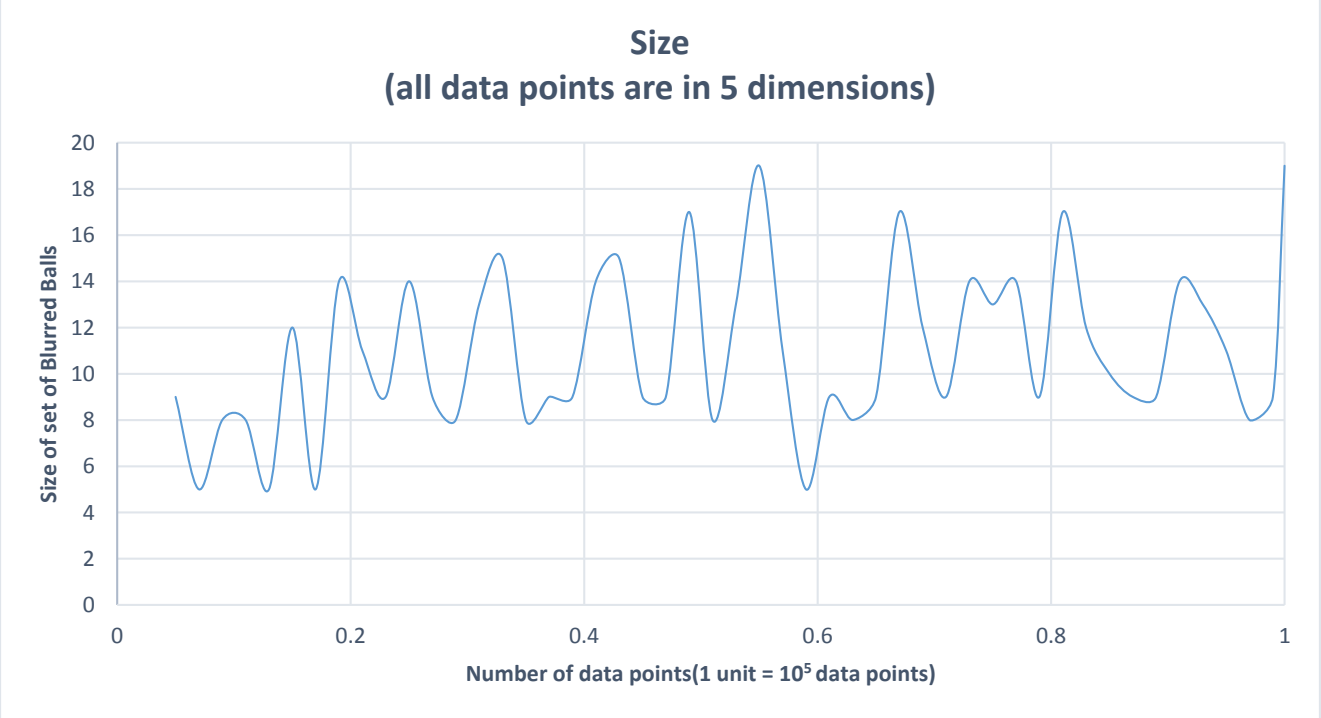
Similar observations are seen in results for other values of epsilon.

$\epsilon = 0.3$ $((1/\epsilon)^2 \log_2(1/\epsilon) = 19.2996$

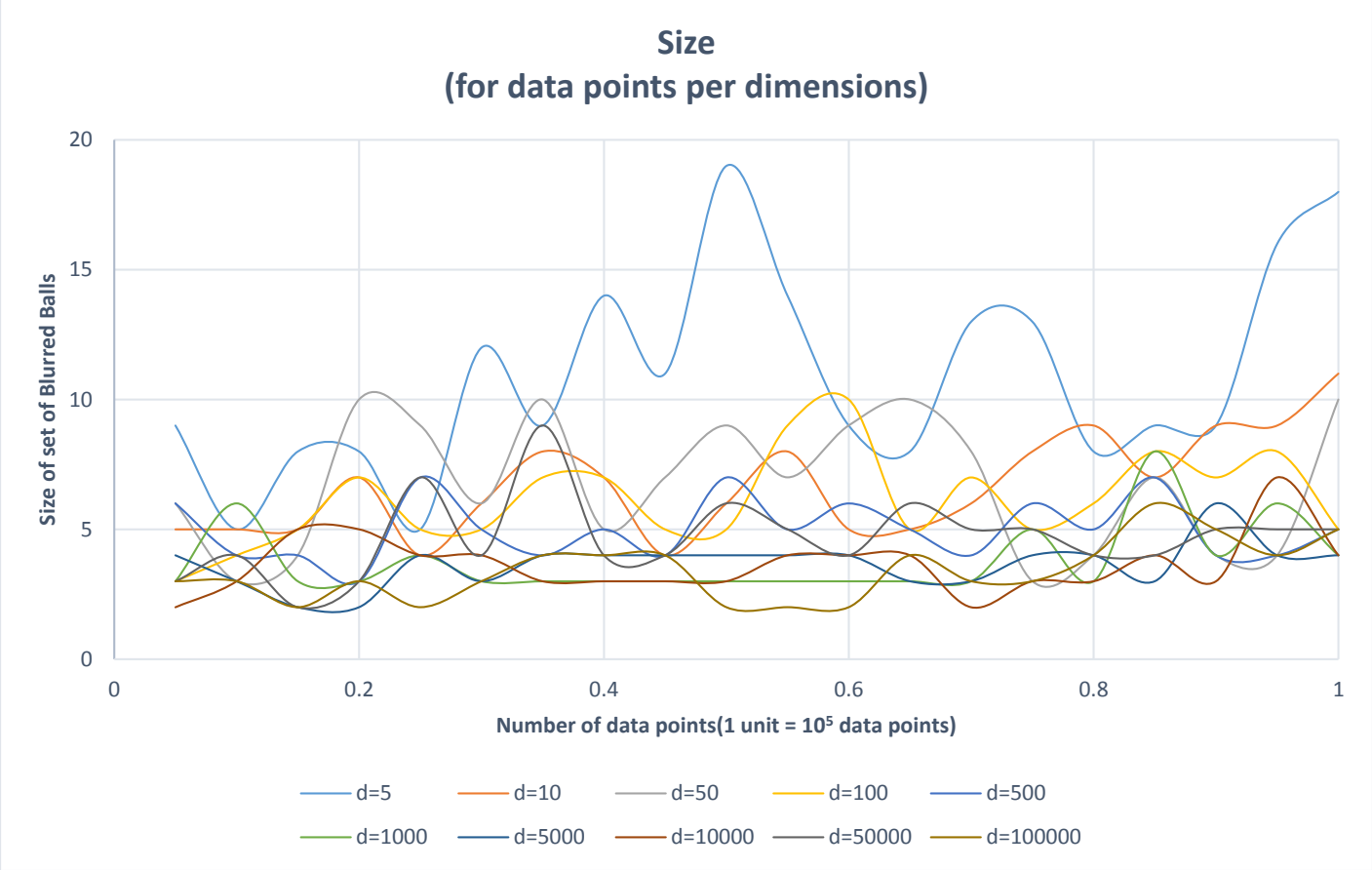
Effect of dimensions



Effect of number of data points



Effect of data points and dimensions



Observations:

Size is bigger for large no of data points.

Size is bigger for smaller dimensions.