http://doi.org/10.35784/iapgos.7081 *received: 15.01.2025 | revised: 19.08.2025 | accepted: 15.09.2025 | available online: 30.09.2025*

MODELING OF INTERCEPTION PARKING LOTS

Larysa Gumeniuk, Volodymyr Lotysh, Pavlo Humeniuk, Oleksandr Reshetylo, Yuriy Syrota
Lutsk National Technical University, Lutsk, Ukraine

Abstract. *This paper is devoted to solving the problem of effective traffic flow management and parking lot organization by creating interception parking lots that help reduce traffic jams and improve parking availability. An algorithm has been proposed, and a simulation model of an interceptor parking lot has been implemented in Python. The development allows for the examination of the dynamics of parking lots, taking into account changes in traffic intensity throughout the day and week. It provides extensive opportunities for configuring parameters such as the number of lanes, parking lot capacity, vehicle arrival intensity, and duration of stay. The developed model can be helpful for urban planners, commercial parking lot owners, and researchers working on optimizing the use of urban space. Its integration with modern IoT solutions opens up prospects for the creation of intelligent parking systems that will help reduce traffic congestion, increase driver comfort, and reduce the impact of motor transport on the city's environment.*

Keywords: computer simulation, public transport, urban planning

MODELOWANIE PARKINGÓW PRZECHWYTUJĄCYCH

Streszczenie. *Niniejszy artykuł poświęcony jest rozwiązaniu problemu efektywnego zarządzania przepływem ruchu i organizacji parkingów poprzez tworzenie parkingów przechwytujących, które pomagają zmniejszyć korki i poprawić dostępność miejsc parkingowych. Zaproponowano algorytm i zaimplementowano model symulacyjny parkingu przechwytującego w języku Python. Pozwala on na badanie dynamiki parkingów z uwzględnieniem zmian natężenia ruchu w ciągu dnia i tygodnia. Zapewnia szerokie możliwości konfiguracji parametrów, takich jak liczba pasów ruchu, pojemność parkingu, intensywność przyjazdu pojazdów i czas pobytu. Opracowany model może być pomocny dla urbanistów, właścicieli parkingów komercyjnych oraz badaczy pracujących nad optymalizacją wykorzystania przestrzeni miejskiej. Jego integracja z nowoczesnymi rozwiązaniami IoT otwiera perspektywy tworzenia inteligentnych systemów parkingowych, które pomogą zmniejszyć korki, zwiększyć komfort kierowców i zmniejszyć wpływ transportu samochodowego na środowisko miasta.*

Słowa kluczowe: modelowanie komputerowe, transport publiczny, planowanie urbanistyczne

Introduction

In modern cities around the world, there is an increasing need for effective traffic flow management and the optimal use of available space. One of the pressing issues is the organization of parking facilities, particularly park-and-ride lots, which can help alleviate congestion in central (historical) parts of cities and improve parking accessibility for residents and visitors.

Parking in historical city areas can be a challenging and complex issue that requires a thoughtful and comprehensive approach. Solutions may involve a combination of measures, such as parking management, regulation, pricing policies, public transportation, and alternative mobility options.

The evolution of car parking has progressed over time, with early forms of parking consisting mainly of on-street parking or open lots. In the early 20th century, cities began constructing multi-level parking garages to accommodate the growing number of vehicles on the roads.

One significant advancement in this area was the introduction of automated parking systems, which first emerged in Southern Europe [9, 15]. These systems enabled vehicles to be efficiently parked in dense urban environments where available parking spaces were limited.

Modern parking systems can incorporate elements of artificial intelligence, automated control, and real-time parking status monitoring [6, 10].

In recent years, there has been a shift towards environmentally friendly parking solutions, such as charging stations for electric vehicles and "park and ride" systems, aimed at improving the efficiency and convenience of vehicle parking [14].

Technologies such as mobile applications and smart parking systems allow drivers to easily locate available parking spaces, register their parking time, and even make payments via their smartphones. This significantly enhances the convenience and efficiency of parking, especially with quick and easy access to information.

Many cities now employ intelligent parking systems that use sensors to detect when a parking space is occupied or vacant, providing real-time information to drivers through mobile apps or street displays. This technology can help reduce traffic congestion and make it easier for drivers to find available parking spaces.

In addition to the aforementioned changes, progress is also being made in the design and construction of parking facilities. For example, many modern parking structures are now being built with energy-efficient lighting and ventilation systems, and some incorporate green roofs and other design elements.

Another area of parking development is the use of autonomous vehicles. As self-driving cars become more widespread, the need for traditional parking structures may decrease, as these vehicles will be able to drop off passengers and then independently drive to designated parking spots.

The focus on traditional transportation may also stimulate the development of new parking solutions. For instance, charging stations for electric vehicles and parking areas will become more widespread, as well as the use of shared transportation options such as bike and scooter sharing.

Overall, the development of parking facilities has been driven by the need to accommodate the growing number of cars on the road, as well as address issues such as congestion and space efficiency. As technologies and societal needs continue to evolve, it is highly likely that new and innovative solutions for vehicle parking will be developed.

The modeling and optimization of park-and-ride facilities is becoming an increasingly relevant task in the context of urban infrastructure development. In general, park-and-ride facilities can play a crucial role in enhancing the comfort of urban areas by reducing traffic congestion, air pollution, and parking challenges. This approach can also help make cities more accessible, convenient, and safe for all residents and visitors, while supporting the growth of local businesses and the economy.

1. Analysis of research and publications

Parking in historical parts of cities can be a challenging and multifaceted issue. These areas are often characterized by narrow streets, limited access, and a high demand for parking spaces. The restricted availability of parking in such districts can lead to difficulties in finding spaces, high parking costs, and negative impacts on the local economy, visitor experiences, preservation of historical structures, and the aesthetic appeal of the area.

In this context, it is crucial to understand the specific problems and challenges cities face concerning parking in historical districts and to develop comprehensive solutions that balance the needs of visitors, residents, and the preservation of historical areas.

Parking in historical city areas faces numerous challenges, including:

- limited space: many historical districts feature narrow streets and small parking lots, making it difficult to find available parking spaces;
- high demand: tourists and visitors frequently flock to historical areas, leading to a high demand for parking;
- restricted access: certain areas may have limited vehicle access, such as pedestrian-only zones, complicating the search for parking;
- high costs: parking in historical districts can be expensive due to high demand and limited supply;
- damage to historical buildings: parked vehicles in these areas can cause damage to building structures, facades, and streets;
- traffic congestion: a high concentration of parked vehicles in historical areas can lead to road congestion, hindering the movement of both residents and visitors;
- incompatibility with historical character: vehicles parked in historical districts can be incongruent with the area's historical ambiance, diminishing the aesthetic appeal of the neighborhood.

In addition to the aforementioned issues, parking in historical districts can also have a negative impact on the environment. Vehicles idling while searching for parking spaces can contribute to air pollution, while increased traffic intensity may result in noise pollution.

Many cities around the world face challenges with parking in historical districts. Let us consider some examples.

Paris, France. Narrow streets and high parking demand in the historic Marais district make it challenging for visitors to find parking spaces. Additionally, many streets in this area are closed to vehicular traffic, further limiting parking options. Currently, all street parking in Paris is paid and limited to a maximum duration of two hours [16].

Munich, Germany. Since the early 1990s, municipal authorities in Munich have been exploring the possibility of changing parking management policies as a way to reduce traffic in the city center. Munich has a complex parking system with various rates and time restrictions [17].

Venice, Italy. The historic city of Venice has limited space for cars, and parking is restricted in many areas. Visitors often have to park in nearby cities and travel to the city center by boat. This is inconvenient and expensive.

Edinburgh, Scotland. The medieval Old Town of Edinburgh is a popular tourist destination, but the number of parking spaces is limited and expensive. Visitors often have to park in neighboring districts or in parking lots outside the city center, which can be far from the historic area.

Parking outside the historic city centers, also known as park-and-ride, is a strategy used by cities to reduce traffic congestion in high-traffic areas such as historic districts.

This approach involves creating parking spaces in remote locations, such as parking lots or garages, where visitors can park their cars and then use public transportation to reach the historic district. This helps reduce the number of cars on the roads in the city center, alleviating congestion and making it easier for visitors to navigate the area [1, 3, 12, 13].

2. Problem statement

The aim of the work is to develop a comprehensive model that will allow for simulating the movement of vehicles and the organization of the operation of interceptive surface parking lots.

3. Parking lot modeling software

Modern parking lot modeling is highly complex and can take into account a wide range of factors to simulate the behavior of vehicles in a parking lot, as well as generate detailed visualizations and insights that can be used to optimize the design, management, and utilization of the parking facility.

One of the most significant achievements in parking lot modeling is the use of computer vision and machine learning methods to automatically track and analyze the movement of vehicles within the parking lot. This allows for the generation of detailed visualizations of traffic flow and congestion, which can be used to identify bottlenecks, optimize traffic flow, and enhance overall parking efficiency [4].

Another key achievement in parking modeling is the use of big data analytics. Parking facilities generate a large volume of data, including information on the number of vehicles entering and exiting, the duration of each vehicle's stay, and the occupancy of each parking space. By analyzing this data, researchers and engineers can gain insights into parking usage patterns and optimize planning, management, and pricing strategies [2].

Moreover, the integration of Internet of Things (IoT) technologies, such as sensors and cameras, enables real-time monitoring and control of parking lots, allowing for dynamic pricing and efficient management [5].

Let's consider examples of parking modeling software.

TransModeler. This is traffic simulation software developed by Caliper Corporation. It is used to model and analyze transportation networks and systems, including highways, arterial roads, transit systems, as well as pedestrian and bicycle facilities. TransModeler can be utilized to assess the impact of various infrastructure projects and management strategies on traffic flows, travel time, and congestion [11]. A drawback of TransModeler is that it requires a high-performance computer to run and comes with a high license cost.

ParkCAD. This is a software tool developed by AutoTURN, designed for the planning and optimization of parking layouts and configurations. It is a computer-aided design (CAD) software that allows users to create and analyze parking lots with 2D and 3D visualizations, as well as make changes to the design and instantly view the results [8,18]. A drawback of ParkCAD is that it is complex and difficult to use. The software also requires a high-performance computer to operate, and the license cost is high.

AutoTURN. This software tool, developed by Transoft Solutions, is used for modeling vehicle movement and turning maneuvers in parking lots and other confined spaces. It is a vehicle trajectory analysis software that allows users to create and analyze vehicle movements when turning in 2D and 3D, including the ability to check clearance, conflict situations, and safety [19]. A drawback of AutoTURN is that it is complex and difficult to use for those who are not familiar with vehicle trajectory analysis.

The most well-known software for system modeling includes PTV VISSIM.

PTV VISSIM is traffic simulation software developed by PTV Group. It is used for modeling and analyzing the movement of vehicles and pedestrians in urban and transportation networks [7]. It is often employed to assess the impact of various transportation infrastructure projects and management strategies on traffic flows and congestion.

4. Main results

Let's consider the algorithm and software implementation that will allow modeling and studying the operation of parking lots.

For the development of the model, the Python programming language was used. Thanks to powerful libraries such as numpy and matplotlib, Python ensures efficient execution of numerical operations and data visualization.

The model algorithm takes into account the intensity of vehicle arrivals, the number of lanes, and the total parking capacity. The simulation model uses statistical methods for generating vehicle arrivals and determining the duration of their stay in the parking lot, which allows for realistic results and ensures the accuracy of the analysis.

4.1. Basic simulation model of parking lot operation

The basic simulation model assumes that:

- the vehicle arrival intensity at the parking lot in the simulation is 12 vehicles per hour;
- the parking lot has 4 lanes;
- the parking lot has 300 spaces;
- vehicles enter the parking lot randomly (according to a Gaussian distribution);
- the parking duration varies randomly between 1 and 8 hours.

Algorithm for Implementing the Basic Model of Parking Lot Operation.

Import the numpy library for numerical operations and matplotlib.pyplot for data visualization.

Define the ParkingSimulation class with the following parameters:

- num_lanes (the number of parking lot lanes),
- intensity (the vehicle arrival intensity),
- parking_capacity (the parking lot capacity).

Also, create a parking_lot list filled with None to represent the parking lot state, where each space can either be empty or contain the parking duration.

Define the method simulate(self, total_hours) that runs the main simulation.

Create an empty list daily_stats to store the daily parking lot occupancy statistics.

We iterate over the hours from 0 to total_hours–1:

- within the hourly loop, we iterate over each lane of the parking lot;
- we generate the number of arriving vehicles for this hour, using a normal distribution with a mean of self.intensity and a standard deviation of 2. This allows modeling the variation in the number of vehicles arriving at the parking lot;
- we limit the number of incoming vehicles to ensure it doesn't become negative and convert it to an integer;
- for each arriving vehicle, we call the self.park_car() method, which places the vehicle in an available parking spot;
- after the vehicles have arrived at each lane, we call the self.clear_parking() method, which reduces the parking duration for each occupied spot and frees it if the duration has expired;
- after processing all the hours, we count the number of occupied spots in the parking lot and add this number to the daily_stats list.

We output parking lot status information for each hour using the self.print_parking_status(hour) method.

We call the self.plot_hourly_stats(daily_stats) method, which visualizes the hourly parking occupancy statistics.

We define the park_car(self) method, which checks for available spaces in the parking lot, finds the first available spot, and parks a vehicle with a random parking duration.

We define the clear_parking(self) method, which reduces the parking duration for each occupied spot in the parking lot. If the duration reaches 0, the spot is freed.

We define the print_parking_status(self, hour) method, which counts the number of occupied spots in the parking lot and outputs the parking status information for a specific hour.

We define the plot_hourly_stats(self, hourly_stats) method, which creates a graph to visualize the hourly parking occupancy statistics.

In the if__name__==«__main__»: block, we define parameters for the simulation (number of parking lanes, car arrival intensity, parking lot capacity, and the total duration of the simulation in hours).

We create an object simulation of the ParkingSimulation class with the corresponding parameters and call the simulate(total_hours) method to start the simulation.

Thus, we obtain code for modeling the parking lot operation and creating a graph that demonstrates how the parking lot status changes throughout the day (figure 1).

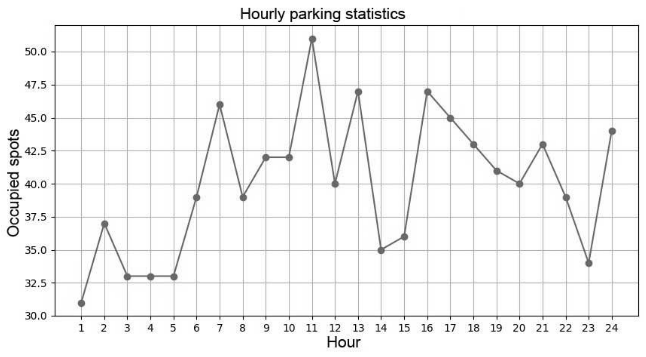


Fig. 1. Results of modeling the operation of a parking lot with the specified input parameters

4.2. Simulation model of the parking lot operation taking into account changes in traffic intensity during the day

We add to the obtained base model the variation in car arrival intensity at the parking lot during the simulation as follows:

- morning from 6 AM to 10 AM – arrival intensity of 12 cars per hour;
- daytime from 11 AM to 6 PM – arrival intensity of 20 cars per hour;
- evening from 7 PM to 11 PM – arrival intensity of 8 cars per hour;
- night from 12 PM to 5 AM – arrival intensity of 5 cars per hour.

Algorithm for the Parking Lot Model Implementation.

We import the necessary libraries: numpy for generating random numbers and matplotlib.pyplot for visualizing the data.

We define the ParkingSimulation class with the following parameters:

- num_lanes (number of parking lot lanes),
- parking_capacity (capacity of the parking lot).

We create a parking_lot list that represents the state of the parking lot, using None for available spots.

We define the simulate(self, total_hours) method, which runs the simulation over the specified number of hours.

We create an empty list daily_stats to store the parking lot occupancy statistics.

We iterate through the hours within the specified number of hours:

- determine the intensity for the current hour by calling the get_intensity(hour) method;
- for each parking lane, we perform the following steps:
  - generate the number of incoming vehicles arrivals using a normal distribution with a mean of intensity and a standard deviation of 2,
  - ensure that the number of incoming vehicles is non-negative, i.e., arrivals is not less than 0,
  - for each incoming vehicle, call the park_car() method, which parks the vehicle in an available spot in the parking lot,
  - call the clear_parking() method, which decreases the remaining time for each occupied spot and frees up the spot if the parking time has expired;
- calculate the number of occupied spots in the parking lot using a list generator and the sum function, and add this number to daily_stats;
- print the parking status for the current hour by calling the print_parking_status(hour) method;
- call the plot_hourly_stats(daily_stats) method to visualize the hourly statistics.

We define the method park_car(self), which parks a car in the first available spot in the parking lot.

We define the method clear_parking(self), which decreases the remaining time for each occupied parking spot. If the time expires, the spot is freed.

We define the method get_intensity(self, hour), which returns the appropriate vehicle arrival intensity based on the current hour.

We define the method print_parking_status(self, hour), which calculates and prints the parking lot status for the given hour.

We define the method plot_hourly_stats(self, hourly_stats), which visualizes the hourly parking occupancy statistics.

We introduce parameters for the simulation (the number of parking lanes, parking lot capacity, and the total duration of the simulation in hours).

We create an object simulation of the ParkingSimulation class with the specified parameters.

We run the simulation by calling the simulate(total_hours) method of the simulation object.

The graph of traffic intensity variation throughout the day is shown in figure 2.

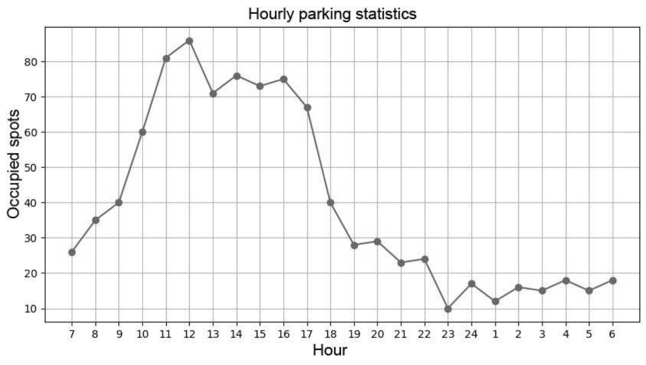


Fig. 2. The result of modeling the parking lot operation taking into account changes in traffic intensity during the day

4.3. Simulation model of the parking lot operation taking into account changes in traffic intensity hourly during the day

Algorithm for Implementing the Parking Lot Simulation Model with Hourly Traffic Intensity Variation over Seven Days of the Week.

1. Importing libraries: import the numpy library as np.
2. Defining the ParkingSimulation class: define the ParkingSimulation class for simulating the parking lot. The class has a constructor __init__, that initializes the number of lanes (num_lanes), parking lot capacity (parking_capacity), and a list for the parking lot (parking_lot).
3. Simulate method: define the simulate method, which runs the simulation for the specified number of days and stores the results in the provided results_list.
4. Simulation start: set the current_hour to 7 AM and begin iterating over the simulation days.
5. Iteration over hours: for each simulation day (7 days), iterate through the hours from current_hour to current_hour + 24. If the hour exceeds 24, subtract 24 from it.
6. Determining the day of the week and intensity: calculate the day of the week using the remainder of division by 7. Obtain the intensity based on the day of the week and hour from the get_intensity method.
7. Iteration over lanes: for each parking lane, iterate through a loop. Generate the number of vehicles (arrivals) using a normally distributed random variable with intensity, ensuring that the number is non-negative.
8. Parking vehicles: add vehicles to the parking lot using the park_car method.
9. Parking clearance: perform parking clearance (decreasing the remaining time at the parking spots) using the clear_parking method.
10. Displaying parking status: display information about the parking status, indicating the day of the week, hour, and the number of occupied spots using the print_parking_status method.
11. Park_car method: describe the park_car method, which allows parking vehicles in the parking lot if there are available spots.
12. Clear_parking method: describe the clear_parking method, which decreases the remaining time on occupied spots and frees them if the parking period ends.
13. Get_intensity method: describe the get_intensity method, which determines the intensity of vehicle traffic based on the day of the week and the hour. It uses baseline intensity values for different days of the week.
14. Print_parking_status method: define the print_parking_status method, which outputs the parking status with details such as the day of the week, the hour, and the number of occupied spots.
15. Get_day_name method: define the get_day_name method, which returns the name of the day of the week based on the day number.
16. If__name__==«__main__» block: begin the main part of the program, where simulation parameters are set: the number of lanes, parking lot capacity, total number of days for the simulation (total_days), and the number of simulations (num_simulations).
17. Object creation and simulation: for each simulation, create a simulation object of the ParkingSimulation class with the defined parameters. Then, call the simulate method on this object, passing the simulation results to simulation_results.
18. Saving results: add the simulation results to the overall data list.
19. Data visualization: for each day of the week, iterate through hours and simulations. Calculate the average occupancy of parking spots for each hour and display a graph of the average occupancy of the parking lot for each day of the week.
20. Displaying graphs: output graphs showing the average occupancy for each day of the week using the matplotlib library.

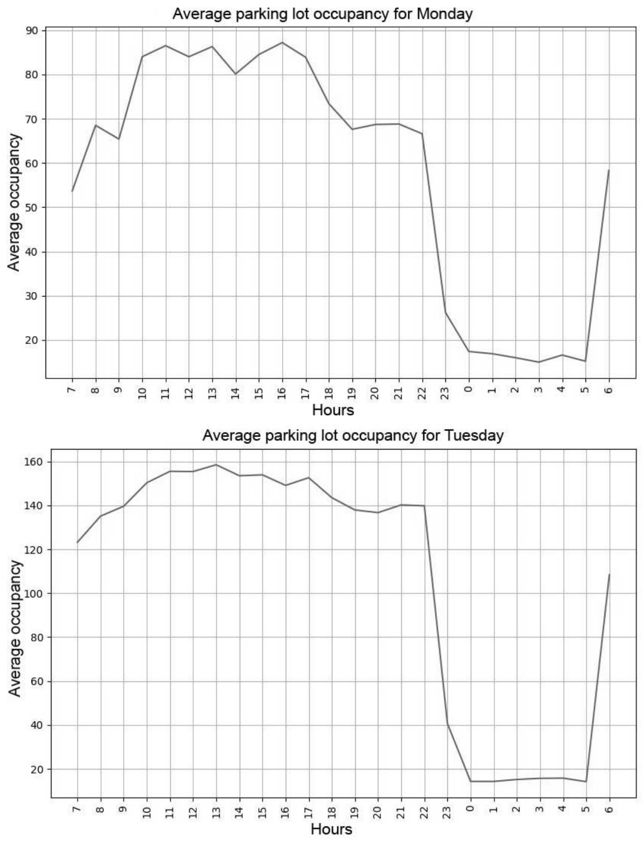


Fig. 3. Averaged results of ten simulations of the parking lot operation model, accounting for hourly traffic intensity changes from Monday to Tuesday

The averaged results of ten simulations of the parking lot operation model, accounting for hourly traffic intensity changes throughout the seven days of the week, are presented in Fig. 3–5.

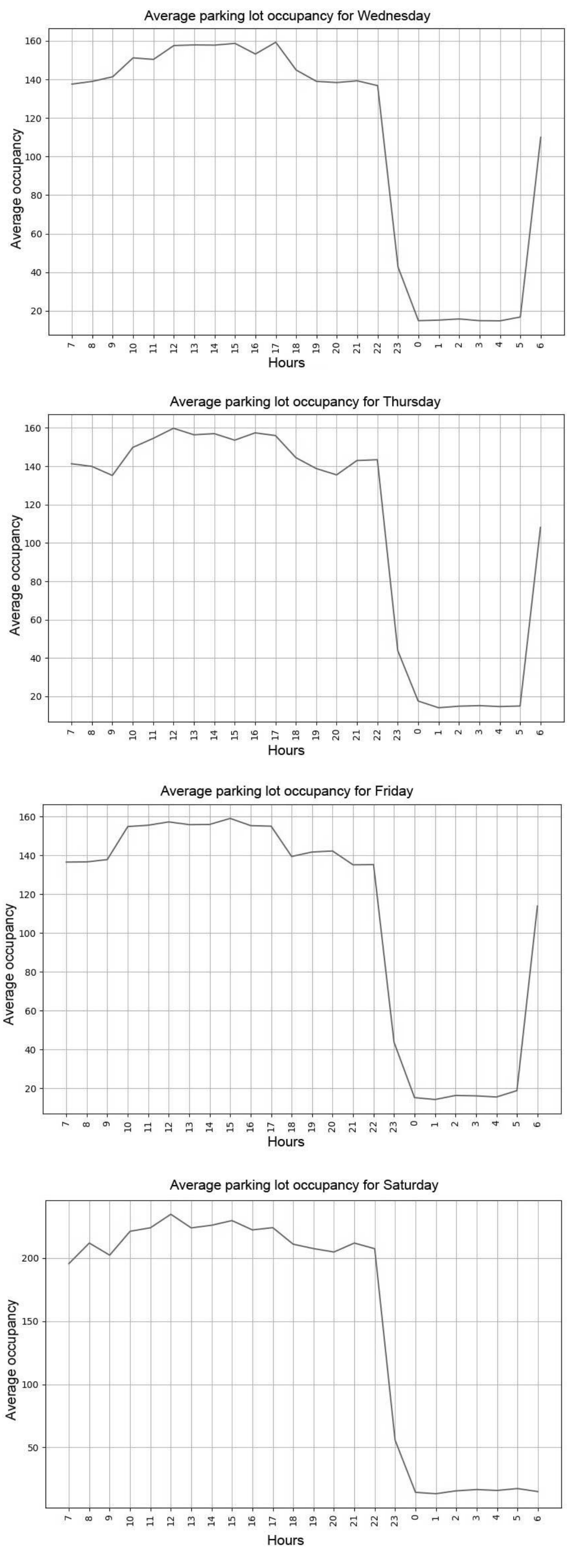


Fig. 4. Averaged results of ten simulations of the parking lot operation model, accounting for hourly traffic intensity changes from Wednesday to Saturday

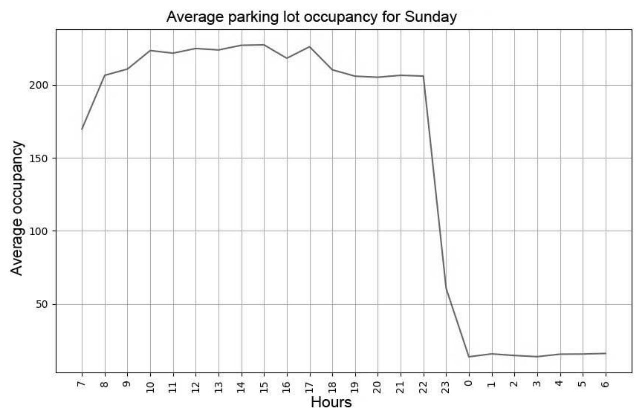


Fig. 5. Averaged results of ten simulations of the parking lot operation model, accounting for hourly traffic intensity changes during the Sunday

The input data for the simulation included the following parameters:

- parking capacity: parking_capacity = 300,
- simulation duration: total_days = 7,
- number of simulations: num_simulations = 10,
- base traffic intensities for different days of the week:
  Monday: 10 cars,
  Tuesday–Friday: 30 cars,
  Saturday–Sunday: 50 cars,
- hourly traffic intensity for car arrivals during the simulation:
  from 6:00 AM to 9:00 AM: +10 cars,
  from 10:00 AM to 5:00 PM: +15 cars,
  from 6:00 PM to 11:00 PM: +10 cars,
  from 12:00 PM to 6:00 AM: +5 cars.

5. Conclusion

The study proposes an algorithm and implements a simulation model of the operation of an intercept parking lot, developed using the Python programming language. This development allows for the investigation of the dynamic operation of parking lots, taking into account changes in vehicle traffic intensity throughout the day and week. It provides extensive opportunities for customizing parameters such as the number of lanes, parking lot capacity, vehicle arrival intensity, and the duration of their stay.

The key components of the model include simulating vehicle arrivals, calculating their duration of stay, and dynamically updating the occupancy status of parking spaces. The combination of these components enables the creation of realistic simulations that reflect various parking lot load scenarios. This approach facilitates a more detailed analysis, allows the identification of peak usage periods, uncovers bottlenecks in the parking process organization, and evaluates the effectiveness of management strategies.

The implementation of the model leverages the advantages of Python, such as ease of use, high development speed, support for the object-oriented approach, and the ability to integrate with powerful libraries for numerical computations and data visualization. Additionally, the model allows for repeated simulations with varying parameters to obtain statistically significant results necessary for making informed decisions.

The developed model can be valuable for urban planners, commercial parking lot owners, and researchers working on optimizing urban space utilization. Its integration with modern IoT solutions opens up opportunities for creating intelligent parking systems that can help reduce traffic congestion, improve driver convenience, and minimize the environmental impact of vehicles on city ecosystems.

Thus, the presented study has significant practical potential and can serve as a foundation for implementing innovative approaches to parking system management in urban environments.

References

[1] Garrett M. E.,Wachs M.: Transportation Planning on Trial. SAGE Publications, Inc. 1996.
[2] Huang X., Zhang J., Lin Y.: Intelligent Parking System Based on Big Data and Internet of Things. Applied Mathematics, Modeling and Computer Simulation 42, 2023, 733–742 [https://doi.org/10.3233/ATDE231013].
[3] Litman T.: Parking Management Best Practices. Routledge, New York 2005. [https://doi.org/10.4324/9781351179546].
[4] López A. M., et al.: Computer Vision in Vehicle Technology: Land, Sea & Air. First Edition. John Wiley & Sons Ltd. Published, West Sussex, England 2017. [https://doi.org/10.1002/9781118868065.refs].
[5] Oladimeji D., et al.: Smart Transportation: An Overview of Technologies and Applications. Sensors 23(8), 2023 [https://doi.org/10.3390/s23083880].
[6] Paidi V., et al.: Smart parking sensors, technologies and applications for open parking lots: a review. IET Intell. Transp. Syst. 12, 2018, 735–741 [https://doi.org/10.1049/iet-its.2017.0406].
[7] Prakash A., et al.: VISSIM Based Traffic Flow Simulation Analysis on Road Network. International Conference on Sustainable Goals in Materials, Energy and Environment – ICSMEE'24. E3S Web of Conferences 529, 2024 [https://doi.org/10.1051/e3sconf/202452903009].
[8] Sameer M., Hasan A.: How to use ParkCAD software - Basics. Conference: King Fahd University of Petroleum and Minerals 2023. [https://doi.org/10.13140/RG.2.2.13506.04809].
[9] Shoup D.: Parking and the City. Routledge, New York 2018. [https://doi.org/10.4324/9781351019668].
[10] Shoup D.: The High Cost of Free Parking. Journal of Planning Education and Research 17, 1997, 3–20.
[11] Ullah M. R., et al.: Vehicular Traffic Simulation Software: A Systematic Comparative Analysis. PakJET 4(1), 2021, 66–78 [https://doi.org/10.51846/vol4iss1pp66-78].
[12] Willson R.: Parking Management for Smart Growth. Willson R.: Parking and the City. Routledge, New York 2018, 222–227 [https://doi.org/10.4324/9781351019668-22].
[13] Zijlstra T., Vanoutrive T., Verhetsel A.: A meta-analysis of the effectiveness of park-and-ride facilities. European Journal of Transport and Infrastructure Research 15(4), 2015, 597–612 [https://doi.org/10.18757/ejtir.2015.15.4.3099].
[14] [https://europarkingservices.com/park-and-ride-system/] (available: 14.01.2025).
[15] [https://imegcorp.com/services/parking-planning-design/] (available: 14.01.2025).
[16] [https://parisjetaime.com/eng/convention/article/coach-traffic-and-parking-a1289] (available: 14.01.2025).
[17] [https://www.mrlodge.com/information-for-tenants/parking-munich] (available: 14.01.2025).
[18] [https://www.transoftsolutions.com/civil-and-transportation/software/site-and-parking-design/parkcad/] (available: 14.01.2025).
[19] [https://www.transoftsolutions.com/civil-and-transportation/software/swept-path-analysis/autoturn/] (available: 14.01.2025).

Ph.D. Larysa Gumeniuk
e-mail: l.gumeniuk@lntu.edu.ua

Lutsk National Technical University, Ph.D. (tech.), Department of Automation and Computer-Integrated Technologies.
Research interests: modeling of reliability and safety of the automated control systems.

https://orcid.org/0000-0002-7678-7060

Ph.D. Oleksandr Reshetylo
e-mail: o.reshetylo@lntu.edu.ua

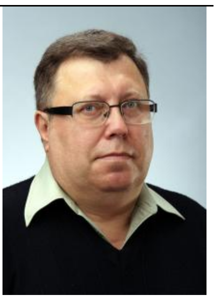
Lutsk National Technical University, Ph.D. (tech.), Department of Automation and Computer-Integrated Technologies.
Research interests: programming, computer-integrated technologies, electronic and intelligent vehicle systems.

https://orcid.org/0000-0003-2555-4205

Ph.D. Volodymyr Lotysh
e-mail: auvp@lntu.edu.ua

Lutsk National Technical University, PhD. (tech.).
Research interests: open-source software applied for simulations of problems using distributed platforms.

https://orcid.org/0000-0003-0899-8015



M.Sc. Yuriy Syrota
e-mail: auvp@lntu.edu.ua

Yuriy Syrota received his Master's degree in Automation, Computer-Integrated Technologies and Robotics from Lutsk National Technical University. His main areas of research are modeling and computer-integrated technologies.

https://orcid.org/0009-0001-9429-2854

Ph.D. Pavlo Humeniuk
e-mail: p.gumeniuk@lntu.edu.ua

Lutsk National Technical University, Ph.D. (tech.), Department of Automation and Computer-Integrated Technologies.
Research interests: programming, robotics.

https://orcid.org/0000-0002-6251-8548