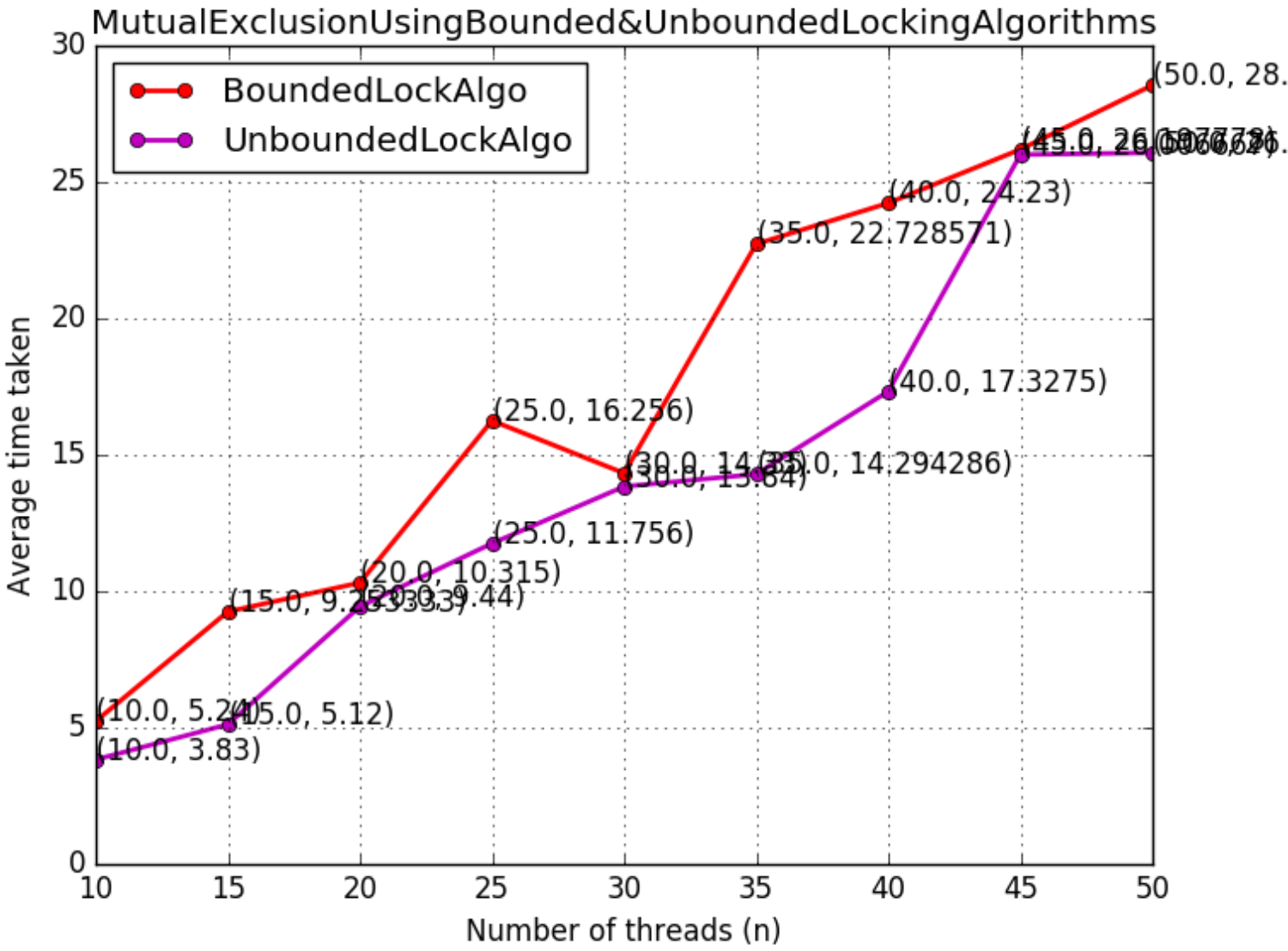
Mutual Exclusion with Bounded and Unbounded Waiting Times

Report

Aim : To implement bounded mutual exclusion algorithm for n-threads using TEST_AND_SET (atomic) operation.

Graph representing average time taken by a thread (in milliseconds) to enter its CS v/s number of threads (n) :



The above graph shows the average time taken by each thread v/s the number of threads in both Bounded as well as Unbounded mutual-exclusion locking algorithms.

A *critical section* (CS) is a sequence of instructions that can be executed by at most one thread at the same time. In order to explain the graph, we need to know what each of the implemented algorithms mean in context with the critical section (CS).

Bounded waiting : This algorithm involves a bound on the number of times other threads are allowed to enter their corresponding critical sections after a thread has already made a request to enter its critical section and before that request is granted.

Normal/Unbounded waiting : If no other thread is currently executing in its critical section and some threads wish to enter their critical sections, then in this case, only those threads which are not executing in their corresponding remainder sections can participate in deciding which thread will enter its CS next, and this selection cannot be postponed indefinitely. This clearly suggests that it will take a limited amount of time to select a thread to enter its CS. In particular, no deadlock will occur. And after this,the thread will enter its CS and do its work.

Thus, this is a key factor in causing lesser delaying of a thread (since limited amount of time for thread selection), for entering in its critical section as compared in Bounded waiting, after it has made a request to enter the same, and the same observation can de inferred from the graph.

Therfore, the average time taken by a thread to enter its critical section is *generally* higher in **BoundedWaitingLockingAlgo** than in **UnboundedWaitingLockingAlgo** with mutual-exclusion.

Anomalies :

- ➔ Sleep function of the **unistd.h** library is used in the above programs, which enforces a delay for a specified amount of time for the program invoking it. So if the argument is something between 0 and 1, it gets typecasted as 0, and thus may cause errors, as the desirable amount of delay may not be achieved as expected in the critical section.

➔ The program may have average time taken by a thread to enter its critical section, more in Unbounded locking algo than in Bounded locking algo, since this depends on CPU state, as well as other factors such as preemption and starvation of low-priority threads.