CS3523: Operating Systems - II

Programming Assignment 1 - Report

Soumi Chakraborty

ES19BTECH11017

1 Overview

This program parallelises the brute force approach to solve the travelling salesman problem. Given a starting point, the number of threads and a list of destinations points, this program finds the closes point by finding the distances to all the endpoints and then chooses the minimum distance. However, the program makes use of threads to do so by splitting up all the destination points over multiple threads to speed up the process.

2 Threads and Multithreading

A thread is a path of execution within a process. Threads belonging to the same process share memory space. An operating system that supports multithreading allows multiple threads to execute concurrently. Multithreading parallelises and speeds up code which would have otherwise taken a very long time to run had it been sequential.

3 Implementation

3.1 Headers

The program uses the following header files:

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <math.h>
#include <string.h>
#include <sys/time.h>
```

3.2 Macros and Global Variables

The program makes use of macros to set the max limit for all the arrays being used throughout the program. It also includes a few global variables so that all the threads and the thread function (`min_dist()`) has access to them throughout the scope of the program.

The descriptions and use of each global variable have been elaborated upon in the source code file as comments beside each declaration.

3.3 Functions

The program includes two functions to help calculate and choose the minimum distance between two points.

`euclidean_dist()`
This function accepts a pair of coordinates and returns the euclidean distance between them.

`min_dist()`
This function is the thread function. It iterates through the subset of points assigned to one particular thread, calculates and stores their euclidean distances in an array and also computes the minimum distance per subset of points.

`main()`

The main function mainly deals with reading the input file, creating the threads, joining them after they finish executing and finding the overall minimum distance. It also computes the time taken for the code to find the minimum distance.

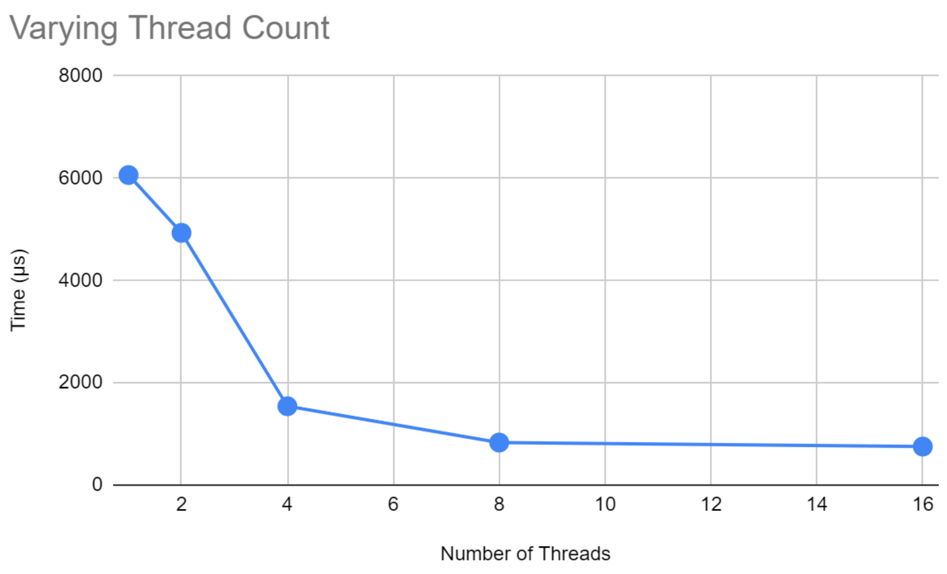
Since the input file is read as a string, to find and convert the coordinates into numbers, the code makes use of the `atoi()` and `strtok()` functions.

The function of each line of code has been further explained in the source code file.

4 Analysis

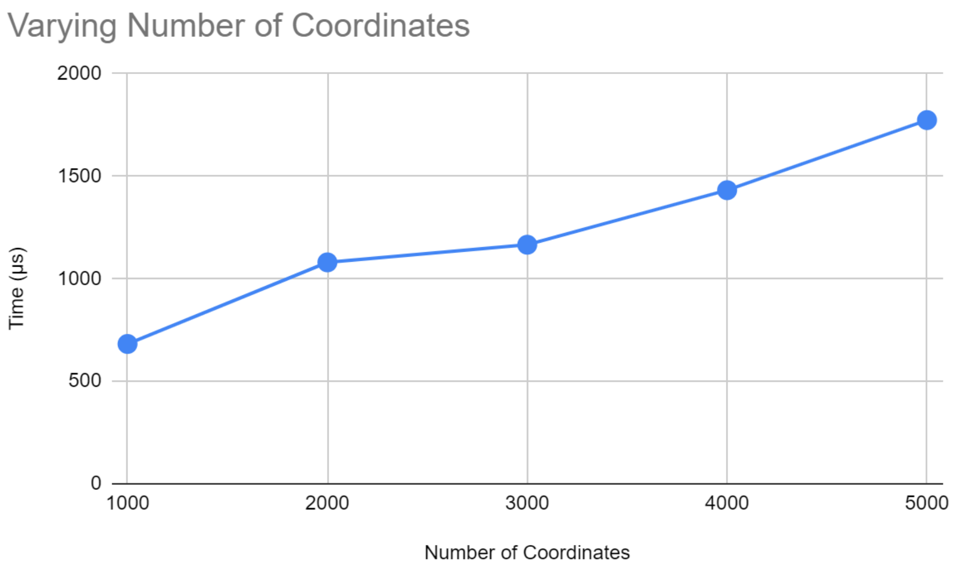
As specified in the problem statement, the analysis of the code was done in two ways:

1. **Varying the number of threads** between 1, 2, 4, 8 and 16 while keeping the number of destination points constant at 5000 points. The time taken by each iteration has been plotted in the graph below:



   As we can see, the time taken by the program decreases significantly as the number of threads increase. This is expected since increasing the number of threads splits up the code into multiple parallel blocks all of which run simultaneously thereby reducing the run time of the entire program.

2. **Varying the number of destination coordinates** between 1000, 2000, 3000, 4000, and 5000 while keeping the number of threads constant at 16. The time taken by each iteration has been plotted in the graph below:



As we can see, the time taken by the program increases slightly as the number of coordinates increases. This is expected since increasing the number of points means each thread needs to calculate more distances to find the minimum amongst each subset.