Simulation of a Four-Car Elevator Operation Using MATLAB

Saw Soon King & Omrane Bouketir
School of Electrical and Electronic Engineering
University of Nottingham, Malaysia Campus
Jalan Broga 43500 Semenyih Selangor, MALAYSIA
Tel: 60-3-8924-8159   E-mail: omrane@nottingham.edu.my

Abstract
The design and simulation of a four-cars-elevator controller in a nine storey building is described in this paper. The design and simulation were accomplished using MATLAB™ fuzzy logic toolbox. The logic of the controller of a multi-car elevator has to be designed in such a way that the average waiting time is minimized while keeping the energy consumption of the system optimum. This is a multi-criteria optimization problem in stochastic environment and is best approached through Artificial Intelligent techniques. The work here focuses mainly on extracting the rules to minimize factors (i.e. waiting time, travelled distance and riding time) in order to minimize the energy consumed by the system. In this paper a detailed algorithm is presented to achieve the multiple objectives of minimizing the waiting time and the distance travelled simultaneously. This was accomplished by distributing different weightage to different quantities and then minimizing a combined cost. A simulator has been built with interactive GUI in Matlab to evaluate the efficacy of the algorithm.

Keywords: Elevator, Fuzzy Logic, Membership Function, Optimization, Simulator

1. Introduction
Elevators are often a requirement in new buildings with multiple floors throughout the world. In high rise buildings, there is a need for intelligent elevator control system in order to obtain optimum performance. Usually a high rise building contains more than one elevator to increase the optimization of the elevator performance. When a few hall requests are registered, the elevator controller will decide which of the elevator will serve which of the incoming hall requests. Therefore the controller plays an important role in determining the time for which the passengers have to wait before the elevator arrives (waiting time). The waiting times of the passenger for an elevator to arrive or to serve their request are often heard as complaints by the passengers. Consequently the controller will reduce the waiting time of the passenger. However, in the process of reducing the waiting time, other quantities like the time spent by the people in the elevator (riding time) and the total distance travelled by the elevator increases, which is also unwanted. Therefore an intelligent elevator control system is required to solve this problem of optimization of multiple objectives. The multiple objectives in the current controller design are minimization of waiting time and distance travelled simultaneously.

However, an optimum solution to this problem is not known yet (Crites and Barto, 1998), or perhaps not possible due to its stochastic nature. In (Crites and Barto, 1998) Reinforcement Learning (RL) algorithms have been demonstrated to be powerful heuristic methods for addressing large-scale control problems and have been applied to elevator group control. A team of RL agents, each of which is responsible for controlling one elevator car was used. The team receives a global reward signal which appears noisy to each agent due to the effects of the actions of the other agents, the random nature of the arrivals and the incomplete observation of the state. The results demonstrated the power of multi-agent RL on a very large scale stochastic dynamic optimization problem of practical utility. Other works in this area based on fuzzy logic can be found in (Tan et al., 1997), (Kim et al., 1998), (Marja-Liasa, 1998), (Wang et al., 1999), (Koehler & Ottiger, 2002) and (Tanuj, 2004).

In this paper a detailed algorithm is presented to achieve the multiple objectives of minimizing the waiting time and the distance travelled simultaneously. This was accomplished by distributing different weightage to different quantities and then minimizing a combined cost. A simulator has been built with interactive GUI in Matlab to evaluate the effectiveness of the algorithm.

2. Elevator Configuration
The number of the elevators was fixed to stress more on the group control of elevators rather than the selection of
optimum number of elevators. Most of the features are easy to understand. Thus, some basic features of the elevators have been adopted as follows:

- The building has 9 floors (1-9), 4 elevators which are independent of each other operationally but controlled by a common supervisory group controller based on fuzzy logic.
- The elevator travels in a fixed constant speed which is 1.5 meters per second. The inter-floor distance between each floor is selected to be 3 meters, so the elevator will only take 2 seconds to move from one floor to another. For each stop, the elevator will take 6 seconds to allow passengers to board or to alight the car.

3. Traffic Generation

The traffic generation for the hall requests is produced and generated in such a way that it mimics real life situations. It is well known that the real world stochastic processes like arrival of users for a service follow a Poisson distribution. For this case, the stochastic process is the arrival of passengers for the elevator services.

The probability that there are exactly $x$ occurrences that take place during a fixed amount of time-interval is given by:

$$P(x, \lambda) = \frac{e^{-\lambda} \lambda^x}{x!}$$  \hspace{1cm} (1).

The traffic generation for this elevator system is divided into 5 different modes:

1. busy mode, 2. moderate mode, 3. up-peak mode, 4. down-peak mode, and 5. off-peak mode. The traffic for each mode is created using the random integer generator (“randint”) in MATLAB. In up-peak mode, all the hall requests are registered at the ground floor while in down-peak mode, all the car request’s destination is to the ground floor. On the other hand, for other traffic modes, their hall requests are generated randomly from the ground floor to the highest floor. Table 1 describes specifications of the 5 different modes and their respective arrival rates.

3.1 Input and Output Parameters

In order to achieve good performance, the elevator fuzzy logic controller uses 3 parameters as the control input and one parameter for output. These parameters represent the criteria or objective to be optimized in this elevator system. The input variables for the elevator system are:

- **Waiting time:** is the total time that the passenger has to wait before an elevator serves his/her request.
- **Travelling distance:** is the total distance travelled by an elevator before reaching a request from present state. Minimizing the distance means minimizing the energy consumed by the system.
- **Loading:** is the number of passengers in an elevator. This is to have an idea about the load of each elevator so the controller can decide which car may serve a request.

Whereas the output variable is:

- **Priority:** Output of the fuzzy logic controller, where the elevator (among the four) with the highest value of priority will be assigned.

3.2 Estimation of the Range for Input Variables

The range of the inputs for the elevator system have been estimated and found to be:

- Maximum waiting time is when the request is from the 9th floor and there is a request in every floor. The expected waiting time is expressed in terms of number of floors and number of stops since the constant speed of the elevator is 1.5 meter per second which is in agreement to existing systems (Tanuj, 2004) while the height of each floor is 3 meter (inter-floor distance = 3 meters). Therefore the elevator takes only two second to travel from one floor to another. For each elevator stops, it takes approximately 6 seconds to unload or to load passengers. So the expression for calculating the waiting time is:

$$ewt=6*ns + 2*nf;$$

where:

- $ewt$: expected waiting time
- $ns$: number of stops
- $nf$: number of floors between car position and the request.

- Maximum distance required to be travelled if from the ground floor to the ninth floor is 21 meters.

The expected distance is equivalent to the number of floors that a particular elevator has to travel to reach the request under consideration and assuming that there are no new requests assigned to this elevator during the duration for the elevator to travel to the request. The maximum value which is 24 occurs when the hall request is at the 9th floor while the elevator is situated at the ground floor (1st floor). So the expression for calculating the distance is:
$ed = 3 \times nf$  where:

$nf$: number of floors between car position and the request.

4. Membership Functions and Fuzzy Sets

The membership function that is used for each quantity for the input variables is triangle function. For each quantity, a total of five membership functions were used. As for the output variable, three triangle membership functions and two trapezoids function are used. Figure 1 shows the membership function for waiting time input.

There are 4 different types of fuzzy rule sets constructed with each to minimize a respective parameter. One of the fuzzy rules set minimizes waiting time, other set minimizes the distance travelled by the elevators, third set minimizes the loading and the last set minimizes all of them simultaneously. A sample of the first set is given in the passage below.

![Rule Set 1](image1)

There are only 5 rules used and all of the rules have the same weightage which is one.

In the last set there are altogether 15 IF/THEN rules. All rules are described in one dimensional space for each input and output as shown below.

![Rule Set 2](image2)

The number appearing at the end of each rule indicates the rule’s weight. Every rule has a weight which ranges from 0 to 1. Rules with higher weight will have a higher priority. The input waiting time is of the highest priority since the rule weight for it is 1.0 which is higher as compared to the other input variables. Based on the rules above, the fuzzy logic-based elevator will optimize the minimization of waiting time before considering the other quantities.

The rules above try to minimize all objectives simultaneously with the input variable of waiting time given the highest priority followed by loading and distance according to the different weightage assigned to them.

The fuzzy output which is priority indicates the suitability of a particular elevator to be assigned to a particular hall request. Depending on the fuzzy inputs and the rule bases, the output fuzzy set, ‘priority’ is computed using Mamdani inference scheme.

5. Rules Implementation in MATLAB

Here extracts from the overall source code developed in MATLAB are given.
5.1 Traffic Generation

```c
5.1 Traffic Generation

Traffic Generation:

```c
76 \ To choose the traffic mode
77 traffic_mode = input([1 for busy(peak), 2 for moderate and 3 for off-peak])
78 while (traffic_mode > 3) & (traffic < 1) >> will ask user to input again
79 traffic_mode = input([1 for busy(peak), 2 for moderate and 3 for off-peak])
80 end
81 nl = 0;
82 for c = 1:1:300
83 if c > nl
84     if traffic_mode == 1
85         interval = randint(1,1,[1,4,2]) % 1 input in 4-5 seconds, 30-70 inputs in 5 mins
86     elseif traffic_mode == 2
87         interval = randint(1,1,[6,20]) % 1 input in 6-20 seconds, 15-50 inputs in 5 mins
88     elseif traffic_mode == 3
89         interval = randint(1,1,[20,60]) % 1 input in 20-60 seconds, 8-15 inputs in 5 mins
90     end
91     ct = randint(1,1,[1:4:1, interval+n1]) % “generates an output of 1 x 1 integer matrix and the output is
92         % limited by the range given in the square brackets”
93     nl = nl + interval
94 end
95 if c == ct
96
97 ball_requests(1,c) = randint(1,1,[1,2])
```

5.2 Calculation of Waiting Time and Travelled Distance

```c
5.2 Calculation of Waiting Time and Travelled Distance

```c
101 elseif status(1,c) == 1 & direction_of_ball_requests(1,c) == 2 & move_car_position(1,d) > max_carcall(1,d) % ball request is above the highest car request for an elevator
102
103 if ball_requests(1,c) > max_carcall(1,d) % ball request is above the highest car request for an elevator
104
105 waiting_time(1,d) = abs(ball_requests(1,c) - car_position(1,d))
106 distance(1,d) = 1^abs(ball_requests(1,c) - car_position(1,d))
107 m = centroid('Fuzzy_rules'); % To read a fuzzy inference system from Fuzzy_rules.txt file
108 priorities(1,d) = evalfis([waiting_time(1,d) distance(1,d) loading(1,d),a]
109 else if ball_requests(1,c) > min_carcall(1,d) & ball_requests(1,c) < max_carcall(1,d)
110 % ball request is between the lowest and the highest car request
111
112 waiting_time(1,d) = abs(ball_requests(1,c) - car_position(1,d)) + [3*no_of_car_calls(1,d)]
113 distance(1,d) = 1^abs(ball_requests(1,c) - car_position(1,d))
114 m = centroid('Fuzzy_rules');
115
116 elseif ball_requests(1,c) < min_carcall(1,d) % ball request is below the lowest car request
117
118 waiting_time(1,d) = abs(ball_requests(1,c) - car_position(1,d)) + [3*no_of_car_calls(1,d)]
119 no_of_car_calls = 0;
120
121 distance(1,d) = 1^abs(ball_requests(1,c) - car_position(1,d))
122 m = centroid('Fuzzy_rules');
123
124 priorities(1,d) = evalfis([waiting_time(1,d) distance(1,d) loading(1,d),a] end
```

103
5.3 Moving Status Assignment

Definitions of some variables used in the source code are as follows:

**car assignment;** it is an array of dimension 1 x 4. At the start, all the 4 values are set to zero since there is no hall request being registered. Once a hall request is assigned to an elevator, for instance if elevator 1 is being assigned to a hall request, car_assignment(1,1) will be set to 1 which means the first row and the first column of the array car_assignment will be set to 1.

```
car_assignment =
1 0 0 0
```

**car position (move_car_position);** it is also an array of dimension 1 x 4 and its value is only from 1 to 9 which represents the floors in the building. In the example below, elevator 1 is located at 1st floor, elevator 2 is positioned at the 4th floor, elevator 3 is situated at the 7th floor and lastly elevator 4 is placed at the 9th floor.

```
car_position =
1 4 7 9
```

**hall_request;** represents the hall requests that are active or registered. Similar to the car position, the values varies from 1-9. The array’s column size is determined by the number of hall requests served by a particular elevator. If there are 3 hall requests assigned to an elevator, then the array size would be 3 x 4. Example below shows that the 3 hall requests for elevator 1 are at the 2nd floor, 4th floor and 7th floor.

```
hall_request =
2 0 0 0
4 0 0 0
7 0 0 0
```

**direction_of_hall_request;** this variable represents the direction of the hall requests that are registered or active. Similar to the hall request, it has the same array size as the hall request. An up button is represented by ‘1’ while a down button is represented by ‘2’. In the example below a 3 hall requests with the desire of moving upwards are registered for elevator 1 and a hall request moving downwards is active for elevator 2.
direction_of_hall_request =

\[
\begin{array}{cccc}
1 & 2 & 0 & 0 \\
1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 \\
\end{array}
\]

**status:** also of dimension 1 x 4 and it represents the status of each elevator and it could be either 0, 1 or -1 for each elevator corresponding to idle, up and down respectively. Value below shows that elevator 1 is moving up, elevator 2 moving down while elevator 3 and 4 are idle.

\[
\text{status} = \\
\begin{array}{cccc}
1 & -1 & 0 & 0 \\
\end{array}
\]

**loading:** represents the number of people inside an elevator and it has an array size of 1 x 4. The maximum capacity of an elevator is 23 persons (1500Kg). The loading value below shows that there are currently 20, 4 and 13 people in elevator 1, 2 and 3 respectively.

\[
\text{loading} = \\
\begin{array}{cccc}
20 & 4 & 13 & 0 \\
\end{array}
\]

**moving_status:** it is of dimension 1 x 4 and it represents whether there are hall request or car request that are still active for a particular elevator. If there are still hall request that have not been served and the request are above the car position, then it will be 1. Similarly if the request is below the car position then it will be -1. If an elevator has served all of its requests, then it will be 0. Example below shows that elevator 1 and 4 has served all of its requests while elevators 2 and 3 still have active requests below and above them respectively.

\[
\text{moving_status} = \\
\begin{array}{cccc}
0 & -1 & 1 & 0 \\
\end{array}
\]

**Carcall:** it represents the car requests that are active or registered and it has an array size of 4 x 9. Each row represents each elevator and each column represents each floor in the building where the first row represents elevator 1 and so on. The numbers shown below represent the active car requests and their corresponding floors.

\[
\text{Carcall} = \\
\begin{array}{cccccccc}
1 & \text{N} & \text{N} & \text{N} & \text{N} & 5 & \text{N} & \text{N} & 8 & \text{N} \\
\text{N} & \text{N} & \text{N} & \text{N} & \text{N} & \text{N} & \text{N} & \text{N} & \text{N} & \text{N} \\
\text{N} & \text{N} & \text{N} & \text{N} & \text{N} & \text{N} & \text{N} & \text{N} & \text{N} & \text{N} \\
\text{N} & \text{N} & \text{N} & \text{N} & \text{N} & \text{N} & \text{N} & \text{N} & \text{N} & \text{N} \\
\end{array}
\]

**Unloading:** It represents the number of people that is supposed to leave the elevator at a specific floor. It has the same dimension as **Carcall** which is 4 x 9. Similar to **Carcall,** each row represents each elevator and each column represents the number of people leaving the elevator at a specific floor where the first row represents elevator 1 and the first column represents the first floor and so on. The numbers shown in figure 3.21 below represents the number of people leaving the elevator at the corresponding floors.

\[
\text{Unloading} = \\
\begin{array}{cccccccc}
5 & \text{N} & \text{N} & \text{N} & \text{N} & 4 & \text{N} & \text{N} & 2 & \text{N} \\
\text{N} & \text{N} & \text{N} & \text{N} & \text{N} & \text{N} & \text{N} & \text{N} & \text{N} & \text{N} \\
\text{N} & \text{N} & \text{N} & \text{N} & \text{N} & \text{N} & \text{N} & \text{N} & \text{N} & \text{N} \\
\text{N} & \text{N} & \text{N} & \text{N} & \text{N} & \text{N} & \text{N} & \text{N} & \text{N} & \text{N} \\
\end{array}
\]

6. Results and Discussions

The simulation was run for 5 different traffic patterns using the four different fuzzy rule sets. For each traffic pattern, two quantities which are **average waiting time** and **average distance** were evaluated for different arrival rates ranging
from 5 to 38 hall requests per 5 minutes using four different fuzzy set rules with each trying to minimize theirs respective parameters.

Each simulation case was run for a simulated time of 1 hour for each traffic pattern. The averages obtained are based on the number of hall requests registered within the 1 hour simulation time. The traffic patterns are busy, moderate, up-peak, down-peak, off-peak and constant whereby their specification were defined in Table 1. The results are shown in Tables 2, 3, 4 and 5. Graphical representations of the obtained results are shown in figures 2, and 3.

From the results shown in the tables and the figures, it can be seen that the waiting time is minimum when the waiting time minimization logic is used. Similarly, distance travelled is minimum when the distance minimization logic is chosen for all five traffic patterns. Minimization of distance travelled is basically means minimization of energy consumption. The least the elevator travel, the least the energy is needed to move the elevators. As for the minimization for loading, it has the highest value of averages for both the waiting time and distance travelled in all traffic modes because it only concentrates of minimizing the number of passengers in the elevator instead of the waiting time or distance which is the main concern in this system. Another important conclusion; is that the logic that minimizes all parameters simultaneously (optimum) produces good result because its averages stays close to the minimum value of the logic that minimizes the concerned quantities.

The average value for both waiting time and distance travelled decreases as the traffic reduces from busy to moderate and finally to off-peak. This is due to the reduction of arrival rate for each traffic whereby busy has the highest arrival rates followed by moderate and lastly off-peak.

7. Conclusion

A fuzzy logic-based controller of a four-car elevator was presented in this paper. A description of the configuration of the elevator including its design features, its inputs and outputs parameters were given. A set of fuzzy rules was drawn based on practical considerations; mainly minimization of waiting time and energy consumption. The simulation and execution of these rules using MATLAB showed the effectiveness of the controller in minimizing the waiting time compared to the works in the literature. However a hardware implementation is be accomplished in order to ascertain the simulation results.

References


Table 1. Modes of Traffic Generation

<table>
<thead>
<tr>
<th>Traffic Mode</th>
<th>Interval</th>
<th>Arrival Rate</th>
<th>Hall request generation</th>
<th>Car request generation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Busy</td>
<td>1 hall request in every 8 – 12 seconds</td>
<td>25 – 38 hall requests per 5 minutes</td>
<td>floor 1-9</td>
<td>floor 1-9</td>
</tr>
<tr>
<td>Moderate</td>
<td>1 hall request in every 10 – 20 seconds</td>
<td>15 – 30 hall requests per 5 minutes</td>
<td>floor 1-9</td>
<td>floor 1-9</td>
</tr>
<tr>
<td>Up–peak</td>
<td>1 hall request in every 12 – 30 seconds</td>
<td>10 – 25 hall requests per 5 minutes</td>
<td>floor 1</td>
<td>floor 2-9</td>
</tr>
<tr>
<td>Down-peak</td>
<td>1 hall request in every 12 – 30 seconds</td>
<td>10 – 25 hall requests per 5 minutes</td>
<td>floor 2-9</td>
<td>floor 1</td>
</tr>
<tr>
<td>Off-peak</td>
<td>1 hall request in every 12 – 30 seconds</td>
<td>10 – 25 hall requests per 5 minutes</td>
<td>floor 1-9</td>
<td>floor 1-9</td>
</tr>
<tr>
<td>Constant</td>
<td>Interval is selected manually ranging from 8 – 60 seconds</td>
<td>Arrival rate ranging from 5 – 38 hall request per 5 minutes</td>
<td>floor 1-9</td>
<td>floor 1-9</td>
</tr>
</tbody>
</table>

Table 2. Averages for minimizes waiting time only for 5 traffic mode

<table>
<thead>
<tr>
<th>Traffic mode</th>
<th>Busy</th>
<th>moderate</th>
<th>Up-peak</th>
<th>Down-peak</th>
<th>Off-peak</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total waiting time (s)</td>
<td>1128</td>
<td>574</td>
<td>1232</td>
<td>1034</td>
<td>301</td>
</tr>
<tr>
<td>Total distance (m)</td>
<td>2259</td>
<td>1426</td>
<td>2864</td>
<td>2109</td>
<td>886</td>
</tr>
<tr>
<td>Total loading</td>
<td>3493</td>
<td>2300</td>
<td>1727</td>
<td>1594</td>
<td>1725</td>
</tr>
<tr>
<td>Average waiting time (s)</td>
<td>3.14</td>
<td>2.32</td>
<td>7.29</td>
<td>6.30</td>
<td>1.75</td>
</tr>
<tr>
<td>Average distance (s)</td>
<td>6.29</td>
<td>5.77</td>
<td>16.95</td>
<td>12.86</td>
<td>5.15</td>
</tr>
<tr>
<td>Average loading</td>
<td>10</td>
<td>9</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Total number of input</td>
<td>359</td>
<td>247</td>
<td>169</td>
<td>164</td>
<td>172</td>
</tr>
</tbody>
</table>
Table 3. Averages for minimizes distance only for 5 traffic mode

<table>
<thead>
<tr>
<th>Traffic mode</th>
<th>Busy</th>
<th>moderate</th>
<th>Up-peak</th>
<th>Down-peak</th>
<th>Off-peak</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total waiting time (s)</td>
<td>1364</td>
<td>706</td>
<td>1440</td>
<td>1044</td>
<td>353</td>
</tr>
<tr>
<td>Total distance (m)</td>
<td>2172</td>
<td>1385</td>
<td>2660</td>
<td>2019</td>
<td>807</td>
</tr>
<tr>
<td>Total loading</td>
<td>3104</td>
<td>2383</td>
<td>1941</td>
<td>1533</td>
<td>1650</td>
</tr>
<tr>
<td>Average waiting time (s)</td>
<td>3.80</td>
<td>2.86</td>
<td>8.37</td>
<td>6.29</td>
<td>2.02</td>
</tr>
<tr>
<td>Average distance (s)</td>
<td>6.05</td>
<td>5.61</td>
<td>15.47</td>
<td>12.16</td>
<td>4.61</td>
</tr>
<tr>
<td>Average loading</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>Total number of input</td>
<td>359</td>
<td>247</td>
<td>172</td>
<td>166</td>
<td>175</td>
</tr>
</tbody>
</table>

Table 4. Averages minimizes loading only for 5 traffic mode

<table>
<thead>
<tr>
<th>Traffic mode</th>
<th>Busy</th>
<th>moderate</th>
<th>Up-peak</th>
<th>Down-peak</th>
<th>Off-peak</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total waiting time (s)</td>
<td>2340</td>
<td>1396</td>
<td>1336</td>
<td>1144</td>
<td>832</td>
</tr>
<tr>
<td>Total distance (m)</td>
<td>3978</td>
<td>2562</td>
<td>3057</td>
<td>2175</td>
<td>1656</td>
</tr>
<tr>
<td>Total loading</td>
<td>3473</td>
<td>2420</td>
<td>1856</td>
<td>1708</td>
<td>1841</td>
</tr>
<tr>
<td>Average waiting time (s)</td>
<td>6.41</td>
<td>5.92</td>
<td>7.68</td>
<td>6.69</td>
<td>4.89</td>
</tr>
<tr>
<td>Average distance (s)</td>
<td>10.90</td>
<td>10.86</td>
<td>17.57</td>
<td>12.72</td>
<td>9.74</td>
</tr>
<tr>
<td>Average loading</td>
<td>10</td>
<td>10</td>
<td>11</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>Total number of input</td>
<td>365</td>
<td>236</td>
<td>174</td>
<td>171</td>
<td>170</td>
</tr>
</tbody>
</table>

Table 5. Averages for minimizing all parameters simultaneously for 5 traffic mode

<table>
<thead>
<tr>
<th>Traffic mode</th>
<th>Busy</th>
<th>moderate</th>
<th>Up-peak</th>
<th>Down-peak</th>
<th>Off-peak</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total waiting time (s)</td>
<td>1220</td>
<td>690</td>
<td>1308</td>
<td>1162</td>
<td>317</td>
</tr>
<tr>
<td>Total distance (m)</td>
<td>2226</td>
<td>1482</td>
<td>2746</td>
<td>2214</td>
<td>816</td>
</tr>
<tr>
<td>Total loading</td>
<td>3273</td>
<td>2416</td>
<td>1998</td>
<td>1683</td>
<td>1816</td>
</tr>
<tr>
<td>Average waiting time (s)</td>
<td>3.40</td>
<td>2.79</td>
<td>7.69</td>
<td>6.60</td>
<td>1.88</td>
</tr>
<tr>
<td>Average distance (s)</td>
<td>6.20</td>
<td>6.00</td>
<td>16.15</td>
<td>12.58</td>
<td>4.83</td>
</tr>
<tr>
<td>Average loading</td>
<td>9</td>
<td>10</td>
<td>12</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>Total number of input</td>
<td>359</td>
<td>247</td>
<td>170</td>
<td>176</td>
<td>169</td>
</tr>
</tbody>
</table>
Figure 1. membership function for waiting time input

Figure 2. Averages for waiting time at various constant arrival rates

Figure 3. Averages for distance at various constant arrival rates