Traffic Light Simulator

132717

1 Assumptions

- The traffic light period on either side of the road must be greater than 1. If the period is 0, it would defeat the purpose of having a traffic light as the colour would never change. If the period is 1, the traffic light colours would continuously keep changing, not allowing any vehicles to go past as changing the traffic light colours takes 1 period.
- The starting traffic light colour is chosen at random.
- The results are shown to 2 decimal places.

2 Design Decisions

2.1 Queue

A queue was implemented to represent the queue of vehicles behind each traffic light. The queue was implemented using a linked list as it allocates and deallocates memory at run-time – making the queue more efficient than other data structures such as an array. Insertion and deletion (enqueue and dequeue) of this structure is simplified by using pointers that point to the next vehicle in the queue. An *enqueue* function is used to add a vehicle to the queue. A *dequeue* function is used to remove a vehicle from the queue, returning the time it arrived in the queue – this returned data is used in a calculation to find the average and maximum waiting time for the vehicles in the queue. A *getSize* function is used to check if the queue is empty so that the program knows not to dequeue a queue that has no vehicles in it.

2.2 Traffic Lights

A Boolean data type is represented by using *#define TRUE 1* and *#define FASLE 0*. A *typedef char BOOLEAN* was created to create Boolean variables *lGreen* and *rGreen* which represent the information of if the left traffic light or right traffic light is green.

2.3 Store Results as Struct

The results from the *runOneSimulation* function are stored in a struct, therefore the function returns a struct – with all the required information stored.

Using the struct allowed the return of multiple values from the function, which would otherwise not be possible. These values are used by the *runSimulations* program to get the average results over 100 iterations of a specific parameter input.

2.4 Randomness

The randomness introduced in this program is produced from the *getRandom-Choice* function which uses the GNU Scientific Library (GSL). The seed generation is based on the time of day so that the seed is different each time the function is called. This function produces and returns a random float between 0 and 1. This random float is compared to the probability of a vehicle being enqueued. If the *random float * 100* is less than the *arrival rate*, then a vehicle is enqueued.

2.5 Error Handling

Error handling is used to check the user input of the 4 parameters entered. The checks that are made are: only 4 parameters are entered, the values entered are all integers, the left and right vehicle arrive rate are between 0 and 100 – as the probability of a vehicle arriving cannot be greater than 100%, the left and right traffic light green period is greater than 1. Error handling is also used to trap the fact if memory was not allocated when using *malloc*. All the error messages are sent to stderr.

2.6 Memory Allocation

Malloc is used when creating queues as we do not know how much amount of memory is needed before compile time as the queue is built without a fixed upper size. The free method is used to dynamically de-allocate the memory to reduce wastage of memory by freeing it.

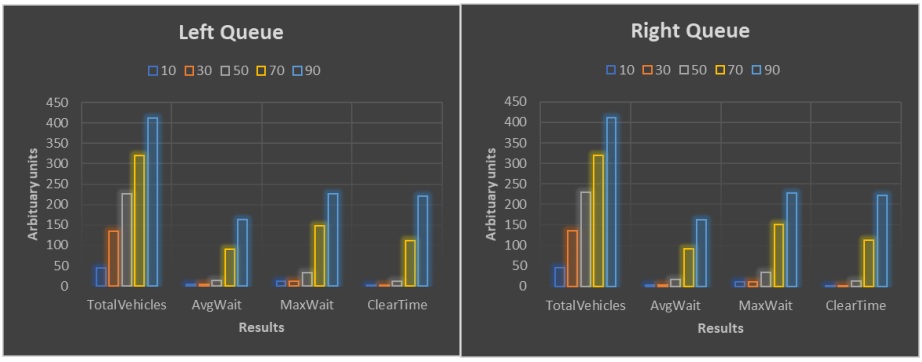
2.7 Command Line Inputs

The user enters the 4 parameters for the program in the command line, for example, *./runSimulations 50 50 10 10*. The parameter meanings are: left vehicle arrival rate, right vehicle arrival rate, left traffic light green period, right traffic light green period.

2.8 File Format

My files are split into 3 main sections: *runSimulations*, *runOneSimulation*, *queue*. Each of these files are split into .c and .h files to provide better organisation and clarity of the program. The header files declare functions and structures for the compiler.

3 Experiment



(a) Left queue (b) Right queue

Figure 1: A figure in which that the probability of vehicles arriving is the same in each direction. The legend shows the vehicle arrival probability for both queues. Both traffic light periods were constant, at 10.

Figure 1 shows that as the probability of a vehicle arriving increases, the total vehicles passed through the lights, the average wait time for a vehicle, the maximum wait time for a vehicle and the time taken for all the vehicles to go past the traffic lights once no more vehicles can join the queue also increases. The rate of increase of these variables as the probability of a vehicle arriving increases is the same for both the left and right queue. This is due to the probability rates being the same. For the time variables, this increase is at an exponential rate. This is due to the traffic build up caused by the traffic light period being the same even though there are more vehicles arriving. To help with this problem, as the probability of the vehicles arriving increases, the traffic light period also should increase so less time is wasted changing light colours.

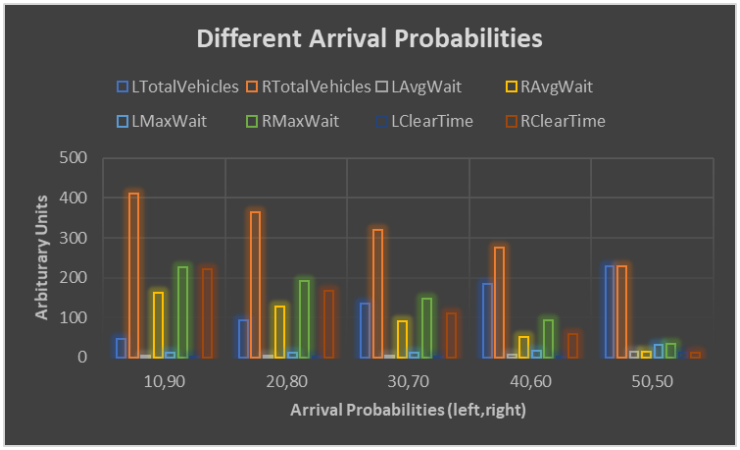


Figure 2: A figure in which the probability of vehicles arriving from one direction is higher than the other. The legend shows the data from the left queue 'L' and the right queue 'R'. Both traffic light periods were constant, at 10.

Figure 2 shows that the larger the difference is between the arrival probabilities coming from the left and right, the larger the difference is with the results from the left and right queue. This means that more time is wasted for the vehicles in the queue where the probability of a vehicle arriving is higher than the probability of a vehicle arriving from the other queue. This is due to the traffic light period being the same for both sides, even though more vehicles come from one side than the other. This causes a build up to happen on one side, and the traffic light being green with little to no passing vehicles on the other. This can be solved by making the light period higher for the queue with a arrival rate probability being higher than the other queue. The difference of the values between the queues decreases linearly as the gap between the arrival probabilities decreases. The difference in values converge towards zero as the difference in arrival rate probability's decrease.

```
ha468@emps-ugcs1:~/ecm2433/ca
[ha468@emps-ugcs1 ca]$ ./runSimulations 45 55 10 15
Parameter values:
        from left:
                traffic arrival rate: 45
                traffic light period: 10
        from right:
                traffic arrival rate: 55
                traffic light period: 15
Results (averaged over 100 runs):
        from left:
                number of vehicles:   209.98
                average waiting time: 41.07
                maximum waiting time: 76.62
                clearance time:       34.33
        from right:
                number of vehicles:   255.61
                average waiting time: 7.08
                maximum waiting time: 19.42
                clearance time:       4.68
```

Figure 3: Example output