Oscar Galindo

Lab 1

Dr. Shirley Moore

Lab 1

1. Graphs

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

Graphical user interface, application, email

Description automatically generated

1. Execution of br-17(modified as needed to actually run) in static:

Text

Description automatically generated

Execution of br-17(modified as needed to actually run) in dynamic:

Text

Description automatically generated

Given that if we add a new node, we would a problem 18 times as big as the br-17, then we can see that if were to “evenly” divide the work then every core would have 2.25times more work, hence if we were to multiply the time take by the current times we would see that the static implementation would take 454.29 seconds \* 4 ~ 1022seconds ~17 minutes, which is in this case non-feasible. On the other hand the dynamic implementation could vary quite significantly, taking the relationship linearly as previously stated would output a time of 317.76 seconds \* 2.25 distributed work factor ~ 714.96 seconds ~ 12 minutes, close, quite possibly in some cases we may not see a termination and in some others we may see a termination.

Experiment with br-17 (modified to have 18 nodes):

Dynamic execution with partition set to 5:

Yellow text on a green background

Description automatically generated with medium confidence

Static execution:

1. For this problem the control of the reading and writing of the best tours is controlled by a read/write lock instead of using the mutex that was originally included in the code. The modifications done are:

To control the access in the reading sense:

Text

Description automatically generated

To control the access to the best tour in the writing sense:

Text

Description automatically generated

Output times

1 thread:

Text

Description automatically generated

2 threads:

Text

Description automatically generated

4 threads:

Text

Description automatically generated

8 threads:

Text

Description automatically generated

Conclusion: The performance is positively affected by the inclusion of a read/write lock. Theoretically, this is possible due to the less overhead of the lock mechanism operations (lock/unlock) than the mutex lock and unlock procedures. Also, the fact that now no threads will get outdated data could help to finish the execution faster.