ECM2433 The C Family Continuous Assessment

175809

1 Design Decisions

The entire project is written in ANSI standard C and uses the GNU Scientific Library for random numbers. This library was chosen to use this library since it is a simulation, requiring high-quality pseudo-random numbers. The program was structured with header files to lay out the function prototypes and structures. This was done to improve the structure and readability of the program itself by removing this. The Queue structure was separated from the main program because it contains functions only related to the structure, and the function like newNode is a local function, so the main program would not need access to it.

1.1 Queue

The queue uses a linked list that can only add nodes to the back of the list and remove nodes from the front of the list. This was done instead of an array because it would require moving the values in the queue upon de-queuing. The nodes in the linked list contain a pointer to the next node to allow for them to be linked. The node will point to NULL when it is at the back of the queue. This was done to allow for easy en-queuing of a node. The nodes contain an integer value that contains the vehicle's iteration entered the queue in the simulation. This is returned when the value is de-queued to calculate the vehicle's wait time. The queue structure contains the front and back nodes in the queue to allow for fast en-queuing and de-queuing of nodes. This helps alleviate one of the advantages of using an array over a linked list: slow retrieval of values in the linked list.

The function createQueue is used to allow for the queue structure to be allocated memory and setup the back and front nodes. The function newNode is a local function since it is only used when en-queuing within the queue, so it does not need to be accessed by anything outside the queue program. The function printQueue will recursively print out the queue from the parameter node to the back node. This was mainly used to test if the simulation functioned appropriately and the results were correct. The function emptyQueue is used to remove all nodes in the queue and then free up the queue to allow easy freeing up of allocated memory once the queue is no longer needed. The function isEmpty checks if both the back and front are NULL, and thus the queue is empty. This was created since several times, a queue needed to be checked if it was empty or not, like checking if the simulation was complete.

1.2 Running one Simulation

This function takes four input parameters, the right and left arrival chance (rate) and the right and left traffic light green period. The arrival rate was represented using chance since vehicles do not appear at constant rates. The traffic light period was only for the number of iterations when they were green since the left traffic light must be opposite in colour to the right one. This allows for control over the red period for each but it must be the same as the opposite green period. The function has another parameter, state. This structure contains all the information about the current state of the iteration, like the queues, current iteration and results. This was done to make the it easier to understand and make it easier to pass by reference as only state needs to be passed into the function.

The function will first check if it is the iteration to switch traffic light colour. Traffic light colour is represented as an integer (zero green on the left and one green on the right) to ensure that both lights cannot be green and red simultaneously. If it is the iteration to switch traffic lights, it recalculates when next to switch. It then increments the iteration and stops. This is done to reduce how many checks are needed and make it clear that no changes can be made to the queues in that iteration once traffic lights change. If the traffic lights do not change and iterations are below 100, it will en-queue a new node when the arrival chance is greater than the random number. It does this once for each queue, starting with the left, representing the zero or one vehicle joining the queue. The function gsl_rng_uniform was used for getting the random number since it creates a uniformly distributed in the range 0 to 1, allowing for an equal probability of any number being picked. The instance of the generator is stored in the state as r because it allows for the seed to function correctly and produce different random numbers each time. If the iterations are above 100, then it begins counting the number of iterations it takes to clear each queue. Removes vehicle from queue depending on which light is green. Upon leaving the queue, the wait time is calculated for the vehicle using the value assigned in the queue. A 50 per cent chance of removing a vehicle from the queue was chosen since zero, or one vehicle had to be removed, which was a middle ground. Nothing is returned since state was passed by reference, so it has been changed, and the input parameters do not change.

1.3 Running the Simulation

This starts by creating the queues and setting up the instance of the random generator. Each parameter is inputted using scanf. Scanf was used instead of argc and *argv to allow for requesting another input if an invalid one was entered, and scanf runs in run-time. If scanf is not one (valid), then it produces an error in both stdout and stderr and then exits. This is done since inputting an invalid type leaves it in the buffer creating an infinite loop. Then it displays the parameters entered before looping the function runOneSimulation until both queues are empty and the iteration is greater than 100. Then the average wait times are calculated. In the division, both values are cast to double to ensure the result will be a double type. Then the results are displayed before freeing up all allocated memory and exiting successfully. Memory is freed at the end to ensure there are no memory leaks.

1.4 Example Output

```
Please enter the left arrival rate (0.0 to 1.0): 0.1
Please enter the right arrival rate (0.0 to 1.0): 0.1
Please enter the left traffic light green period (1 to 100): 10
Please enter the right traffic light green period (1 to 100): 90

Parameter values:
    from left:
        traffic arrival rate:  0.10
        traffic light period:  010
    from right:
        traffic arrival rate:  0.10
        traffic light period:  090

Results (averaged over 100 runs):
    from left:
        number of vehicles:     007
        average waiting time:   105.29
        maximum waiting time:   150
        clearance time:         103
    from right:
        number of vehicles:     010
        average waiting time:   0.70
        maximum waiting time:   003
        clearance time:         000
```

Image one: This is an example output produced for the experiment of low traffic levels with a left period of 10 iterations and a right period of 90 iterations. It displays the parameters entered and the results as specified.

2 Experiments

Two different experiments were conducted. The first testing how a change in period and arrival rate impacted average waiting time, with both queues having identical values. The second tested the same, but each queue would have differing periods to investigate how that impacted average wait time.

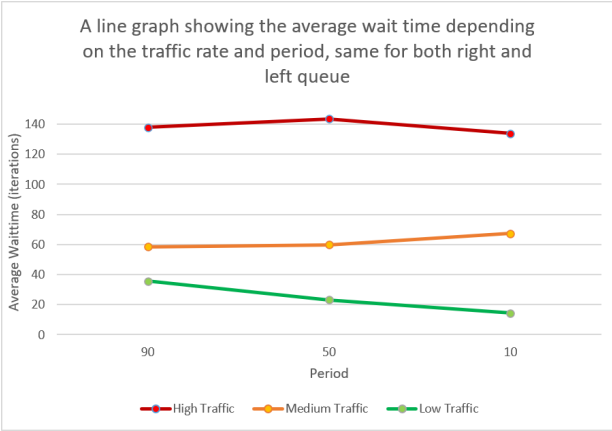
2.1 Experiment One - Same period

High traffic Area (Arrival Rate of 0.9)												
Period	Average Waiting Time										Average	Total Average
Experiment	1		2		3		4		5			
Queue	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right		
High (90)	141.47	175.43	92.40	145.04	106.94	168.26	117.18	163.02	119.22	148.80	137.78	138.43
Middle (50)	126.28	188.12	122.64	166.46	97.68	165.79	119.42	212.93	92.72	144.49	143.65	
Low (10)	109.04	129.23	126.71	166.22	112.08	139.23	166.90	146.18	101.54	141.33	133.85	

Moderate traffic Area (Arrival Rate of 0.5)												
Period	Average Waiting Time										Average	Total Average
Experiment	1		2		3		4		5			
Queue	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right		
High (90)	32.00	87.35	11.81	117.83	16.96	79.46	22.77	100.28	17.46	99.12	58.50	61.82
Middle (50)	31.81	84.15	61.93	85.16	30.71	103.21	29.32	75.17	24.90	71.10	59.75	
Low (10)	88.34	67.81	83.23	65.39	47.70	55.62	53.43	72.10	96.81	41.65	67.21	

Low traffic Area (Arrival Rate of 0.1)												
Period	Average Waiting Time										Average	Total Average
Experiment	1		2		3		4		5			
Queue	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right		
High (90)	0.92	43.90	28.77	52.50	33.38	72.42	8.08	48.82	16.43	51.17	35.64	24.39
Middle (50)	20.33	11.69	7.33	4.83	25.61	88.44	18.00	20.80	12.83	21.25	23.11	
Low (10)	14.33	13.25	14.58	17.40	9.00	25.10	8.18	5.62	17.67	19.18	14.43	

Table one: This shows how changing the period impacts the average wait time in a high traffic area (arrival rate of 0.9).
Table two: This shows how changing the period impacts the average wait time in a moderate traffic area (arrival rate of 0.5).
Table three: This shows how changing the period impacts the average wait time in a low traffic area (arrival rate of 0.1).



Graph one: This line graph shows how changing the period and arrival rates results in changes in average wait time.

A clear trend is that increasing the arrival rate increases the waiting time, with the total average dramatically increasing from 24.39 to 138.43 iterations with an increase in the arrival rate of 0.8. The graph shows that the traffic light period has a small but noticeable impact on wait time; the lower the period, the lower the average wait time. The low period is the best on average with 71.83 iterations but is only 5.48 iterations away from the worst, the high period of 77.31 iterations.

2.2 Experiment Two - Different period

High traffic Area (Arrival Rate of 0.9)

Period		Average Waiting Time										Average	Total Average
Experiment		1		2		3		4		5			
Queue Left	Queue Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right		
High (90)	High (90)	141.47	175.43	92.40	145.04	106.94	168.26	117.18	163.02	119.22	148.80	137.78	318.04
High (90)	Low (10)	42.29	908.10	50.16	1042.66	34.26	961.41	37.79	1109.92	42.77	1023.28	525.26	
Low (10)	High (90)	837.42	47.19	870.15	55.77	914.69	57.84	925.89	67.49	916.05	60.27	475.28	
Low (10)	Low (10)	109.04	129.23	126.71	166.22	112.08	139.23	166.90	146.18	101.54	141.33	133.85	

Moderate traffic Area (Arrival Rate of 0.5)

Period		Average Waiting Time										Average	Total Average
Experiment		1		2		3		4		5			
Queue Left	Queue Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right		
High (90)	High (90)	32.00	87.35	11.81	117.83	16.96	79.46	22.77	100.28	17.46	99.12	58.50	165.81
High (90)	Low (10)	2.87	532.72	11.11	619.48	4.58	491.00	8.14	670.94	5.75	411.80	275.84	
Low (10)	High (90)	506.63	15.50	445.69	16.16	382.00	4.70	624.56	5.62	603.31	12.66	261.68	
Low (10)	Low (10)	88.34	67.81	83.23	65.39	47.70	55.62	53.43	72.10	96.81	41.65	67.21	

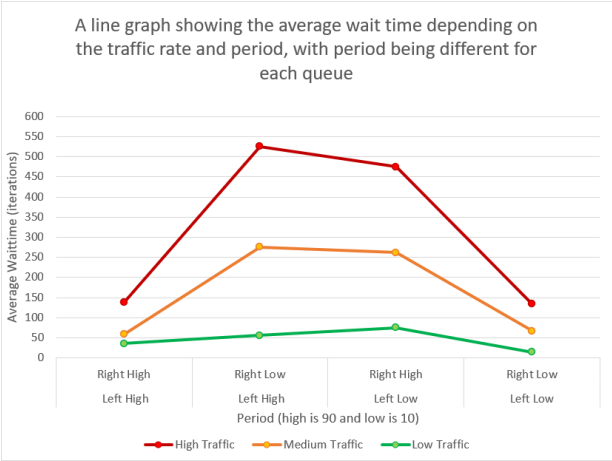
Low traffic Area (Arrival Rate of 0.1)

Period		Average Waiting Time										Average	Total Average
Experiment		1		2		3		4		5			
Queue Left	Queue Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right		
High (90)	High (90)	0.92	43.90	28.77	52.50	33.38	72.42	8.08	48.82	16.43	51.17	35.64	45.28
High (90)	Low (10)	1.71	241.33	1.33	64.78	1.75	65.50	1.50	88.75	1.18	94.25	56.21	
Low (10)	High (90)	201.87	0.67	147.80	0.36	79.00	1.11	209.81	1.64	105.29	0.70	74.83	
Low (10)	Low (10)	14.33	13.25	14.58	17.40	9.00	25.10	8.18	5.62	17.67	19.18	14.43	

Table four: This shows how changing the period of each queue impacts the average wait time in a high traffic area (arrival rate of 0.9).

Table five: This shows how changing the period of each queue impacts the average wait time in a moderate traffic area (arrival rate of 0.5).

Table six: This shows how changing the period of each queue impacts the average wait time in a low traffic area (arrival rate of 0.1).



Graph two: This line graph shows the average wait time changing due to the period of each queue and arrival rate.

This experiment shows that having different periods results in degraded performance of the traffic lights, leading to higher waiting times. On average, the waiting times for the two tests with different periods were 203.62 iterations higher than those with the same period. The graph shows that the impact of higher traffic leads to further increased wait time in the tests with different periods. This implies it has decreased the traffic lights' maximum throughput. Another trend shown is that if a queue has a higher period than the other queue, it will have reduced wait time depending on the difference in the period. This could be useful to reduce traffic buildup on a queue with a higher arrival rate but would need to be explored further in another experiment. In both experiments, the left queue performs better than the right queue, especially when the period is higher for the left queue. This is likely due to the traffic lights starting on the left resulting in no queue building up initially. To improve the simulation, the starting traffic light should be random.