COL380 Lab-1

Question2

Implemented the Quick Hull algorithm.

Made recursive function calls parallel using openMP sections. Made a global variable to keep count of total number threads so that number of threads don’t exceed the number specified.
A new thread is created only when current number of threads is less than the number of threads specified.

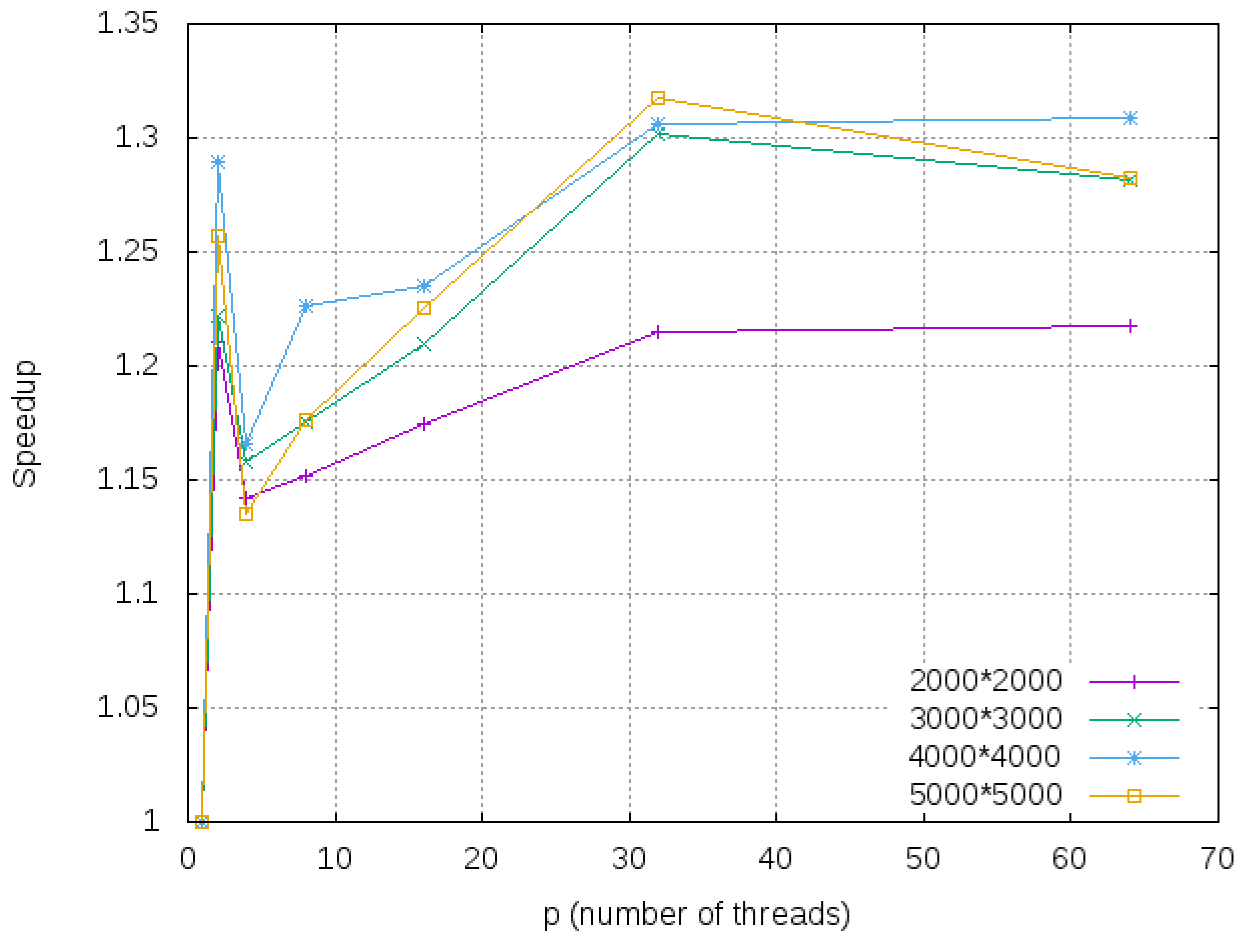
The global variable should be incremented or decremented atomically to avoid data race.

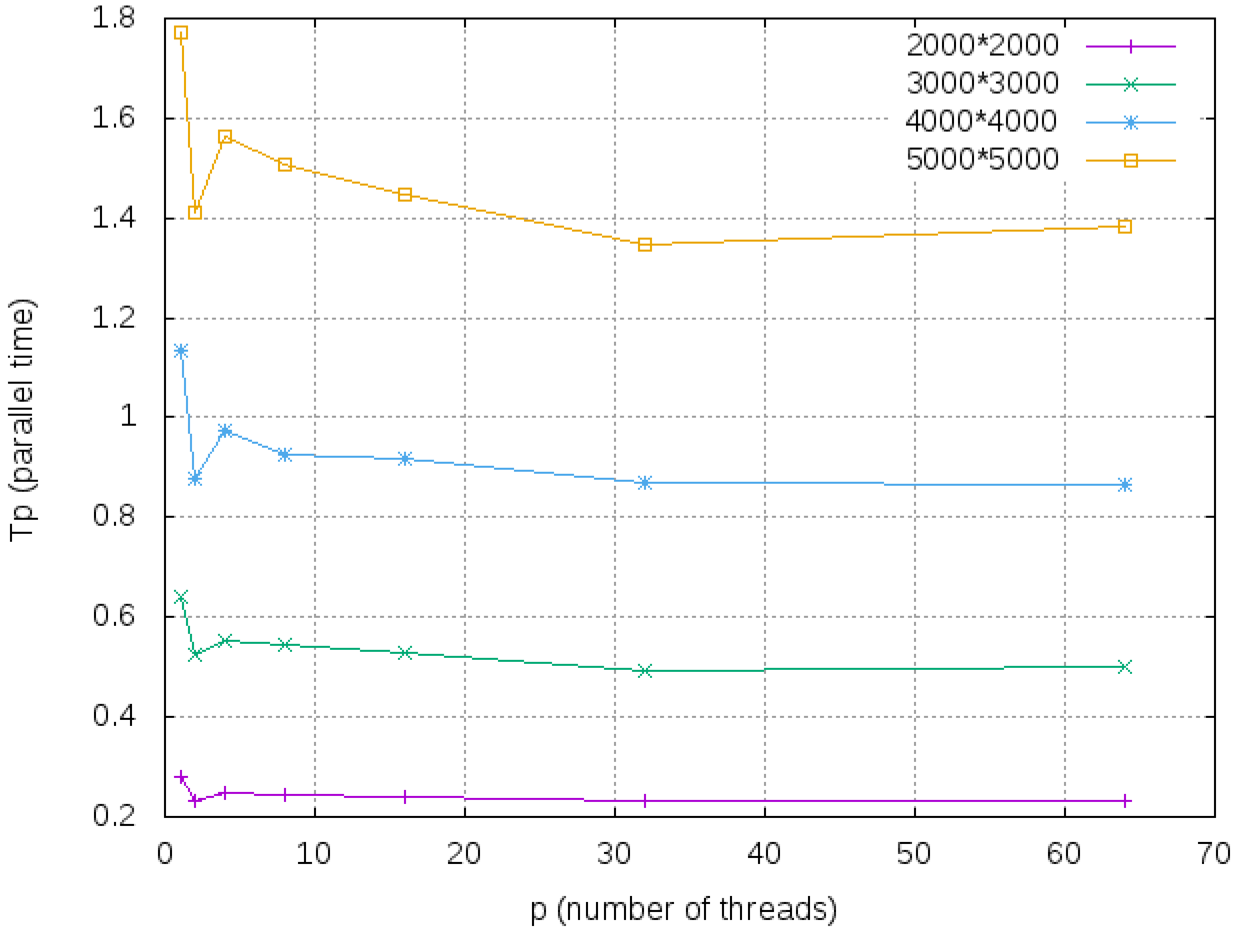
The global vector which contains the points in the convex hull should be updated in a critical section so that 2 vector don’t push into it at the same time, vectors in C++ are not thread safe.

Loops should not be parallelized in Quick Hull algorithm because then almost everything inside loop will have to be put in critical section, which will make the loop serial and also add the overheads and it increases the execution time.

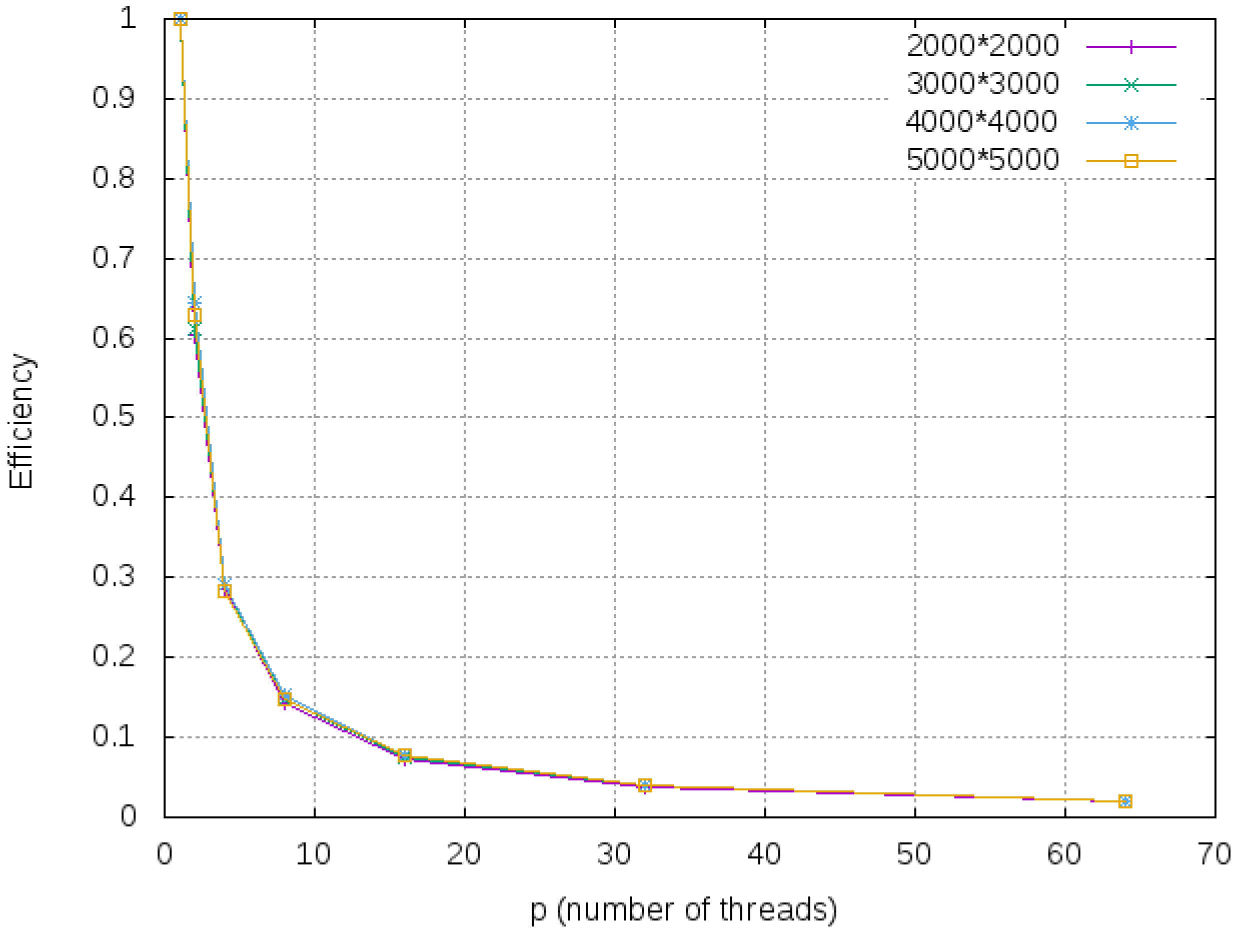
Generated the inputs randomly, tested on input size = 2000*2000, 3000*3000, 4000*4000, 5000*5000 . The code was run on a machine with 2 physical cores and 7.7 GB of RAM. Time was calculated after averaging over multiple runs.

The graphs for speedup, efficiency and time taken are:





Tp (parallel time)
p (number of threads)
2000*2000
3000*3000
4000*4000
5000*5000
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
0
10
20
30
40
50
60
70



Efficiency
p (number of threads)
2000*2000
3000*3000
4000*4000
5000*5000
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
10
20
30
40
50
60
70