Traffic Light Simulation Report

ECM2433 Coursework

Design Choices

Assumptions Made

- Each time-period in which the lights change from red to green, renders the system safe for vehicles to pass through it in the next time-period.
- Within each time period either the traffic lights change state, or there is a potential for a car to arrive to the left queue, the right queue and for a vehicle to pass through the system.
- There are no factors which alter the capacity of the system to allow exactly one vehicle pass through the light system at any given time-period. For example, to pass through the system, three bikes would take the same amount of time as three cars.
- The lights are on an automated timer with no manual input, they will never deviate from the time-period specified for them to change.
- All Drivers in the queues are competent and their cars mechanically sound, if they are at the front of their queue and their light is green, they will be able to pass through successfully unhindered by stalls or breakdowns.
- All Drivers perfectly follow the Highway Code, for example no cars will try to skip through the red light even if there are visibly no cars in or approaching the opposite queue.

Design Choices

- A major design choice was to split the queue functionality from the simulator into its own files, this was to allow for greater reusability of the code produced, but also to help make the 'runOneSimulator' file more readable.
- Another major decision was to use the GNU Scientific Library's (GSL) random number generator instead of C's own rand() function. There were two reasons for this. The first being that the quality of random numbers produced would be much greater. The second was clearer when thinking about the program on a larger scale. In it's current state, rand() would have most likely provided sufficiently random numbers. However, if you were to increase the scope of the program from the current 100 iteration to 1000, 10 000 or more iterations of runOneSimulator, the higher quality of random numbers would soon become much more relevant.
- The GSL was seeded in runSimulators over runOneSimulation is due to the program running quickly enough that should it have been seeded for every individual simulation, the majority would return the same values. Therefore, a pointer to the GSL random number generator static variable is passed to the individual simulations for use.
- I decided to allow user input through the command line using scanf over allowing the program to be given arguments to allow for better error handling. If I had allowed user arguements to the program it would have introduced multiple points of failure, relating to the conversion of the string arguments to integer values. By using scanf, I was able to effectively ensure the correct typing was used for input as well as avoid the riskier string to int conversion.
- The reason for using both for and while loop in runOneSimulation also relate to scope. Primarily I believe it to have made the code much more readable by avoiding the addition of further nesting of if-else statements to determine the current state of the simulation, for example determining whether 500 initial iterations had passed. If desired, it also offers

scalability in terms of the number of iterations for which cars can arrive at either queue as it is easily increased by changing the parameters if the initial 'for' loop.

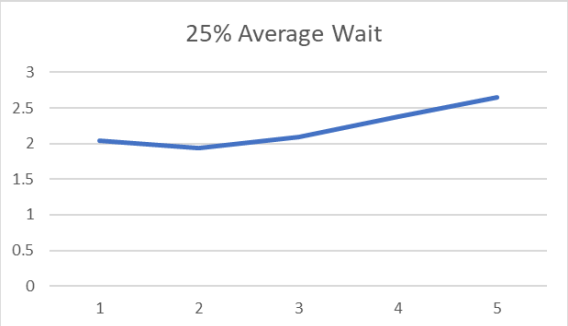
- Error Handling in the program is minimal due to its self-contained nature. Within the program I identified two areas in which error handling would be essential, namely the user inputs, and allocation of memory when Pushing to the queue. User inputs are checked to confirm they are of the correct type, however due to the scientific nature of the task, it has been assumed that the values input to program would be of acceptable scope to obtain the desired results from the program. Both User input error messages, and memory allocation errors in the queue are output to stderr.

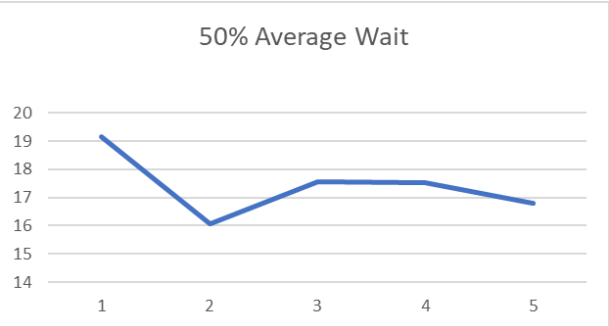
Experiment

With my simulator I decided to investigate the effects that increasing or decreasing the traffic light period would have. My initial hypothesis for this would was that there would be an ideal traffic light period in which cars could expect the lowest average wait time when compared to different traffic light periods.

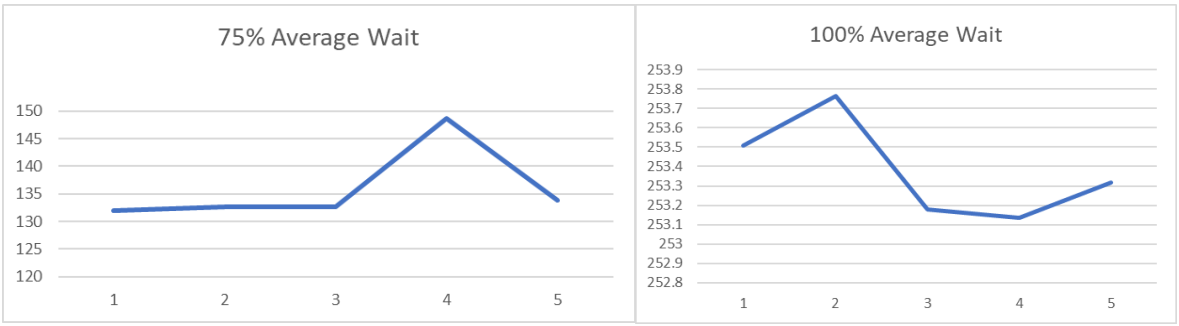
My methodology for conducting this experiment was quite simple. The program would be run multiple times where the arrival rate of vehicles would remain constant throughout and the traffic light period would be the only parameter to change with both right and left hand sides having equal parameters. I would repeat this three times each for the light periods of one through 5, and repeat the whole process four times at varying traffic arrival rates, namely 25%, 50%, 75% and 100%.

My findings demonstrated trends which both supported my initial hypothesis, whilst also working against it in some regards. To a minor extent the data demonstrated that there were indeed ideal lengths of traffic light periods in which traffic would flow better. In the two graphs below, we can see the average number of periods a car had to wait before being able to proceed at a 25% arrival rate and a 50% arrival rate respective.





A less expected, but quite unsurprising result from the data was tat as the load on the the system was increased, an ideal traffic light period is still somewhat present. The results at a 75% arrival rate seem somewhat anomalous not really demonstrating a trend. However, the 100% arrival rate demonstrates a trend upwards to a higher traffic light phase being more efficient. This can be seen in the graphs below.



Overall I would say that there is a relationship between the traffic light periods and the average wait experienced, however further investigation would be needed in order to verify this, and to determine how this discovery could be used. A useful insight could be gained from looking into varying the light period to benefit either left or right hand sides of the system depending on which sees more traffic. For example we can extrapolate the trend demonstrated above to hypothesise that if one side had an arrival rate of say 80% compared to the opposite side having 20%, then the side with greater traffic should have a longer light period compared to the side with less traffic.

Example Output

```
[dlk204@emps-ugcs2 Coursework]$ ./runSimulations
Enter 4 integers each seperated by a space, they should follow the format the follows:
Left-Arrival Rate, Left Light Period, Right-Arrival Rate, Right Light Phase.
45 5 45 5
Parameter Values:
    from left:
        traffic arrival rate: 45%
        traffic light period: 5
    from right:
        traffic arrival rate: 45%
        traffic light period: 5
Results (Averaged over 100 runs):
   from left:
       average vehicles passed:194.740
       average waiting time:   7.753
       average max wait time:  20.990
       average clearance time: 26.160
   from right:
       average vehicles passed:194.180
       average waiting time:   7.635
       average max wait time:  21.180
       average clearance time: 26.280
[dlk204@emps-ugcs2 Coursework]$
```

```
[dlk204@emps-ugcs2 Coursework]$ ./runSimulations
Enter 4 integers each seperated by a space, they should follow the format the follows:
Left-Arrival Rate, Left Light Period, Right-Arrival Rate, Right Light Phase.
100 5 100 5
Parameter Values:
    from left:
        traffic arrival rate: 100%
        traffic light period: 5
    from right:
        traffic arrival rate: 100%
        traffic light period: 5
Results (Averaged over 100 runs):
   from left:
       average vehicles passed:422.180
       average waiting time:   250.287
       average max wait time:  502.980
       average clearance time: 1008.990
   from right:
       average vehicles passed:422.180
       average waiting time:   256.347
       average max wait time:  509.040
       average clearance time: 1015.050
[dlk204@emps-ugcs2 Coursework]$ vi runSimulations.c
```