Research Article

An Accelerated-Time Simulation of Queues at Ticket Offices at Railway Stations

A. Almech ¹ and E. Roanes-Lozano ²

¹Universidad Complutense de Madrid, Madrid 28040, Spain

²Instituto de Matemática Interdisciplinar & Departamento de Didáctica de las Ciencias Experimentales, Sociales y Matemáticas, Facultad de Educación, Universidad Complutense de Madrid, Madrid 28040, Spain

Correspondence should be addressed to E. Roanes-Lozano; eroanes@mat.ucm.es

Received 28 March 2020; Revised 17 April 2021; Accepted 1 May 2021; Published 26 May 2021

Academic Editor: Bekir Sahin

Copyright © 2021 A. Almech and E. Roanes-Lozano. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Although nowadays many railway tickets are bought online, still many are bought through rail appointed travel agents and ticket offices at stations. There are several works on microscopic and accelerated-time simulations, some of them related to the topic of this paper, treating passengers movements in railway stations (both of general purpose and also with a focus on specific topics like evacuation, stations design, ticketing, etc.). We focus on a very specific topic: modelling queuing at ticket offices at a main Spanish station where “AVE” (“High-Speed”), “Larga Distancia” (“Long-Distance”), “Media Distancia” (“Middle-Distance”), and “Cercanías” (“Suburban Services”) dedicated windows exist. The existence of “Last Minute” desks is also considered. The goal is to provide the user with a tool that allows to choose the best option for windows distribution along time, after different microscopic simulations based on given data and windows possible distribution are performed (as done in a previous work of one of the authors for airport terminals check-in counters). Special attention is paid to “Last Minute” windows and shared windows (for example simultaneously selling tickets for “Larga Distancia” and “Media Distancia”). Input is given by arrival curves or can be generated by the package. The output is the detailed situation of any window at any moment and the evolution of queues by train or window type. There are different further possible extensions of this work. The implementation has been developed in a computer algebra system in order to minimize the development time.

1. Introduction

Nowadays many railway tickets are bought online or through mobile apps, and electronic tickets have been adopted for many services (like Suburban Services). Therefore the use of traditional ticket offices and even of train ticket vending machine at railway stations is decreasing. Nevertheless, it is important to model the flow of passengers at peak moments or dates in front of the windows of the ticket office of a railway station. This can improve the service and therefore the customer’s satisfaction. It can be applicable in other environments too.

The purpose of this work is to enable its user (the railway infrastructure administrator running the railway station) to simulate different possible distributions of these windows

with a computer tool that allows choosing the best option for windows distribution along time, after performing and comparing different microscopic simulations (as proposed in [1]). The intended user of the software developed is the person in charge of assigning the type of passengers each window of the ticket office will serve.

From the scientific point of view there is no theoretical advance, as it is yet another microscopic simulation, but, as we shall detail in the next section, we know of no software specifically devoted to this particular goal. Nevertheless, we humbly consider that the new work has interest in real life. Consider, for instance, that [1] describes a similar tailor-made software developed within the frame of a research project for an aeronautical consulting regarding a microscopic simulation of the passengers’ flow in a real airport,

running with extremely detailed real data provided by the airport authorities, and the situation addressed here is relatively similar (see the next paragraph). The second author was the main researcher of that research project and he was also one of the authors of the article “An Accelerated-Time Microscopic Simulation of a Dedicated Freight Double-Track Railway Line” [2], developed in cooperation with the Spanish Railway Foundation.

The idea to develop this software arose some time ago when the second author was trying to buy a ticket for a long-distance journey at a certain main station. He was very nervous because the queues at the two long-distance windows were long. This was not an isolated case: some other travellers begged from time to time the people in front to allow them to overtake their queue because otherwise they would miss their train. From time to time, the queues at other windows (devoted to a different kind of trains) were empty and the workers serving them kindly shouted asking for travellers in a hurry to move to their windows.

In the proposed work, the possible different combinations of windows at the railway station ticket office (selling tickets for High-Speed/Long-Distance/Middle-Distance/Suburban Services or more than one service, and also “last minute”) are considered, as mentioned in the Abstract. The problem is approached designing and implementing a virtual machine that “sells” tickets to all the passengers (although they are situated in a unique virtual queue, independently of their kind of train, they are served according to their characteristics). The arrival order is respected, but each virtual passenger is distributed according to the best possible queue (shortest one) according to his characteristics.

The simulation has been implemented in *Maple* (*Maple* is a trademark of *Waterloo Maple Inc.*), a multipurpose and very powerful computer algebra system [3–6].

2. Related Works

There is a huge bibliography on computational approaches to airport terminals procedures. From the first step of the process: polls [7], baggage handling [8] (the same authors use a similar approach simulating traffic on a motorway in [9]), gate assignment [10], aircraft positioning [11], etc. In particular, passengers flow within the airport terminal has been extensively studied from different points of views [1, 12–15].

Reference [1] is the closest one to this paper. Nevertheless, the present paper deals with a specific process (queueing for ticketing); meanwhile, reference [1] analyses the whole passengers’ flow at airport terminals (very different from passengers’ flow at railway stations). We could underline the following as differences:

In airport terminals, the building is always divided into two isolated parts, the “land side hall” and the “air side hall,” separated by security controls, what is not usually the case in railway stations and therefore is not considered in the present work.

There are ticket offices and/or ticket vending machines on most railway stations, but their distribution (for instance: High-Speed/Long-Distance/Middle Distance/Suburban Services, as considered in this paper) is completely different from the assigning of the check-in counters at airport terminals (where a check-in-counter can be assigned to a certain airline, to a certain airline and restricted to a class or some classes, to a certain flight, to a set of flights, etc.).

Other processes that add complexity in airport terminals, like boarding control, security control, etc., are not common in railway stations (for instance in Spain they only take place for boarding high speed trains) and therefore are not considered in the present work.

There is no final waiting time at gates before boarding (as there are no gates and platforms at railway stations are accessible once the security controls, if any, have been passed).

Finally, the implementation, although written in *Maple* too, has nothing to do with the one in [1] and presents the results in more detail (as it only addresses the queueing process).

There are not so many works in the field of passengers’ flow within railway stations. Some have related goals, like [16], that monitors movements of the real passengers. Other are closer to the work presented here, like [17], simulating the processes from the arrival of the passenger to the railway station, obtaining the ticket at the vending machines, validating the ticket, etc. Meanwhile [18] focuses on the movements of passengers in the railway station (considered as homogeneous pedestrians) using a cellular automaton. Finally, [19] uses Markov chains for modelling queueing at the ticket office of a railway station.

We know of no approach that considers windows dedicated to different kinds of trains. Therefore, we have decided to implement an accelerated-time simulation of queueing at ticket offices at a big Spanish railway station, where ticketing for High Speed/Long Distance/Middle Distance/Suburban Services can have different windows assigned.

3. Materials and Methods

3.1. Formulation. Let us underline the finite character of the process and the discretization of time. It is updated in 1 minute intervals (nevertheless, this can be changed by altering the value of the global variable step in file *procedures.mpl*).

The process can be summarized as follows (Figure 1):

- (1) The timing when the simulation starts and its length has to be introduced.
- (2) The arrival passengers flow is given by appropriate distributions (arrival curves), that depend on the train type and its schedule. In this implementation for railway stations, the end user has two possibilities:

FIGURE 1: Scheme of the model.

“Real life environment”: the arrival data of the passengers for each train are introduced using lists containing the number of passengers arriving each minute that will buy a ticket at the ticket office. The arrival of virtual passengers for a given train will depend on the length of the list provided: it will begin “the length of the list” minutes before the departure time (a calculation that is performed internally). The (internal) list of departing trains to be considered in the simulation (`TrainInput`) is updated later by the user using procedure `addTrain` one time for each new declared train (see step 3 below).

Remark: in [1] real data from passengers’ arrival to the airport under study were available from [7] and from the airport authorities of Malaga Airport (that had entrusted studies to specialized companies). Real data about the airport, flight schedules, and plane characteristics were also provided by the airport authority. In this case we are working on our own and we have used invented data in Examples 1 and 2 below. If the software was used by a railway infrastructure company in a real station, polls would have to be carried out in order to determine the real arrival curves to that station in order to build the corresponding lists of passengers buying a ticket.

“Testing environment”: the arrival data of the passengers for each train (that will buy a ticket at the ticket office) are generated using a statistical distribution in order to determine the delay of each passenger with respect to the moment the simulation for that train starts. For this purpose, a multinomial distribution with parameters n and p has been chosen by default. In this case $p = (p_1, \dots, p_r)$ correspond to the possible delays chosen and n is the number of passengers modelled for that train. This choice is not based on any real world experiments and has only the purpose to easily test the package for validating purposes and to show how it works (see Example 3

below). In this testing environment, the passengers arrive to the station depending on the kind of train. The values assigned by default are 30 minutes in advance for Suburban Service trains; 60 minutes in advance for Middle Distance and Long-Distance trains; and 90 minutes in advance for High-Speed trains (although this can be changed by altering the values of the global variable `randomLength` in file `randomize.mpl`).

(3) The data of the trains are introduced depending on the environment chosen.

“Real life environment”: the “number” of the train, its type, departure time, and list of passengers buying a ticket minute by minute are introduced manually for each train (using procedure `addTrain`). The number of passengers is obtained from the corresponding list of step 2.

“Testing environment”: the data for the trains (departure time, number of passengers to be generated and “number” of the train) are introduced using four lists (`SuburbanTrain`, `MDTrain`, `LDTrain`, and `HSTrain`) and the process is carried out automatically.

(4) The distribution of the windows and the serving timing for each type of the train have to be introduced, as well as the time to redirect a passenger to the last minute queue (if applies). The variables where they have to be stored are `suburban`, `middledistance`, `long-distance`, and `highspeed` (in case the window is a shared one, it appears in more than one of these lists and the average serving time applied will then depend on the type of passenger). A special case is constituted by the last minute windows (if applicable). The “minutes to departure” required to be reallocated in such a window is stored in the global variable `alarm` and the windows serving this kind of passengers have to be input in list `lastminute` (the serving time will also depend on the type of passenger and will be the one given for those windows).

(5) Each minute, all the passengers arriving are provisionally stored in an “arrival queue” (a virtual inexistent single queue).

(6) Every minute, the passengers in the “arrival queue” are moved to the best possible window queue (the shortest one), according to their characteristics (considering if they are last minute passengers, if this kind of window exists). To achieve this, the list of windows is pruned for each passenger and he/she is directed to the shortest one in the pruned list (in case of the existence of more than one queue with the same length, he/she is placed in the window with the lowest number).

(7) Each window at the ticket office works at a certain unloading speed (given by the average time that the kind of passenger requires) and is also updated every minute (eliminating from each window queue the already served passengers). The passengers’ output flow from the window queues therefore consists of the passengers that already have their tickets.

(8) Every minute the system looks for last minute passengers at the window queues (if applies) and moves each one to the shortest last minute queue (if and only if this queue is shorter than the one the passenger is at).

3.2. Elements of the Model. Once the problem is formulated, we can describe the model and dynamics of the ticket office of a big railway station.

The following elements are considered (the computational details of the input are detailed in the examples below):

Passenger: they are internally described by a list with the format

`[train type and number, arrival time, state]`

Such a list is automatically generated by the package for each arriving passenger. The “arrival time” corresponds to the arrival of the passenger to the railway station. The state is the time required (or left) to obtain a ticket.

Train: they are internally described by a list with the format

`[departure time, number of passengers, train number]`

Windows distribution and serving times lists: a list has to be introduced by the end user for each type of train served with the format:

`[set of window identifiers, average serving time]`

Window: an internal list of passengers (waiting in the queue of that window of the ticket office).

3.3. Process of the Model. As said above the time is discretized. The time variable is denoted as t .

The nomenclature used in Figure 1 and more details of the different elements involved can be found afterwards. Note that $e(t)$ is initialized as an empty list and $q(t)$ as a list of

empty lists (of appropriate length: the number of windows at the ticket office).

$e(t)$ is the passengers’ virtual ‘arrival queue’, that is, the list of all passengers that have already arrived at time t (with their characteristics), built according to the arrival curves of each train.

$q(t)$ is the ‘windows state’: it is the list of queues in front of the windows at time t . Note that, due to the discrete nature of the process, the passengers in this state have waited at least one temporal step at the arrival queue, that is, have stayed in $e(t-1)$, before moving to $q(t)$.

Window choice: the passengers in $e(t)$ choose (one after the other) the window with the shortest queue in $q(t)$ (of those appropriate for him). The node returns the updated list of windows that will enter the transition clock.

$s(t)$ is the list of passengers that already have their tickets (and can go to their corresponding platforms).

Clock: it advances e , q , and s one temporal step (1 minute by default):

Updating $e(t)$ to $e(t+1)$, by erasing those passengers moving to the window queues and adding those passengers arriving to the station

Updating $q(t)$ to $q(t+1)$, by erasing from each queue those passengers that have got their tickets and adding those who come from the arrival queue

Updating $s(t)$ to $s(t+1)$, a list including the passengers that have obtained their tickets at minute t

Urgency: this label allows passengers that enter the “last minute” category to choose that window (if it exists).

The possible existence of last minute windows makes the window choice more complex (Figure 2). The choice of the last minute passengers prioritizes the departure time over the type of train. They somehow break the ordering of the main queue.

4. Results and Discussion

The main goal of the package is to obtain the state of each window at each minute. This is accomplished from different points of view, as shown in the following examples, as the end user can obtain:

The plot of the passengers’ arrival, classified by train type, for any time interval

A diagram of the passengers queueing at any moment

The plot of the evolution of the queues by window type, for any time interval

The plot of the evolution of the queues by train type, for any time interval

The visual nature of the output allows to easily correct (or improve) an unsatisfactory situation.

The results are very accurate due to the short step of the simulation, but obviously depend on the accuracy of the data

FIGURE 2: Flowchart of the process with shared windows and last minute windows (T is the ticketing time for that passenger).

introduced (that is, in case of real data, on the accuracy and adequacy of the pols carried out).

Some examples are included afterwards as illustration of the possibilities of the package.

4.1. Examples Introducing the Arrival Data through Lists

Example 1. Only two Middle-Distance trains (M2378 and M4552) are considered, one scheduled at 19:30 and the other one at 19:50. The system is restarted and the code is loaded firstly. Afterwards the starting time for the simulation and its duration are introduced and procedure `intitalize` has to be executed:

```

restart:
read('C:/.../procedures.mpl'):
initial := minute(18, 10, 0):
duration := 110:
inititalize():

```

Now the arrival lists of passengers for each train (that will buy a ticket) have to be introduced (in this case using a list for each train):

```

Train2378 := [0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 2, 3,
1, 2, 2, 3, 0, 2, 0, 3, 2, 1, 0, 0, 2, 2, 3, 3, 1, 2,
1, 2, 2, 0, 1, 0, 0, 0, 0, 0, 2, 1, 1, 1, 1, 2, 0, 2,
2, 0, 0, 2, 1, 1, 0, 0, 0, 0, 0, 0, 1]:
addTrain(Train2378 , MDTrain , minute(19,
30, 0), Train2378):
Train4552 := [0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0,
0, 1, 0, 0, 2, 4, 1, 4, 1, 0, 4, 0, 2, 2, 1, 4, 3, 1,
2, 3, 0, 1, 2, 3, 1, 0, 1, 1, 2, 1, 3, 0, 1, 0, 1, 0,
1, 1, 2, 0, 0, 0, 0, 1, 0, 0, 0, 0]:
addTrain(Train4552 , MDTrain , minute(19,
50, 0), Train4552):

```

(According to what was stated above, these lists have 60 elements.)

Afterwards the type of trains each window (numbered 1 to 18 in this case) serves and the average time to serve this kind of passenger, as well as the 'minutes to departure' required to be reallocated in a last minute window (global variable `alarm`), are specified:

```

suburban := [{1, 3, 5, 6, 7, 8, 9}, 1.0]:
middledistance := [{10, 11, 12}, 1.5]:

```

```

longdistance := [{13, 14, 15, 16}, 2.0]:

```

```

highspeed := [{13, 14, 15, 16}, 1.5]:

```

```

alarm := 10.0:

```

```

lastminute := [{17, 18}, alarm]:

```

Now, the plot of the passengers' arrival can be requested (Figure 3):

```

eGraphic(TrainInput);

```

And after running the simulation by typing

```

run(TrainInput):

```

a diagram of the passengers at any moment at each window, representing each passenger by a coloured ball (Figure 4) can be requested. The queue in front of each window is represented by a column of balls. The windows are represented in the abscissa axis. Each circle has a colour depending on the type of train of the passenger:

red for suburban,
navy blue for middledistance,
green for longdistance,
cyan for highspeed.

It is enough to type, for instance

```

instant := minute(19, 19, 0):
plotWindows(instant);

```

Also, a plot of the evolution of the maximum length of the queues for each type of window (Figure 5) and a plot of the evolution of the accumulated queues for each type of train (Figure 6), both at the given instant, can be asked for by typing

```

maxQueus();
TypeTrain();

```

This simulation takes little more than half a second on a standard laptop computer running *Maple 2021*.

Let us finally remark that it is possible to change the windows assigning on the move using procedure `movements`.

Example 2. in this example the same data as in Example 1 are considered, except that window 12 is not only selling tickets for Middle-Distance trains but also for Long-Distance trains:

```

longdistance := [{12, 13, 14, 15, 16}, 2.0]:

```

In this case, the Long-Distance passengers queues do not appear empty: if `maxQueus()` procedure was executed, the

FIGURE 3: Passengers' arrival curve.

FIGURE 4: Passengers queuing at 19:19:00. There are only Middle-Distance passengers (navy blue balls).

FIGURE 5: Evolution of the queues by window type.

plot in Figure 7 would be obtained (instead of the one in Figure 5).

4.2. Example Using the Automatic Generation of Data from a Statistical Distribution

Example 3. Let us consider now a more complex example with 9 suburban trains, 2 Middle-Distance trains, 1 Long-

Distance train, and 1 High-Speed train scheduled between 19:00 and 20:00. There is no difference in the beginning with respect to the previous examples.

```
restart:
read('C:/.../procedures.mpl'):
initial := minute(18, 10, 0):
duration := 110:
```

FIGURE 6: Evolution of the accumulated queues by train type.

FIGURE 7: Evolution of the queues by window type (only Middle-Distance trains are introduced, but there is a shared window for Middle-Distance and Long-Distance trains).

```

initialize():
SuburbanTrain := [[minute(19, 16, 0), 40, C0001],
[minute(19, 20, 0), 40, C0011],
[minute(19, 25, 0), 40, C0002],
[minute(19, 30, 0), 40, C0021],
[minute(19, 35, 0), 40, C0003],
[minute(19, 40, 0), 40, C0031],
[minute(19, 45, 0), 40, C0004],
[minute(19, 50, 0), 40, C0041],
[minute(19, 55, 0), 40, C0005]]:
MDTrain := [[minute(19, 30, 0), 40, M0001],
[minute(19, 50, 0), 40, M0002]]:
LDTrain := [[minute(20, 0, 0), 50, L0001]]:
HSTrain := [[minute(19, 40, 0), 60, AVE0001]]:
suburban := [{1, 3, 5, 6, 7, 8, 9}, 1.0]:

```

```

middledistance := [{10, 11, 12}, 1.5]:
longdistance := [{13, 14, 15, 16}, 2.0]:
highspeed := [{13, 14, 15, 16}, 1.5]:
alarm := 10.0:
lastminute := [{17, 18}, alarm]:

```

But afterwards, the second file with procedures for passengers generation has to be loaded and the arrival simulation has to be executed:

```

read('C:/.../randomize.mpl'):
AllSimulationArrivals():

```

Then the different plots can be visualized (Figures 8-11):

```

eGraphic(TrainInput);
run(TrainInput):
instant := minute(19, 19, 0):
plotWindows(instant);
maxQueus();
TypeTrain();

```

FIGURE 8: Passengers' arrival curves.

FIGURE 9: Passengers queuing at 19:19:00. There are passengers buying tickets for the four types of trains. Observe that window 13 is a shared one.

FIGURE 10: Evolution of the queues by window type.

FIGURE 11: Evolution of the queues by train type.

This simulation takes less than one second on a standard laptop computer running *Maple 2021*.

5. Conclusions and Future Work

We believe that this can be a useful package for windows distribution planning at ticket offices with dedicated windows at big railway stations. Considering the initial virtual arrival queue makes it possible to easily treat shared windows. A simple approach (finite automata) that uses lists of lists makes it possible to deal with any combination of windows.

Although the percentage of passengers buying tickets at the stations is decreasing more and more due to the arrival of other possibilities (mainly Internet), customers still use windows at ticketing offices for different reasons: lack of ability to use Internet and web applications, help needed, problems with the language, need of personalized help, or just a preference.

As shown above, the model developed and implemented is complex but very flexible: there are no restrictions on the number of trains, the number of passengers that buy tickets, the length of the simulation, and the number and distribution of windows at the ticket office. Moreover, the visual representation of data allows to easily understand the situation (for instance for improving windows distribution). We know of no comparable work.

A possible future work could be to consider more realistic movements of the passengers from their queue to a last minute queue. Now, a scenario where all passengers can see all queues and are aware that “last minute windows” exist is considered (for instance they could be informed from time to time by loudspeakers that these special queues exist and the conditions to be used). But another scenario could be considered, where only a certain ratio of the last minute passengers is aware of the possibility to change to a last minute queue.

Another possible future work could be to randomize the number of passengers for each train within a range (according to its type).

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was partially supported by the research project PGC2018-096509-B-I00 (Government of Spain).

References

- [1] E. Roanes-Lozano, L. M. Laita, and E. Roanes-Macías, “An accelerated-time simulation of departing passengers’ flow in airport terminals,” *Mathematics and Computers in Simulation*, vol. 67, no. 1-2, pp. 163–172, 2004.
- [2] A. Hernando, E. Roanes-Lozano, and A. García-Álvarez, “An accelerated-time microscopic simulation of a dedicated freight double-track railway line,” *Mathematical and Computer Modelling*, vol. 51, no. 9-10, pp. 1160–1169, 2010.
- [3] R. M. Corless, *Essential Maple: An Introduction for Scientific Programmers*, Springer Science & Business Media, Berlin, Germany, 2013.
- [4] A. Heck, *Introduction to Maple*, Springer Science & Business Media, Berlin, Germany, 2011.
- [5] M. J. Wester, *Computer Algebra Systems: A Practical Guide*, John Wiley & Sons, New York, NY, USA, 1999.
- [6] E. Roanes-Macías and E. Roanes-Lozano, *Cálculos Matemáticos por ordenador con Maple V. Rubiños-1890*, Rubinos, Madrid, Spain, 1999.
- [7] A. Canals, J. Rojas, and F. Robusté, “Gestión de Procesos Aeroportuarios en los Aeropuertos de Barcelona y Málaga,” in *El Transporte en el Siglo XXI. Actas del III CIT*, A. López-Pita and F. Robusté, Eds., vol. 2, pp. 731–738, CIMNE, Barcelona, Spain, 1998.
- [8] G. Aguilera-Venegas, E. J. L. Mérida-Casermeiro, P. Rodríguez-Cielos, and P. Rodríguez-Cielos, “An accelerated-time simulation of baggage traffic in an airport terminal,”

- Mathematics and Computers in Simulation*, vol. 104, pp. 58–66, 2014.
- [9] G. Aguilera, J. L. Galán, E. Mérida, J. M. García, P. Rodríguez, and P. Rodríguez, "An accelerated-time simulation of car traffic on a motorway using a CAS," *Mathematics and Computers in Simulation*, vol. 104, pp. 21–30, 2014.
 - [10] S. Yan and C.-M. Huo, "Optimization of multiple objective gate assignments," *Transportation Research Part A: Policy and Practice*, vol. 35, no. 5, pp. 413–432, 2001.
 - [11] J. Pfalzgraf, K. Frank, J. Weichenberger, and S. Stolzenberg, "Design, implementation, and application of a tool for optimal aircraft positioning," in *Proceedings of EvoWorkshops 2000*, S. Cagnoni, Ed., Springer-Verlag LNCS 1803, Berlin-Heidelberg, Germany, pp. 382–393, April 2000.
 - [12] R. Aniyeri and R. Nadar, "Passengers queue analysis in international airports terminals in Kerala using multiphase queuing system," *International Journal of Mathematics in Operational Research*, vol. 12, no. 1, pp. 1–30, 2018.
 - [13] A. Kierzkowski and T. Kisiel, "Simulation model of security control system functioning: a case study of the Wrocław Airport terminal," *Journal of Air Transport Management*, vol. 64, no. B, pp. 173–185, 2017.
 - [14] R. Stolletz, "Analysis of passenger queues at airport terminals," *Research in Transportation Business & Management*, vol. 1, no. 1, pp. 144–149, 2011.
 - [15] P. P.-Y. Wu and K. Mengersen, "A review of models and model usage scenarios for an airport complex system," *Transportation Research Part A: Policy and Practice*, vol. 47, pp. 124–140, 2013.
 - [16] J. P. Li, "Train station passenger flow study," Edited by J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, Eds., in *Proceedings of the 2000 Winter Simulation Conference Proceedings*, vol. 2, pp. 1173–1176, Orlando, FL, USA, December 2000.
 - [17] J. Li and L. Wang, "Microscopic simulation on ticket office of large scale railway passenger station," in *Proceedings of the 7th Advanced Forum on Transportation of China (AFTC 2011)*, pp. 41–47, Beijing, China, October 2011.
 - [18] T.-Q. Tang, Y.-X. Shao, and L. Chen, "Modeling pedestrian movement at the hall of high-speed railway station during the check-in process," *Physica A: Statistical Mechanics and Its Applications*, vol. 467, pp. 157–166, 2017.
 - [19] J. Ponicki, J. Camaj, and M. Kendra, "Possibilities of simulation tools for describing queuing theory and operations service lines in railway passenger transport," in *Proceedings of the 2016 International Conference on Engineering Science and Management. Advances in Engineering Research Series*, November 2016.