Title:

An Introduction to Simulation and Modeling

HW2

CS543

Author:

Adib Rastegarnia

arastega@purdue.edu

Fall 2014

1. Introduction:

In this study, at first a series of N M/M/1 queue is simulated. In this part the effect of N on simulation time is investigated and the results are shown.

Second, an M/M/1 queue with two different kinds of customers (High and Low priority) is simulated. In this study only preempt scenario is investigated.

Finally, application of Genetic algorithm (GA) for solving problems is investigated. In this part, GA solution for the card pilling problem is analyzed and described. In addition, the program is executed and the results are reported.

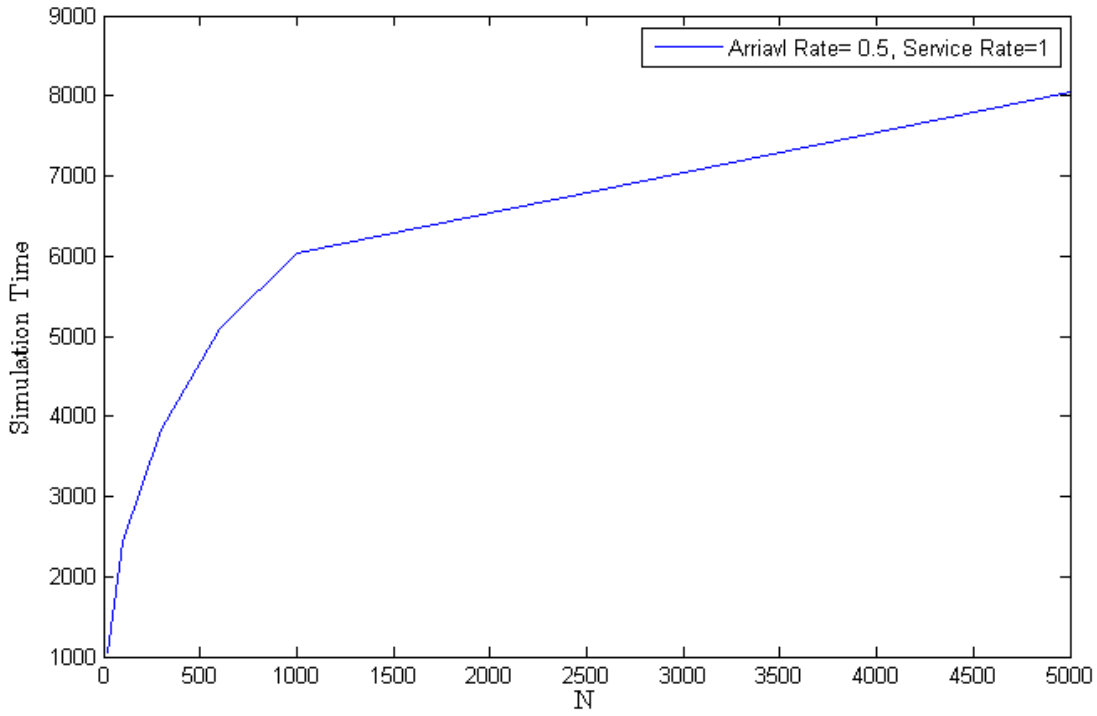
2. Part1:

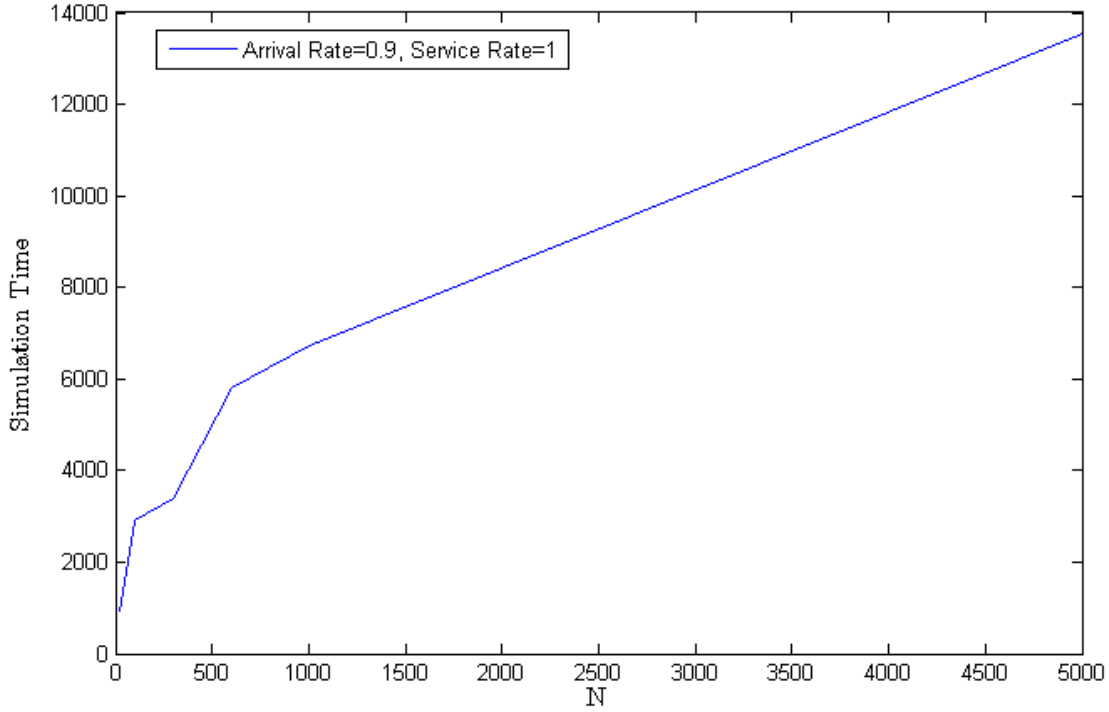
For this part a small modification on the source code of first assignment (single server queue) is made in order to implement a tandem queue. Simulation parameters are listed in Table I.

Table I. Simulation parameters for Part 1

Simulation Parameter	Value
Arrival rates	0.5,0.9
Mean service time	1
Number of customers	2000

The effect of N on the simulation time for different arrival rates are shown in Figures 1 and 2. As we can see in the Figures 1 and 2, as long as the N is increases, simulation time increases as well. This matter originated from this fact that, when N increases, customers should stay in the system for a longer period of time.


Arriavl Rate= 0.5, Service Rate=1
Simulation Time
N
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000


Arrival Rate=0.9, Service Rate=1
Simulation Time
N
14000
12000
10000
8000
6000
4000
2000
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000

3. Part2:

A. Preempt/Resume case: some small modifications is applied to the M/M/1 implementation from the first homework and then results is generated based on different values of λ_1 and λ_2.

Simulation Parameter	Value
λ_1 (High priority)	0.2,0.5
λ_2 (low Priority)	0.05,0.1,0.2,0.3,0.4,0.5
Mean service time	1
Number of customers	2000

From simulation results that are shown in Figures 3 to 8 we understand the when λ_1 is fixed and λ_2 is increased, average number of high priority customers is increased as well. In addition, the simulation results show that when average number of high priority customers is increased, average number of low priority customers will be decreased (Although in my results there are some fluctuations)

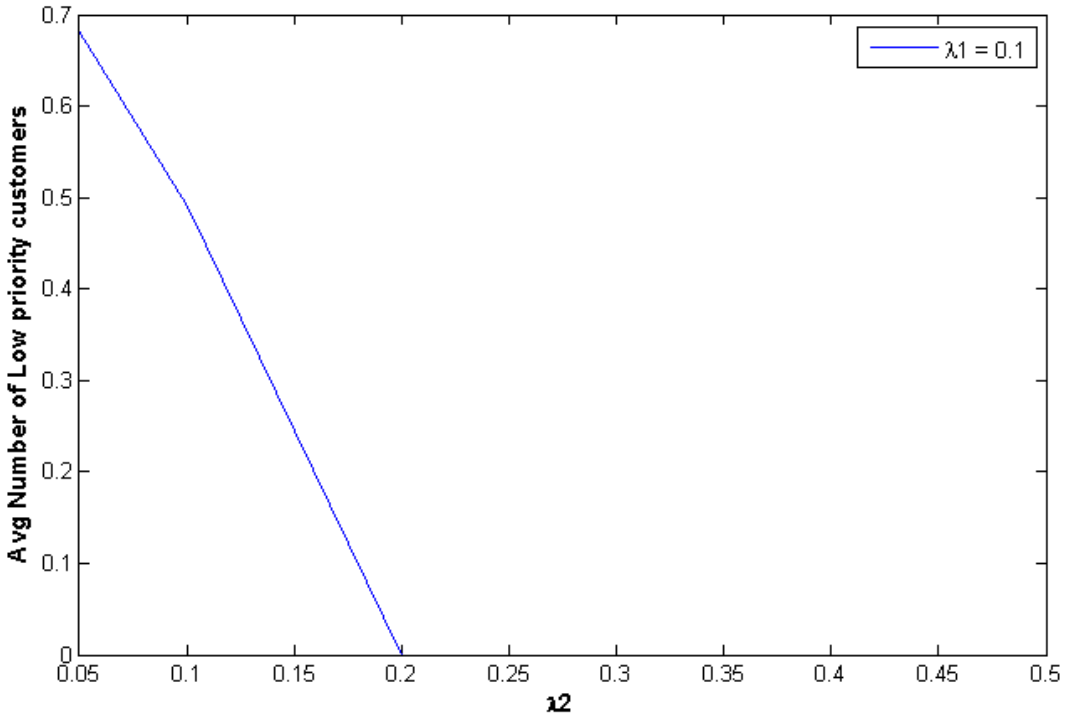


Fig 3. Average Number of Low priority customers in Queue vs. different values of λ_2

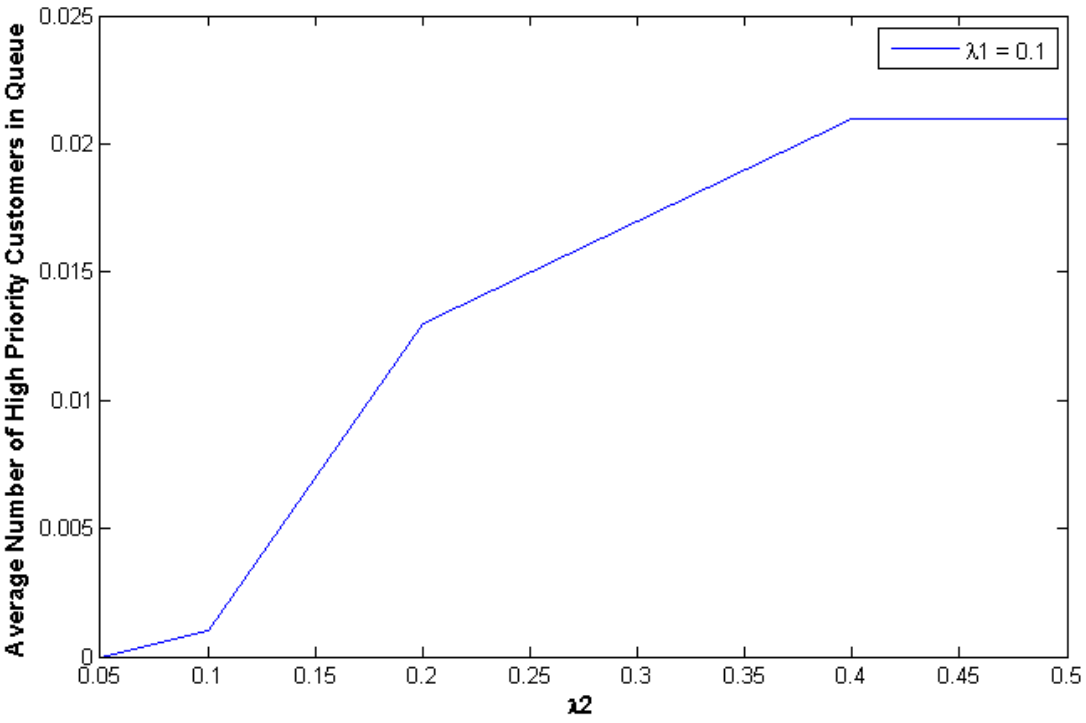


Fig 4. Average Number of high priority customers in Queue vs. different values of λ_2

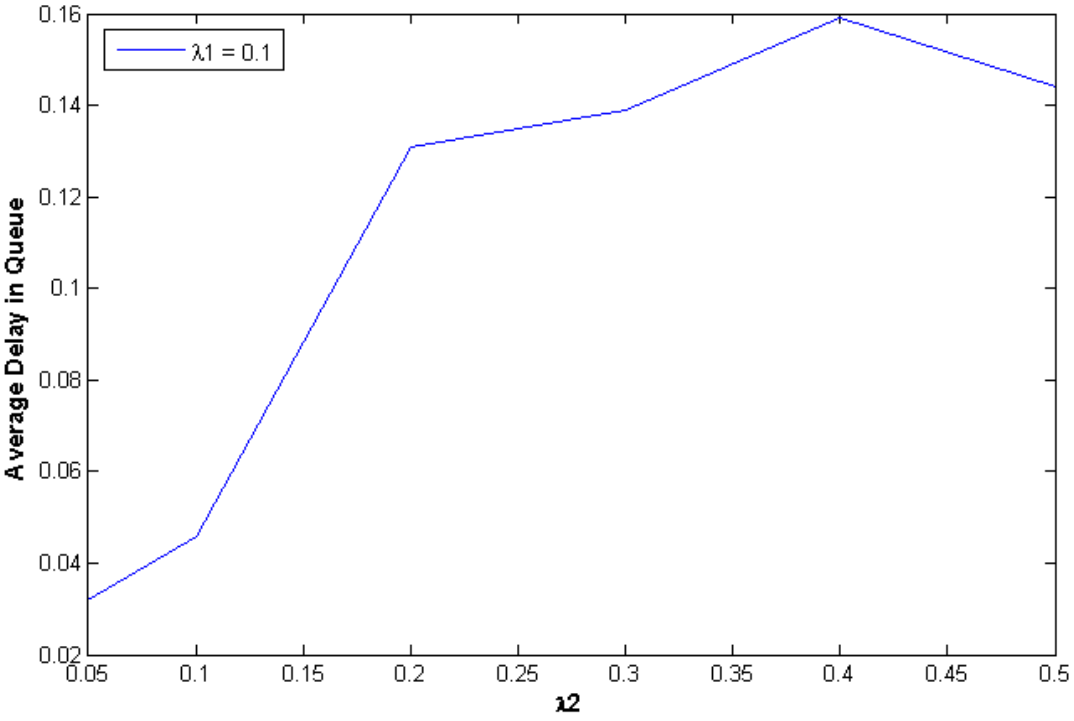


Fig 5. Average delay in Queue vs. different values of λ_2

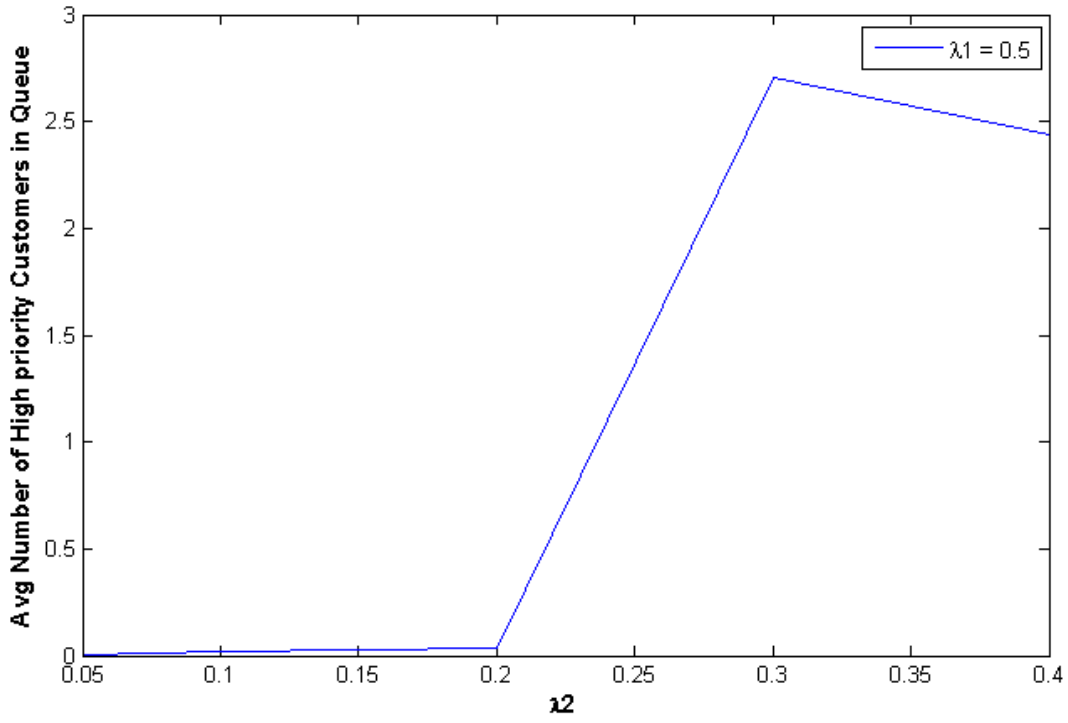


Fig 6. Average number of high priority customers vs. different values of λ_2

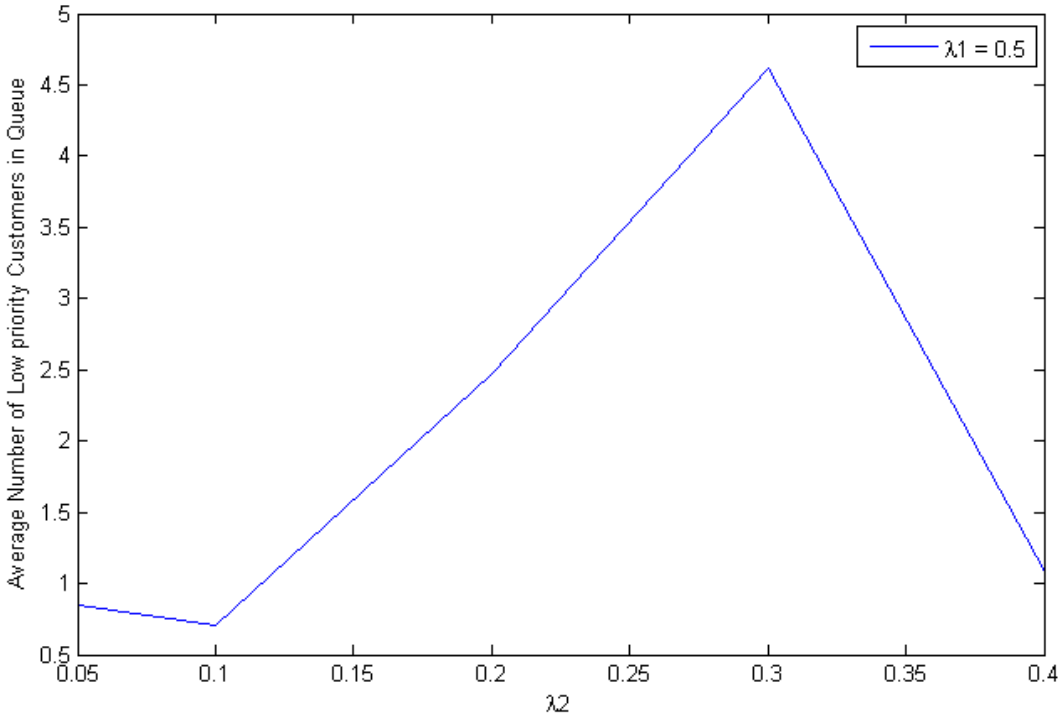


Fig 7. Average number of low priority customers vs. different values of λ_2

1

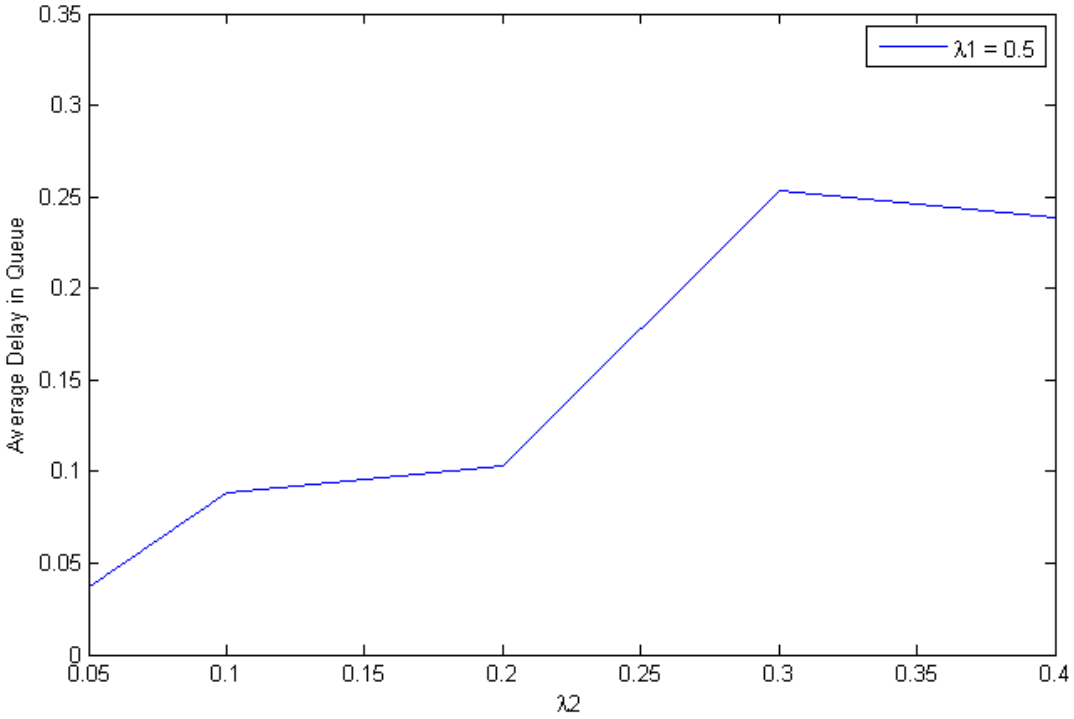


Fig 8. Average delay in Queue vs. different values of λ_2

4. Part3:

For this part card pilling is selected.

Problem definition:

We have 10 cards numbered 1 to 10. We wanted to divide them into two piles. Sum of the first pile should be 36 and product of all in the second pile should be 360. Solving this problem by hand is a time consuming job and maybe you makes mistake. Consequently, Genetic algorithm is used in order to solve it.

Parameters definition for proposed GA :

In this problem an array with size of 30 * 10 is defined for saving each of the chromosomes. In other words, initial population has 30 chromosomes and each of the chromosomes is represented by a 10 bit array. In addition, each of the cards is represented by one bit (gene) in 10 bits array. In this problem bit 0 means the card which has this value belongs to the first pile (Sum of them should be 36) and bit 1 means the card belongs to the second pile. The bits in each of the chromosomes are initialized to 0 and 1 in a random way.

GA Algorithm:

1- **Selection:** In this step, two chromosomes are selected in a random way from initial population in order to generate next generation.
2- **Evaluation:** In this step, based on the evaluation function which is defined as follows, two selected chromosomes are evaluated and compared with each other. The main goal of this step is to determine which chromosome is better than another one. In other words, after evaluation a chromosome is winner and another chromosome is loser. Evaluation function computes sum and products of the bits which belongs to first and second piles respectively. And then scaled sum and product errors are calculated. These values are summed together (combined error) and the final value returned for evaluation. The chromosome with lower error is selected as the winner chromosome.

//// Evaluation function

```
if (gene[n,i] == 0)
{
sum += (1 + i);
}
//if the gene value is 1, then put it in
//the product (pile 1), and calculate sum
else
{
prod *= (1 + i);
}
```

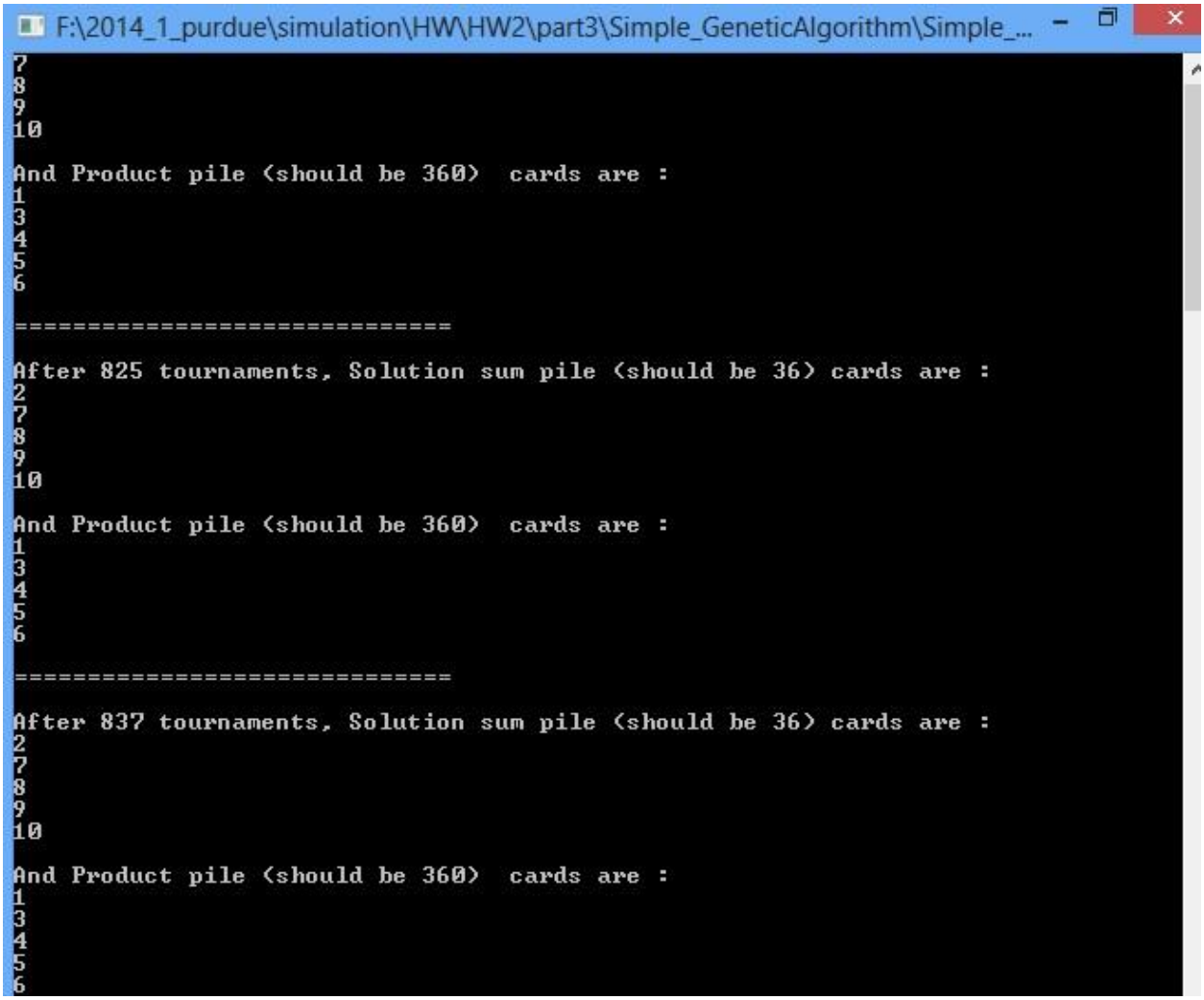
```
scaled_sum_error = (sum - SUMTARG) / SUMTARG;
            scaled_prod_error = (prod - PRODTARG) / PRODTARG;
            combined_error = Math.Abs(scaled_sum_error) +
                             Math.Abs(scaled_prod_error);
```

3- In the third step, loser chromosome should be improved by using recombination and mutation operators. In this step, mutation and recombination operations are selected based on a predefined rate. Recombination and mutation operation for this problem is described as follows:
    - Recombination: For each of the genes of loser chromosome, a random number is generated and compare with recombination rate and then if its value is less than recombination rate, then each of the bits in loser chromosomes are replaced with winner bits.
    - Mutation: In this step the same as previous step, for each of the genes of loser chromosome a random number is generate and compare with mutation rate and if its value is less than mutation rate, then bit 0 is changed to 1 and vice versa.

- After recombination and mutation for each of the bits of loser chromosome, evaluation function is called and if the combined error is zero then one of the solutions will be printed.

4- The above procedure is repeated for 1000 generations

After execution of the program, I understood the GA algorithm is converged sooner than 1000 generations. A sample of results are illustrated as follows:



5. Conclusion:

In this study, a tandem queue, a priority M/M/1 queue and a Genetic algorithm case study is investigated and the produced results are reported.