Chirag Soni: 665930262
Sanamdeep Singh Anand: 677787570

PARALLEL PROCESSING: LAB4

GOAL: To solve the travelling salesman problem using branch and bound algorithm in a Message Passing Interface (MPI) system. The program read an input data file that contains the number of cities and their x, y coordinates. The program finds the cost matrix, and then compute the best path it travels between the cities. The matrix was created by the source node (process 0), then it was distributed to the other process to calculate the best cost and the best path.

GIVEN:

- n, size of the TSP: used small data set used in solving Drilling Problem by *Ludwig.*
- p, number of processors: p = 1,2,4,8,16,32,64

MPI Functions used in the ring and hypercube algorithm:

1. **MPI_INIT**: Initiate an MPI computation. (Called only once during the execution of a program)
2. **MPI_COMM_SIZE**: Determine number of processes.
3. **MPI_COMM_RANK**: Determine my process identifier.
4. **MPI_SEND**: Send a message.
5. **MPI_RECV**: Receive a message.
6. **MPI_Get_processor_name (processor name, &name_len)**: Returns the name of the processor on which it was called now of the call.
7. **MPI_Wtime:** Returns the floating-point number of seconds, representing elapsed wall-clock time since some time in the past.
8. **MPI_Cart_create(MPI_Comm comm_old, int ndims, const int dims[], const int periods[], int reorder, MPI_Comm *comm_cart)** : This command is used to create a new communicator to which the communication information has been attached.

The ndims specifies the number of dimensions in our cartesian grid, dims tell us the number of processors in each dimension and periods is a logical array that specifies whether the grid is periodic or not.

9. **MPI_Cart_coords (MPI_Comm comm, int rank, int maxdims, int coords[]):** This method is used to specify the cartesian coordinates of the specified processor within the implemented cartesian topology.
10. **MPI_Cart_rank(MPI_Comm comm, const int coords[], int *rank)** : This command is used to specify the rank pf the given processor in the network.

Formulation:

A modified version of parallel depth first branch and bound algorithm is implemented to solve a symmetric travelling salesman problem. The parallelization of the branch and bound is fully distributed in such way that each processor performs the same sequential algorithm on a different branch of the tree. This algorithm performs a depth first search through the solution space. It cuts off a branch whenever its bound is worse than the best solution found up to that point.

The codes take a data file consisting the total number of cities and their x, y coordinates. These values are stored in a 2D array. The matrix is created by the source process. This matrix is distributed to all the processors in a way such that each node is assigned some branch of the search tree to find the path with the lowest cost of that branch. The program starts to compute the least cost path by comparing the current cost of a path to the least cost. The node finds the lowest cost path, it will store the best cost. Once the entire branch is traversed, the best cost is sent back to the source node. Now, the processor 0 or the source processor compares the best cost to its current best cost and update the cost with the lower value, the visited city and the best path.

Chirag Soni: 665930262
Sanamdeep Singh Anand: 677787570

Results:

P	N	Time	Speed Up	Efficiency
1	10	0.101	0.0061	0.07
2	20	0.105	0.01	0.097
4	30	0.124	0.014	0.12
8	40	0.144	0.024	0.178

1	10	0.101	0.0061	0.07
2	10	0.103	0.02	0.099
1	20	0.121	0.012	0.14
2	20	0.142	0.022	0.175

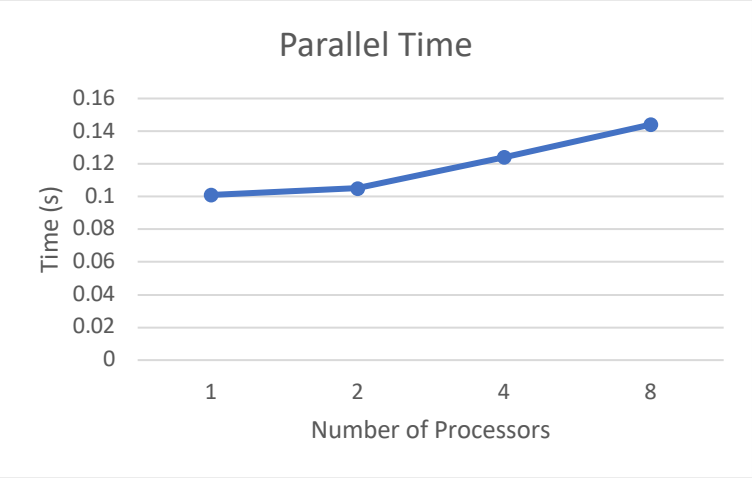

Figure 1

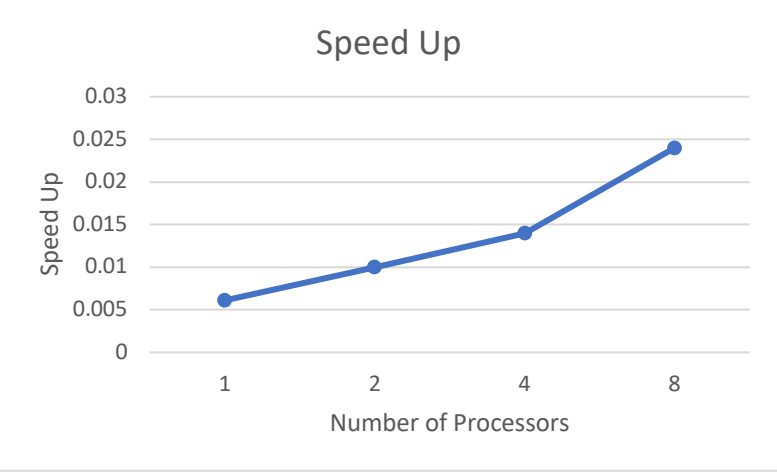

Figure 2

Chirag Soni: 665930262
Sanamdeep Singh Anand: 677787570

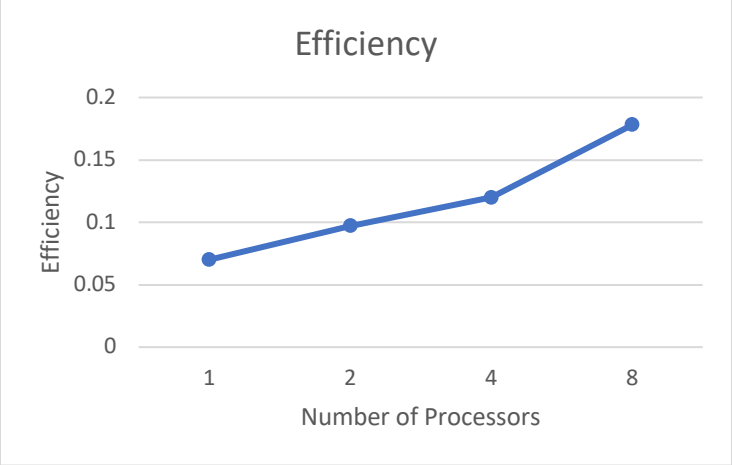

Figure 3

Analysis:

- From the experiment it can be analyzed that each node must wait for its leaf nodes to finish the processing. It then compares and update the path with the lowest cost.
- From the above results, it can be deduced that on increasing the number of processors, the time, speedup, and efficiency also increases.
- The time might be increasing due to the increased intercommunication delay due to increase in number of processors.
- The speed up is directly proportional to the serial time and the serial time increases with the increase in number of processors (serial time increases in a greater proportion as compared to the increase in the parallel time). Hence, the speed up is also increasing.

- Efficiency is directly proportional to the speed up, so, increase in speed up also increases the efficiency. Moreover, it can be concluded that the depth first branch and bound algorithm achieves a linear speedup.

Lesson learnt:

- This programming assignment helped us to learn a parallel programming technique to solve the NP-hard problem like travelling salesman problem. The problem is solved in less time.
- A parallel algorithm can reduce the time and increase the performance, but as the input size increase, the time will increase.
- More processors require more computation time due to another workload of the node.
- In also helped us to learn a branching strategy of a current set into subsets. After this it calculates the solution and each subset cost. It improves the lower bound for each sub-problem.