Developing a Toolbox for Modeling and Simulation of Elevators

Reggie Davidrajuh

Electrical and Computer Engineering
University of Stavanger
Stavanger, Norway
e-mail: Reggie.Davidrajuh@uis.no

***Abstract* - The objective behind this paper is to present a MATLAB Toolbox for elevator simulations; this toolbox is known as 'ElevatorSIM', and it is still under development. ElevatorSIM is based on the GPenSIM, which is software for modeling, simulation, performance analysis, and control of discrete-event systems. In this paper: firstly, the software GPenSIM (General-purpose Petri Net Simulator) is introduced. Secondly, the toolbox ElevatorSIM is presented; the components of ElevatorSIM and some of the functions are also presented. Finally, an application example is given to show the usefulness of the toolbox. The application example compares the performance of two different elevator algorithms. The idea behind the development of ElevatorSIM is to measure the performance of elevators through simulations; for this purpose, steps have been taken to collaborate with a high-tech elevator company. However, this paper – the first paper on ElevatorSIM – is to introduce the existence of the toolbox.**

Keywords - Elevators; Petri Nets; GPenSIM; Discrete-Event Systems, ElevatorSIM.

I. INTRODUCTION

Elevators are crucial components in the modern world, to transport human and materials between the different levels of buildings. The primary objective behind this paper is to present a toolbox for elevator simulations (known as '*ElevatorSIM*') that is under development. This toolbox runs on top of the GPenSIM, which is software for modeling, simulation, performance analysis, and control of discrete-event systems.

In this paper: Sections-II and III present the software GPenSIM and *ElevatorSIM*, respectively. Section-IV presents an application example. Further work on *ElevatorSIM* is presented in the section on conclusion (Section-V).

II. GPENSIM

The General-purpose Petri net Simulator (GPenSIM) is a new tool for the modeling, simulation, and performance analysis of discrete-event systems [1-2]. GPenSIM is a MATLAB toolbox that is freely available [3]. GPenSIM is also a real-time controller with which external hardware and software can be controlled [4]. GPenSIM is developed by the author of this paper.

GPenSIM was developed with three specific goals:

1. Easy integration of Petri net models with any other technologies: since the Petri Net model implemented by GPenSIM run on MATLAB, the models can be easily integrated with the numerous toolboxes (e.g., Fuzzy logic, Neural Network, Machine learning, Control systems, and Advanced Statistics) that run on MATLAB. GPenSIM provides a well-designed interface for integrating the Petri Net models with the other toolboxes [1].
2. Simple to learn and use: GPenSIM is simple to learn and use as the programming language of GPenSIM is MATLAB, which is a very simple Basic language clone.
3. Simple to extend: GPenSIM facilitates a file-based model development as all the functions and files exist as M-files. A modeler can easily extend the functions available in GPenSIM either by tailoring the individual M-files and by replacing them.

Implementing a Petri net model with GPenSIM results in four M-files such as the Petri net Definition File, the Main Simulation File, and the pre- and post-processor files:

1. Petri net Definition File (PDF) declares the static structure of the Petri net. The set of places, the set of transitions, and the set of arcs are declared in this file.
2. Main Simulation File (MSF) declares the initial dynamics (such as the initial tokens in the places, and the firing times of the transitions) and runs the simulations.
3. The pre- and post-processor files: the common pre-processor file (COMMON_PRE) declares any additional conditions for the enabled transitions to satisfy before firing. The common post-processor file (COMMON_POST) declares any post-firing actions that are to be performed after firing of transitions.

III. ELEVATORSIM

An elevator (also known as "lift" in the UK and its territories and former colonies) is a vertical transportation mechanism that transports humans and material between levels of a building. The basic characteristics of elevators are given in the following subsection.

A. Basics of Elevators

Modern elevators, especially for human traffic, are powered by electric motors. However, hydraulic-powered elevators are also in use in the heavy industries, due to their enormous lifting power [5]. The electrically operated elevators are often driven by traction ("pulling") cables. Given below are some of the factors that determine the performance of an elevator:

- The traffic: the frequency of all journey (arrival of passengers) that starts at the different levels of the building.
- The elevator algorithm: whether the algorithm is used is a simple algorithm (known as the "standard elevator algorithm," a crude "Shabbat algorithm," or an advanced algorithm that is supported by modern science like machine learning.
- Use of modern technologies that control the elevator (e.g., The Internet of Things connecting people and items through networks).

```
Start
{
 Open door
 Wait for open-door time
 If (no load inside) & (no arrival at any level),
      { Freeze: stop & wait }
 Else
       { Close door if no hindrance detected }
       If (passengers inside car travelling
           in the direction of the car),
              { Keep that direction }
      If (passenger waiting outside in
          forthcoming levels to go in
          the same direction),
              { Stop and pick them }
      If (all the requests from passengers both
         inside and outside are satisfied),
              { Change the direction }
      If (No further requests),
              { Freeze (stop & wait) }
      Else
              { Move to next destination level }
}
```

Figure 1. The Standard Elevator Algorithm (SEA).

B. The Standard Elevator Algorithm

The standard elevator algorithm (SEA) is the one usually implemented in controllers of the elevators, due to its simplicity. The SEA (figure 1) minimizes changes of direction of travel, by using cyclic ('round-trip') travel [6]. The SEA algorithm is also implemented in the *ElevatorSIM* as the standard algorithm.

Though the elevator algorithm shown in figure 1 is simple to execute, it is neither optimal nor suitable for very tall buildings. In tall buildings (e.g., 'skyscrapers') executing a round-trip cycle will take too much time for the passengers. Therefore, newer algorithms can be easily added to the *ElevatorSIM*. The application example given later shows how newer algorithms can be implemented on *ElevatorSIM*.

C. The Petri Net Model

Elevators are discrete-event systems. Thus, it is natural to model an elevator as a Petri Net.

The Petri Net model of an elevator is shown in figure 2. As shown in figure 2, the Petri Net is composed of three modules:

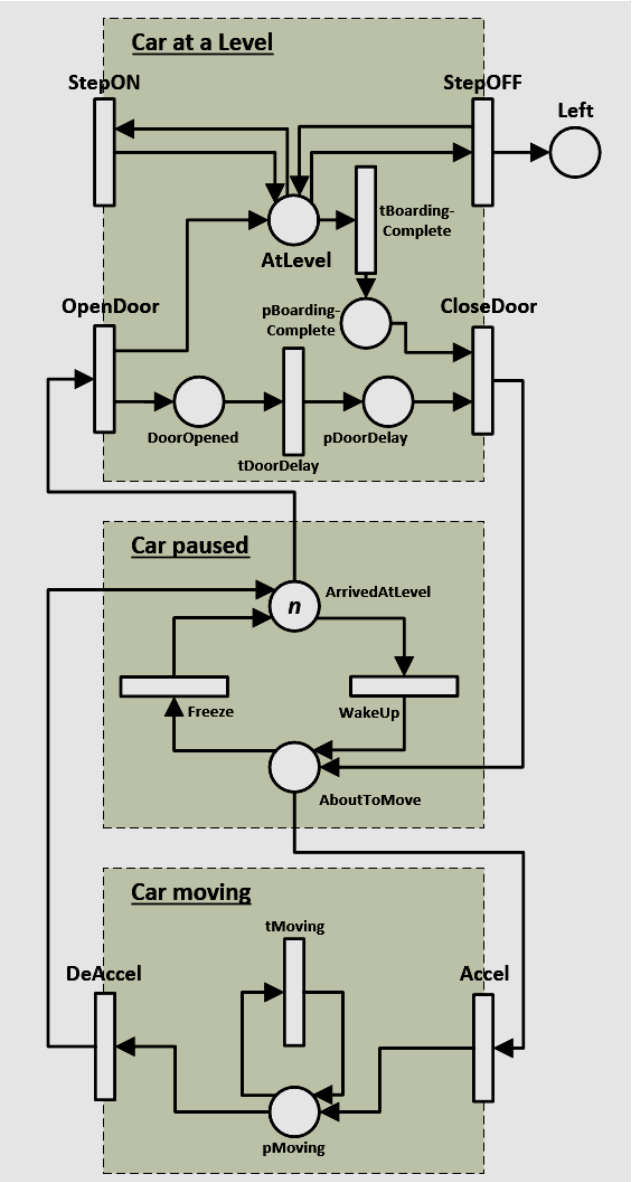

Figure 2. The Petri Net model.

- Car at a level: this module is to capture the activities that are involved when the car arrives at a level. Some of the activities are open the door, let passengers leave and enter, and close door.
- Car paused: when there is no passengers travelling or waiting at any levels, then the car is stopped. It will be wakened up upon arrival of passengers at any level.
- Car moving between levels: this is the module that concerns the activities such as accelerating the car and moving the car between levels. The elevator algorithm controls these two activities. (in *ElevatorSIM*, an elevator algorithm is only applied to these two activities; the other activities are not under the influence of the elevator algorithm.)

Also, the cars are initially introduced at the place "ArrivedAtLevel" (in the module "Car paused"). The number ***n*** represents the number of cars in operation. The passenger arrival is not shown explicitly in the Petri Net. The passenger arrival is a background process that enables the transition "StepON" (in the module "Car at a Level") to fire, whenever a passenger arrives at the relevant level. The place "Left" represents the passengers leaving the car at different levels.

D. Components and the Functionality of ElevatorSIM

The *ElevatorSIM* software has the following components (see figure 4):

1. The Petri Net model.
2. The Elevator Algorithm (the standard elevator algorithm (SEA) is inbuilt; any other newer algorithms (e.g., 'Shabbat') can be easily added if necessary).
3. The global options: E.g., the levels of the building, the capacity (max. number of passengers) of the cars, and passenger arrival distribution.
4. Library functions: the collection of functions that are needed for simulation. Table-III presents some of these functions.
5. Passenger arrival generation (or the loading the history of passenger arrival).
6. The module for analyzing the simulation results and plotting the results.

A. Performance Evaluation

Figure 5 shows the formulae that are computed in the analyzer module.

IV. Application Example

This section presents a simple comparison of two different elevator algorithms to show the simplicity of running simulation with the help of *ElevatorSIM*. The elevator algorithms are 1) The Standard Elevator Algorithm (SEA), and 2) the Shabbat algorithm.

A. Shabbat Elevator

Shabbat elevators originated for religious reasons, where people are not supposed to handle devices on religious days ("Shabbat" days). Thus, Shabbat elevator was designed to operate autonomously without any need for push control button for the user, either inside or outside the car [7]. This means the elevator should stop at every level when there is a passenger inside the car. Also, the elevator must automatically detect the arrival of passengers at different levels and pick them. The Shabbat algorithm is certainly simple as shown in figure 3.

```
Start
{
  Open door
  Wait for open-door time
  If (no passengers inside) and
     (no arrival detected at any level),
          {Freeze}
  else
          {Close door if no hindrance
detected}
   If (end of the level is reached),
          {Change travel direction}
   Move to the next level
}
```

Figure 3. The Shabbat Algorithm.

Comparing with the Shabbat algorithm, the standard elevator algorithm (figure 1) is not cyclic and it not autonomous as its operation is based on the user-operated button inputs.

The purpose of this section is to show that though originated for religious reasons, Shabbat algorithm is useful for certain types of traffic (e.g., large malls with few floors). Two cases are studied using the simulation:

Case-I: A short building (a shopping mall) with very high passenger traffic:

- Number of levels: 4
- Total number of users: 200 (within a period of 30 minutes)
- Arrival rate: a passenger arrives on average every eighth second
- The capacity of the car: max. 20 persons

Case-II: A tall building in which the passenger traffic is light:

- Number of levels: 10
- Total number of users: 20 (within a period of 325 seconds)
- Arrival rate: a passenger arrives on average every fifteen seconds

- The capacity of the car: max. 5 persons.

B. Petri Net Implementation in GPenSIM

As for any Petri Net model implementation in GPenSIM, there are four files for simulation: the main simulation file (MSF), the Petri Net Definition File (PDF), and the processor files (the pre-processor COMMON_PRE and the post-processor COMMON_POST). Due to brevity, none of the program files (code) is shown given in this paper. However, the interested reader is referred to the webpage [8] from where the complete code can be downloaded.

V. Simulation Results

A. Case-I: Short building with Heavy Passenger Traffic

In the case of a short building (a shopping mall) with very high passenger traffic, the seemingly simple and crude Shabbat algorithm outperforms (at least perform equally well) as the standard elevator algorithm. Figure 6 shows the turnaround time (the total travel time between the arrival at the source level and leaving the destination level) for passengers under the two algorithm looks equal. Table-I presents the average time under SEA and Shabbat algorithms.

TABLE I. Case-I: Heavy traffic in a short building

Average time	***SEA***	***Shabbat***
Waiting Time (pickup – arrival)	93.5	90.3
Service Time (travel inside car)	87.5	87.7
Total travel time (dropoff – arrival)	181.0	178.0

Table-I shows that the Shabbat algorithm performs equally well or better for short buildings with heavy passenger traffic. Also, Shabbat algorithm provides some other benefits especially for the elderly and the blind:

- Need not to press any buttons for requesting an elevator.
- While traveling inside the car, need not press any button to indicate the destination.
- While traveling inside the car, if the destination level is missed no need to worry as the car will come back to it in the next cycle.

Considering these additional benefits, we propose that shopping malls switch to Shabbat mode on Saturdays or when a large volume of consumers flocks to the mall.

B. Case-II: Tall Building with Light Passenger Traffic

Figure 7 shows the total travel time experienced by the passengers under the two elevator algorithms. Table-II presents the average time under the two algorithms. As expected, Shabbat algorithm performs poorly. This is obvious as a person wanting to travel from the top floor (10th level) to the ground floor, the elevator will open the door for him, at every level, from 10th level to the ground level.

Average total travel time under SEA and Shabbat are 243.0 and 514.0 seconds, respectively. This means, there is an improvement of 52.7% when we switch from Shabbat algorithm to SEA.

VI. Conclusion

Modern elevators are operated by intelligent devices for optimal transport, taking minimum time for transportation. The intelligent devices that control the elevator operations can detect when a person or material moves toward the elevator door and thus ready an elevator waiting for the transportation. Depending on the history (by the use of RFID tag and machine learning), the intelligent system can also guess the destination of the object to be transported. Thus, the person pressing the call button, waiting for the elevator to arrive, and then press the desired floor-number, all these become obsolete. Because of the combination of modern technologies (such as machine learning and RFID sensors), optimal scheduling algorithms, and model-based control, the turnaround (total travel) time of an object to be transported can be drastically reduced.

TABLE II. Case-II: Light traffic in a tall building

Average time	***SEA***	***Shabbat***
Waiting Time (pickup – arrival)	124.8	334.1
Service Time (travel inside car)	118.2	179.9
Total travel time (dropoff – arrival)	243.0	514.0

This paper presents *ElevatorSIM*, a MATLAB toolbox for modeling and simulation of elevators. *ElevatorSIM* runs on top of the GPenSIM. The idea behind the development of *ElevatorSIM* is to measure the performance of elevators through simulations; for this purpose, steps have been taken to collaborate with a high-tech elevator company. However, this paper – the first paper on *ElevatorSIM* – is to introduce the existence of the toolbox.

References

[1] R. Davidrajuh, Modeling Discrete-Event Systems with GPenSIM: An Introduction; Springer International, The Netherlands, 2018.

[2] R. Davidrajuh, Developing a new Petri net tool for simulation of discrete event systems. In Proceedings of the IEEE Second Asia International Conference on Modeling & Simulation (AICMS 08), Kuala Lumpur, Malaysia, 13–15 May 2008; pp. 861–866.

[3] GPenSIM: A General Purpose Petri Net Simulator. Available online: http://www.davidrajuh.net/gpensim (accessed on 15 December 2017).

[4] R. Davidrajuh, Developing a petri nets based real-time control simulator. Int. J. Simul. Syst. Sci. Technol. IJSSST 2012, 12, 28–36.

[5] Gina Barney. The Elevator Traffic Handbook: Theory and Practice. Spon, London, 2003.

[6] A. Dundes, The Shabbat Elevator. Rowman and Littlefield Publishers, Inc., Lanham, MD, 2002.

[7] Complete Code for the Examples. Available online: http://www.davidrajuh.net/gpensim/Pub/2019/UKSIM/ (accessed on 20 January 2019).

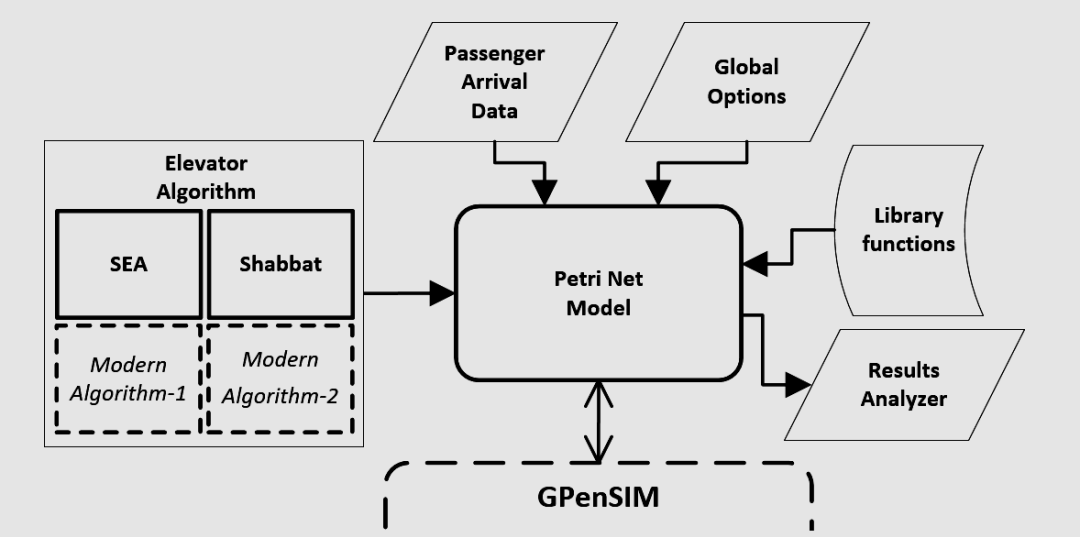

Figure 4. The components of the *ElevatorSIM* software.

TABLE III. SOME OF THE LIBRARY FUNCTIONS

Library function	*Purpose*
Car_Full	Whether the car is full or not
Change_Direction	Change the travel direction of the car
Disp_Current_Passengers	List the passengers (indices) inside the car
Disp_Info	Display the current level and the current time
Is_Extrema_Reached	Whether the extreme levels (top floor or bottom floor) is reached
Fill_Missed_Car_Table	Register when a passenger miss the car because it was full
Number_Of_Passengers	Number of passengers inside the car at a point of time
Passengers_Entering ***Passengers_Leaving*** ***Passengers_Leaving_At_This_Level***	Passengers entering (or leaving) at current level or any other levels
Passengers_Travelling_Up ***Passengers_Travelling_Down*** ***Passengers_Travelling_With_Current_Direction***	Passengers (inside car) currently travelling in the upward, downward, or current direction
People_Waiting_Anywhere ***People_Waiting_At_This_Level*** ***People_Waiting_At_Other_Levels*** ***People_Waiting_In_DownLevels*** ***People_Waiting_In_UpLevels***	Number of passengers waiting at various levels.

$$\boldsymbol{ArrvT}: \textit{the time passenger arrives at the evelvator entrance at the source level}$$
$$\boldsymbol{PickupT}: \textit{the time the passenger enters the car atthe source level}$$
$$\boldsymbol{WaitT}: \textit{the time the passenger wait for the car}$$
$$\boldsymbol{DropT}: \textit{the time the passenger leaves the car at the destination level}$$
$$\boldsymbol{TotalT}: \textit{the total travel time taken for the passenger}$$
$$\boldsymbol{IdealT}: \textit{ideal travel time (the time taken if the passenger travel directly from source to destination, without waiting)}$$

$$\boldsymbol{WaitT} = \boldsymbol{PickupT} - \boldsymbol{ArrvT}$$
$$\boldsymbol{TotalT} = \boldsymbol{DropT} - \boldsymbol{ArrvT}$$
$$\boldsymbol{IdealT} = T_{Close\_Door} + T_{Accelerate} + \left(T_{Inter\_Level\_Travel} \times \Delta_{Levels}\right) + T_{Deaccelerate} + T_{Open\_Door}$$
$$Efficiency\ of\ travel: \boldsymbol{\eta} = \frac{\boldsymbol{IdealT}}{\boldsymbol{TotalT}}$$

Figure 5. The formulae for performance evaluation of elevators.

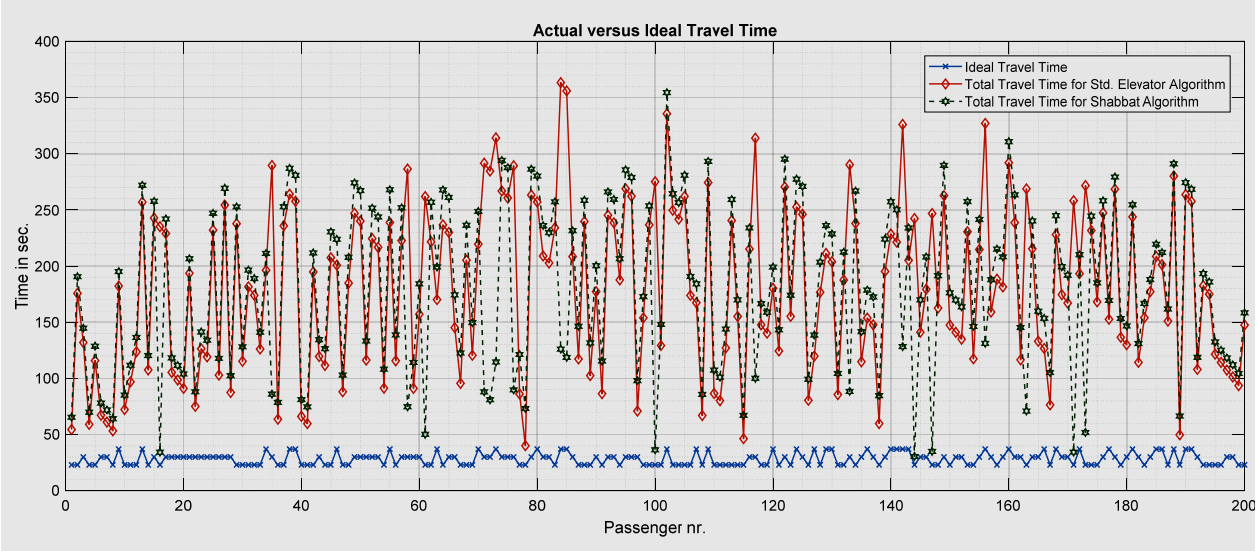

Figure 6. The Turnaround Time under Case-I (Short building with heavy traffic).

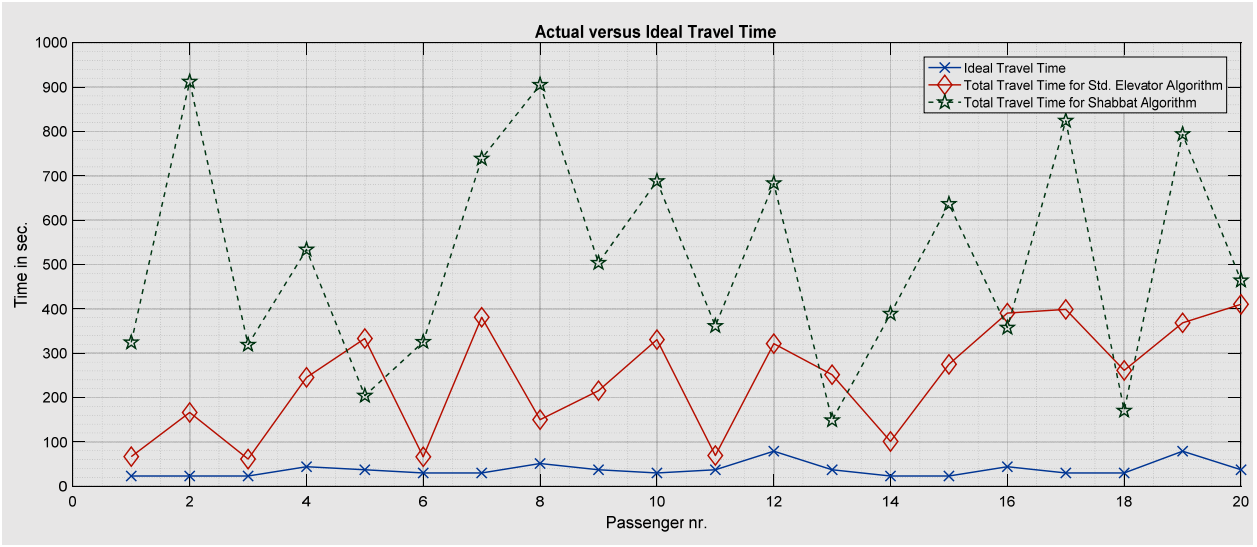

Figure 7. The Turnaround Time under Case-II (Tall building with light traffic).