University of Warsaw
Faculty of Mathematics, Informatics, and Mechanics

Paweł Gora

Metaheuristics in Optimization of Complex Processes

PhD dissertation

Supervisor

Prof. Dr Hab. Eng. Andrzej Skowron

Systems Research Institute
Polish Academy of Sciences

June 2023

Author's declaration:
aware of legal responsibility I hereby declare that I have written this dissertation myself and all the contents of the dissertation have been obtained by legal means.

June 14, 2023 .....
date *Paweł Gora*

Supervisor's declaration:
the dissertation is ready to be reviewed

June 14, 2023 .....
date *Prof. Dr Hab. Eng. Andrzej Skowron*

Contents

Abstract

The optimization of complex processes plays a critical role in several domains, including engineering, traffic control, and finance. However, the inherent complexity and nonlinearity of these processes pose significant challenges to traditional optimization techniques. This thesis proposes and examines a technique based on the integration of computer simulations, metaheuristics (i.e., high-level optimization algorithms that explore large spaces of candidate solutions), and surrogate models (i.e., mathematical approximations of more complex models) to address this problem.

First, the thesis outlines the motivation behind the proposed research and summarizes its objectives. The main objective of this research was to gain a deeper understanding of the nature of complex processes and to take a step towards the development of universal, scalable methods for analyzing and optimizing complex processes using metaheuristics. This was done in the presented research on the basis of considering two complex processes, but some general and theoretical considerations regarding complex processes are included in this thesis too. The first and main complex process studied was urban road traffic controlled by traffic signals. The aim was to optimize selected important traffic characteristics using metaheuristics and to investigate which of them give the best performance in terms of quality of solutions and scalability, including time of computations. The second complex process studied was cancer evolution under radiotherapy treatment. The goal was to minimize the number of cancer cells after treatment by controlling the size and timing of the radiotherapy doses.

After the introduction, the thesis summarizes related works, including the state of the art in understanding, modeling, and optimizing complex systems and processes. The main metaheuristics that were used in the experiments described in this thesis are also presented along with their pseudocodes.

Next, the thesis discusses the author's original contribution, including the Traffic Simulation Framework software (TSF), which was one of the main tools used in the experiments, as well as the general methodology for optimizing complex processes using metaheuristics, computer simulations, and

surrogate models based on machine learning.

The thesis contains theoretical considerations (including the introduction of several new formalisms and definitions) as well as descriptions of experiments aimed at testing the proposed methods in practice. Many series of experiments are described, they were carried out with different metaheuristics, surrogate models, and configurations for the two main use cases (traffic signal control and optimization of radiotherapy for cancer treatment). The thesis includes the descriptions of the setup of the experiments, as well as the presentations and discussions of their results. This is followed by a discussion of the identified limitations of the introduced method, potential ways to overcome them, and possible extensions to enable real-world applications of the developed techniques and tools.

Finally, the conclusions of the thesis summarize the achieved results and confront them with the initial research objectives. It turned out that the combination of population-based metaheuristics with surrogate models based on machine learning (especially techniques based on LightGBM and proposed architectures of sparse graph neural networks) gave the best results in the experiments conducted, and the method developed and tested by the author of this thesis proved successful for both studied complex processes.

Although there are still many obstacles on the way to a successful application of the developed methods in real-world scenarios, the achieved results are promising. It was possible to overcome the initial computational challenges, and the achieved results opened many new research directions, which led to the establishment of a research group TensorCell. Moreover, the research covered in this thesis has been summarized and published in 25 research publications and awarded with several prestigious prizes. There are already ongoing discussions about the applications of the considered method in real traffic management systems and the use of the Traffic Simulation Framework in the recently established SmartCity Lab in Chełm (Poland). Since the TSF software has been developed using the road network and traffic data from the measurements in Warsaw, there is also a potential to run some pilot projects and plan the possible implementations in this city. There is also interest from other scientists to test the developed method for optimizing complex processes in a new field - materials science.

Most of the datasets and some programs used in the experiments have been made publicly available to facilitate further studies.

Streszczenie

Optymalizacja procesów złożonych odgrywa kluczową rolę w wielu dziedzinach, w tym w inżynierii, kontroli ruchu i finansach. Nieodłączna złożoność, jak sama nazwa wskazuje, i nieliniowość tych procesów stanowi poważne wyzwanie dla tradycyjnych technik optymalizacji. W celu rozwiązania tego problemu w niniejszej pracy zaproponowano i zbadano technikę opartą na integracji symulacji komputerowych, metaheurystyk (algorytmów optymalizacji przeszukujących duże przestrzenie potencjalnych rozwiązań) i modeli surogatywnych (matematycznych przybliżeń bardziej złożonych modeli).

Początek pracy przedstawia motywację proponowanych badań i podsumowuje ich cele, m.in. głębsze zrozumienie natury procesów złożonych oraz opracowanie uniwersalnych, skalowalnych metod ich analizy i optymalizacji z wykorzystaniem metaheurystyk. W badaniach skupiono się na dwóch procesach złożonych, ale w niniejszej pracy zawarte są również bardziej ogólne rozważania dotyczące procesów złożonych. Pierwszym badanym procesem złożonym był ruch drogowy w mieście sterowany sygnalizacją świetlną, a celem była optymalizacja niektórych ważnych charakterystyk ruchu z wykorzystaniem metaheurystyk i zbadanie, które z nich są najlepsze pod względem jakości rozwiązań i skalowalności, w tym czasu obliczeń. Drugim badanym procesem złożonym była ewolucja nowotworu podczas radioterapii, a celem było zminimalizowanie liczby komórek nowotworowych po leczeniu poprzez kontrolowanie wielkości dawek i czasu podawania radioterapii.

Po wprowadzeniu praca podsumowuje stan wiedzy w powiązanych z rozprawą obszarach, w szczególności w obszarze modelowania i optymalizacji procesów złożonych. Przedstawione są również główne metaheurystyki, które zostały użyte w prezentowanych w tej pracy eksperymentach, wraz z ich pseudokodem.

Następnie w pracy przedstawiono oryginalny wkład autora, w tym oprogramowanie Traffic Simulation Framework (TSF), które było jednym z głównych narzędzi wykorzystywanych w eksperymentach, oraz ogólną metodologię optymalizacji procesów złożonych z wykorzystaniem symulacji komputerowych, metaheurystyk i modeli surogatywnych bazujących na uczeniu maszyno-

wym.

W pracy zawarte są zarówno rozważania teoretyczne, m.in. wprowadzenie kilku nowych formalizmów i definicji, jak i opisy eksperymentów mających na celu sprawdzenie jakości proponowanych metod w praktyce. Opisano wiele serii eksperymentów z różnymi metaheurystykami, modelami surogatywnymi i konfiguracjami dla dwóch głównych przypadków użycia, tzn. sterowania sygnalizacją świetlną i optymalizacji radioterapii w leczeniu nowotworu. Praca zawiera opisy konfiguracji eksperymentów, a także przedstawia i omawia ich wyniki. Następnie przedstawione są zidentyfikowane ograniczenia opracowanych metod, a także potencjalne sposoby ich pokonania oraz możliwe rozszerzenia tych metod zmierzające do ich praktyczne zastosowania.

W podsumowaniu pracy uzyskane wyniki skonfrontowane są z celami badawczymi. Okazało się, że połączenie metaheurystyk populacyjnych z modelami surogatywnymi bazującymi na uczeniu maszynowym, przede wszystkim modelami LightGBM oraz nowymi architekturami rzadkich grafowych sieci neuronowych zaproponowanych w ramach badań, dało najlepsze wyniki w przeprowadzonych eksperymentach. Ponadto, opracowana i przetestowana przez autora pracy metoda okazała się skuteczna dla obu badanych procesów złożonych.

Chociaż wciąż istnieje wiele przeszkód na drodze do pomyślnego zastosowania opracowanej metody w rzeczywistych scenariuszach, osiągnięte wyniki są obiecujące. Udało się przezwyciężyć początkowe wyzwania obliczeniowe, a uzyskane wyniki otworzyły wiele nowych kierunków badawczych, co doprowadziło m.in. do powstania grupy badawczej TensorCell. Ponadto, badania omówione w niniejszej rozprawie zostały podsumowane i opublikowane łącznie w 25 artykułach naukowych i nagrodzone kilkoma prestiżowymi wyróżnieniami. Trwają też już dyskusje na temat zastosowań opracowanych w ramach przedstawionych badań metod w rzeczywistych systemach zarządzania ruchem oraz wykorzystania programu Traffic Simulation Framework w niedawno powstałym SmartCity Lab w Chełmie. Ponieważ oprogramowanie TSF zostało opracowane na podstawie sieci drogowej i danych o ruchu z pomiarów w Warszawie, istnieje również potencjał do przeprowadzenia pilotażowych projektów i zaplanowania możliwych wdrożeń w tym mieście. Pojawiło się też zainteresowanie ze strony innych naukowców przetestowaniem opracowanej metody w zupełnie nowym obszarze - inżynierii materiałowej.

Większość zbiorów danych i niektóre programy użyte w eksperymentach zostały udostępnione publicznie, aby zapewnić odtwarzalność wyników i ułatwić dalsze badania.

Acknowledgements

The completion of this thesis is the culmination of years of passion and work, but also the support of countless people. Even though one of the goals of this thesis is to optimize complex processes, in particular, to minimize travel times (in the case of urban road traffic), it is also important to remember that very often the journey itself is at least as important as reaching the destination. For me, the time spent on conducting the presented research has been an incredible journey, full of enthusiasm, passion, growth, and education, but also of solving difficult problems, making tough decisions, and taking on formidable challenges. It was an enriching experience that shaped my intellectual and personal development.

During this journey, I have met fantastic people, many of whom have had a great influence on my way of thinking, perceiving, and understanding the world, and consequently on my research. Some of them became my collaborators or co-authors of joint scientific publications, others expanded my knowledge thanks to inspiring discussions. It is infeasible to mention all the people and institutions that have had a positive impact on my research or have helped me in some way during the journey - there were simply too many. However, I would like to express my gratitude to some of them.

First and foremost, I would like to sincerely thank my supervisor, Professor Andrzej Skowron, for his invaluable guidance, unwavering support, and expert advice throughout the entire research process. His dedication, encouragement, and scholarly insights have been instrumental in shaping my research direction and academic growth.

Furthermore, I would like to thank all my research collaborators, especially the co-authors of joint scientific publications, members of the TensorCell research group, as well as colleagues, peers and students from the University of Warsaw and other institutions: Paweł Adacha, Mateusz Adamczyk, Simon Angus, Constantinos Antoniou, Rafał Banaś, Dominika Bankiewicz, Jan Bazan, Julia Bazińska, Marek Bardoński, Paweł Betliński, Mateusz Biesiadowski, Patryk Binkowski, Mateusz Błajda, Dominik Bogucki, Grzegorz Bokota, Latif Bölüm, Michał Borowski, Dawid Borys, Paula Botella, Paweł

Brach, Maciej Brzeski, Damian Burczyk, Sara Carvajal Querol, Wojciech Chmiel, Krzysztof Choromański, Andrzej Cielecki, Francesco Collabrese, Marek Cygan, Andrzej Czyżewski, Nima Dadashzadeh, Krzysztof Diks, Arkadiusz Drabicki, Hubert Dryja, Tomasz Dybicz, Michał Filipiuk, Paweł Fuławka, Yannis Gkourfas, Piotr Golach, Antoni Goldstein, Grzegorz Góra, Paweł Gromek, Damian Hajduk, Eli Halych, Kamil Herba, Agnieszka Ilnicka, Faqhrul Islam, Aleksander Jankowski, Andrzej Jankowski, Krzysztof Jankowski, Andrzej Janusz, Janusz Jabłonowski, Ewelina Kamińska, Tomasz Kamiński, Rafał Karczewski, Michał Kardas, Katarzyna Karnas, Izabela Karsznia, Christos Katrakazas, Wojciech Kaźmierczak, Arkadiusz Klemenko, Adrian Kochański, Dawid Kopczyk, Piotr Kowalkowski, Krystian Król, Piotr Krukowski, Mikołaj Kruszewski, Rafał Kucharski, Magdalena Kukawska, Karol Kurach, Michał Kutwin, Mateusz Lewandowski, Kim Chuan Lim, Marek Litwin, Olaf Łobożewicz, Bartosz Łoś, Mateusz Macias, Marcin Małogrosz, Artur Matyjasek, Łukasz Mądry, Aleksander Molak, Magdalena Molenda, Marcin Możejko, Rahul Nair, William Najarian, Hung Son Nguyen, Sinh Hoa Nguyen, Trung Tuan Nguyen, Michał Niezgoda, Piotr Olszewski, Maria Oparka, Piotr Ostaszewski, Wojciech Ozimek, Przemysław Pardel, Zofia Partyka, Joanna Paszkowska, Mariusz Patyk, Przemysław Perkowski, Krzysztof Perlicki, Łukasz Piekarski, Monika Piotrowska, Szymon Płotka, Rafał Pragacz, Tomasz Przeździęk, Przemysław Przybyszewski, Sebastian Purtak, Inga Rüb, Paweł Senchanka, Norbert Siwek, Łukasz Skowronek, Martyna Sosnowska, Grzegorz Sroka, Jacek Sroka, Anna Staśkiewicz, Sebastian Stawicki, Marek Stawowy, Filip Stefaniuk, Michał Stolarczyk, Agnieszka Strzałka, Wojciech Suchorzewski, Mateusz Susik, Andrzej Szarata, Marcin Szczuka, Anna Szczurek, Witold Szejgis, Miron Szewczyk, Maciej Szymczyk, Dominik Ślęzak, Joanna Świetlicka, Marian Tracz, Tomasz Trzciński, Mikołaj Walczak, Anna Warno, Piotr Wasilewski, Krzysztof Wielocha, Grzegorz Wilczewski, Michał Witas, Marcin Wojnarski, Piotr Wójcik, Mateusz Zakrzewski, Demetrios Zeinalipour, Damian Zięba, Karolina Zygmunt, Szymon Zwara, Maciej Zwoliński.

I am grateful for their stimulating discussions, intellectual engagement, and collaborative spirit. Their input and shared experiences have enriched my research journey and helped me overcome various challenges along the way. Without some of them, some of the results presented in this thesis would have been much more difficult or even impossible to achieve.

I would also like to thank my Family and Friends for their support. Their unwavering belief in me, encouragement, and constant motivation provided the strength and inspiration needed to persevere through the challenging phases of my research work. I am truly fortunate to have them by my side. Here, my special thanks go to Grzegorz Jamróz, Agata Kawa-Sapyta, and Justyna Zawalska who reviewed near-final versions of this thesis and shared

their remarks and observations. It helped me improve the thesis and was an invaluable support at the end.

Finally, I would like to thank all the organizations and institutions that supported my research. I am grateful to the authorities and the officials of the City of Warsaw, especially the Administration of Urban Roads in Warsaw, for sharing the traffic data and origin-destination matrices that helped me develop a better traffic model and bring the Traffic Simulation Framework tool that I have been developing over many years closer to the real-world applications. It is also important to mention that some parts of the research presented in this thesis have been (partially) supported by various research grants:

- "GLAD - Green Last Mile Delivery: a more sustainable way for food home delivery tailored to consumer needs" (grant no 19147, funded by the European Institute for Innovation & Technology (EIT));

- "TOTAL: Technology transfer between modern algorithmic paradigms" (grant no 677651 funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program);

- "INZNAK: Intelligent Road Signs with V2X Interface for Adaptive Traffic Controlling"(grant no. OIR.04.01.04-0089/16, funded by the National Centre for Research and Development);

- "Wider Impacts and Scenario Evaluations of Autonomous and Connected Transport" (COST ACTION CA16222, funded by the Horizon 2020 Framework Programme of the European Union);

- "Multi-cloud Execution-ware for Large-scale Optimized Data-Intensive Computing" (grant no. 731664, funded by the Horizon 2020 Framework Programme of the European Union);

- "Parking places at the Rest and Service Areas" (grant no. DZP/RID-I-44/8/NCBR/2016, funded by the National Centre for Research and Development);

- "Perception Based Computing" (grant no. 2011/01/D/ST6/06981, funded by the National Science Center);

- "Interactive Computations in Layered Granular Network" (grant HOMING PLUS/2011-3/12, funded by the "Foundation for Polish Science");

- "Interactive Process Mining" (grant no. N N516 368334, funded by the Ministry of Science and Higher Education);
- "Interactive computations in knowledge discovery from complex data source (grant no. N N516 077837, funded by the "Ministry of Science and Higher Education").

Even though I was not a direct recipient of those grants, some aspects of the research presented in this thesis intersected with the scopes of these projects, and my participation in those grants enhanced the quality of my work, helped in achieving some results, and played a pivot role in shaping and enriching my research journey.

Last but not least, I would like to thank the institutions that allowed me to use their computational infrastructure, which was crucial for carrying out the experiments described in this thesis. Besides the access to the machines at the Faculty of Mathematics, Informatics, and Mechanics of the University of Warsaw, I also received several computational grants, including:

- "AI for Earth" from Microsoft;
- "AWS Cloud Credits for Research" from Amazon;
- "Google Cloud Research Credits" from Google;
- Two computational grants from the Interdisciplinary Centre for Mathematical and Computational Modelling at the University of Warsaw:
  - "Analyzing road traffic with connected and autonomous vehicles";
  - "Optymalizacja leczenia nowotworu przy pomocy algorytmów uczenia maszynowego i metaheurystyk".

Some experiments were also carried out using "PEGAZ" computing cluster at the Interdisciplinary Centre for Computer Modelling at the University of Rzeszów, or using an NVIDIA server.

Chapter 1

Introduction

1.1 Research motivation

The study of complex systems in a unified framework has become recognized as a new, interdisciplinary scientific discipline [16, 305]. It deals with systems composed of many interacting objects (generating processes), hardly separated from the environment. These objects and their interaction give rise to dynamic, complicated, non-linear, and often chaotic behavior. Such systems are also referred to as *complex systems*, while their evolution over time is referred to as *complex processes*.

As stated in [251]:

> *Complex systems science changes the way we think about science and its role in society. It goes beyond the traditional, reductionist approach of focusing on the parts of a system, to integrating the network of relationships within and between systems. These relationships produce the "emergent" behaviors we see in all physical, biological, social, economic and technological systems. This approach allows researchers to address questions once considered to be outside the reach of science, including human behavior, social interactions and the consequences of policies and decisions of our society.*

The concept of complex systems and processes can include abstract structures (e.g., cellular automata) as well as physical structures that can occur in many domains, from the microscopic (describing the state of each individual object) to the macroscopic (describing the global state of the system) level [95]. Obtaining a comprehensive understanding of the functioning of a complex process and identifying methods to optimize its evolution can be highly

beneficial. However, it is usually extremely difficult to control the behavior of such systems because of the number of components and the nature of the interactions, which introduce properties such as non-linearity, sensitive dependence on initial conditions, and computational irreducibility. In addition, the openness of the system allows interactions with other objects, which can lead to non-determinism. The presence of such features usually indicates that the behavior of the corresponding system may be unpredictable and difficult to understand and control. At the same time, the control of complex processes is very important from a practical point of view. Therefore, it is important to study this area and to try to develop universal methods for the analysis and optimization of complex processes.

The difficulty of analysis and optimization of complex processes have been well characterized by Fredericks Brooks [43]:

> *Mathematics and the physical sciences made great strides for several centuries by constructing simplified models of complex phenomena, deriving properties from the models, and verifying those properties experimentally. This worked because the complexities ignored in the models were not the essential properties of the phenomena. It does not work when the complexities are the essence.*

As presented in Section 2.1, there are already several approaches aiming to optimize complex processes. In this dissertation, the focus is mainly on exploring one of them - the use of metaheuristics (embedded in abstract space). Metaheuristics are high-level problem-independent strategies that guide the process of searching for good solutions to optimization problems, their goal is to efficiently explore the search space in order to find near-optimal solutions [314]. This term was coined by Fred Glover in 1986 and is also often used to refer to a problem-specific implementation of a heuristic optimization algorithm according to the guidelines expressed in a metaheuristic framework [314].

As explained in Section 1.2, one of the goals of this research is to develop universal methods for optimizing complex processes. However, in spite of many similarities, there are fundamental differences between different complex systems, and the general mathematical apparatus is not mature enough to study all possible systems at once. Thus, this thesis focuses on 2 complex processes:

1. Vehicular traffic in cities (described in Section 4.2): a process composed of many cars/drivers aiming to reach a destination safely and as soon as possible. Cars have to share infrastructure, so they interact, which

may lead to traffic jams or traffic accidents. There exist many ways to optimize traffic, but this thesis focuses on traffic signal control.

2. The process of cancer evolution under radiotherapy treatment (described in Section 4.3). In this case, living cells interact according to specific rules, and some of them may become cancer cells. Radiotherapy can affect this process and, eventually, reduce the number of cancer cells.

Optimizing both processes is very important for society and can help solve major economic and civilizational problems such as traffic congestion, car accidents, as well as suffering and death from cancer.

Inefficiency in transport can cause travel delays, stress for drivers, noise, increased air pollution, fuel, and energy consumption, problems in organizing public transport and detours, etc. It is estimated that drivers in the 7 largest Polish cities lose about 3.3 billion PLN a year due to traffic jams [75]. The situation is similar in other countries, e.g., drivers in the USA lose 5.5 billion hours and 2.9 billion gallons of fuel every year [265], and it is predicted that the cost of traffic jams will amount to 293.1 billion USD by 2030 (an almost 50% increase compared to 2013) [50]. In addition, air pollution is estimated to reduce life expectancy by more than 1 year [9], and road traffic is a major contributor to air pollution in cities where a large proportion of the population lives [91].

It is estimated that proper traffic signal control can lead to a reduction in traffic delays of $15-40\%$, a reduction in travel time of up to 25%, a reduction in stops of $10-40\%$, a reduction in fuel consumption of up to 10% and a reduction in harmful emissions (carbon monoxide, nitrogen oxides, volatile organic compounds) of up to 22% [189].

It is therefore important to conduct research to understand the phenomenon of congestion and how to improve traffic flow. There can be many reasons for traffic jams, but in principle, they always occur when the demand for road infrastructure is greater than the supply, i.e., when there are too many vehicles compared to the capacity of the roads.

There exist many ways of reducing demand and increasing supply. In the case of reducing the demand for road infrastructure, scientists, traffic engineers, and urban planners consider approaches such as car-sharing, bike-sharing, and van-pooling services, organizing public transport, building bicycle paths, introducing charges for parking and entering some areas, or banning access to some areas. All of these approaches can reduce the use of private cars and road infrastructure, but they all have some drawbacks, e.g., in some cases, travel times or costs may be higher.

That is why it is also important to look at how the supply side can be improved. The natural and one of the most appealing methods is to build more roads or increase the capacity of existing roads. It is reasonable and important to build new road infrastructure, but the Braess paradox [40] shows that in some cases it can have the opposite effect. Also, the Lewis-Mogridge position implies that as more roads are built, more traffic will consequently fill these roads [201, 240].

Another way to improve the supply side and increase throughput is to optimize traffic signal timing. This is the approach investigated in this thesis. It also has some drawbacks, e.g., similar to the Lewis-Mogridge position [201, 240], improving throughput at some junctions may lead in the long run to more travelers wanting to use that junction. Similarly, the very presence of traffic signals, although important from the road safety and synchronization perspective, can lead to inefficiencies, for example where demand is relatively low. Interestingly, there are ideas that, at some point in the future, traffic signals may no longer be needed, as traffic synchronization could be achieved thanks to wireless communication between autonomous vehicles [323]. At the moment, however, it does not look like traffic signals will be replaced by other means of traffic synchronization in the near future. On the contrary, more and more intelligent traffic management systems are being installed around the world, and more and more research is being done in the field of traffic signal control. Traffic signal control is also one of the means of traffic management that can be directly adapted to the current traffic situation and at the same time have a direct and immediate effect on traffic. It also has a direct effect on some traffic characteristics, such as delays, travel times, or waiting times at red signals, and can be carried out after travelers have made decisions about departure times, means of transport, and routes. Hence, the optimization of the complex process of urban traffic has been selected as the primary research topic for this thesis, with a focus on the control of traffic signals.

In terms of traffic optimization, an interesting and important question is: "What exactly should be optimized?". Many traffic characteristics could be considered, such as travel time, delay, waiting time at red signals, number of waiting vehicles, throughput, etc. These metrics only relate to traffic efficiency, but another important factor that could be considered is traffic safety. Indeed, the interaction of road users can lead not only to congestion but also to road accidents. It is estimated that every year around 1.3 million people die and more than 20 million are injured worldwide as a result of traffic crashes [359]. It is also estimated that car accidents cost the US economy 340 billion USD each year [28]. Therefore, researchers also think about how to reduce the number of accidents and their negative effects to move society

towards "Vision Zero" [341] in which there are no victims of road accidents at all. The introduction of connected and autonomous vehicles (CAVs) is one of the expected ways to get closer to this goal. Co-operating CAVs will also form a complex system that can be optimized for performance and safety by manipulating the parameter values of their driving and communication algorithms. However, even though the author of this thesis has also carried out some research into traffic modeling of CAVs and their impact on traffic safety and efficiency under different conditions [132, 128], these aspects remain beyond the scope of this dissertation. Instead, the focus was on solving the so-called *Traffic Signal Setting problem* (defined in Section 8.1) in which the primary optimized metric was the total time of waiting at red signals in a given urban area during a certain period. The selection of such a metric was motivated by the fact that it is directly influenced by the settings of traffic signals, contrary to other popular metrics like travel times or delays that are also influenced by the speeds of travel between intersections with signals.

Another complex process that was investigated in this dissertation is the process of cancer evolution using radiotherapy treatment. According to the World Health Organization (WHO), cancer is a leading cause of death worldwide, accounting for nearly 10 million deaths in 2020, or nearly one in six deaths [358]. In addition, many cancers are curable if detected early and treated effectively. One of the most common treatments for cancer is radiotherapy. As the interactions between cells in the living organism and the effects of radiotherapy doses on them are complex, the process of cancer development in the presence of radiotherapy can also be considered a complex process that can be optimized.

As presented in Chapter 8, the study's optimization problems can be defined as combinatorial optimization problems. For such problems (especially NP-hard), one of the natural approaches is applying metaheuristics [30], which do not guarantee that the optimal solution will be found, but have the potential to find near-optimal solutions. However, contrary to typical combinatorial optimization problems, like the Traveling Salesman Problem [198], in the case of complex processes, the evolution of qualities of possible solutions can be time-consuming, due to the aforementioned property of complex systems, computational irreducibility [163], which implies that it is impossible to obtain the exact future state of a complex system in a simpler way than by observing the evolution of the system or by running an exact simulation. It is especially hindering for metaheuristics which usually require the evaluation of a significant number of different candidates in order to find suitable solutions.

In some preliminary studies by the author of this thesis on solving the Traffic Signal Setting problem (defined in Section 8.1), metaheuristics (espe-

cially genetic algorithms) gave promising results [121], but the time of computations was significant, ruling out traffic signal control in real-time, and even severely hindering the process of running experiments and validating different metaheuristics with different settings. This motivated the author of the thesis to further investigate how metaheuristics can be applied to solve the Traffic Signal Setting problem and, potentially, optimize other complex processes, in a reasonable computational time. The primary focus was still on the Traffic Signal Setting problem using metaheuristics, but the natural research objective was to develop techniques that could be universal enough to be applied to the optimization of many other complex processes.

Distributing computations on high-performance computing (HPC) clusters and running experiments in cloud infrastructure did not provide sufficient acceleration, so it was necessary to look for other techniques. Eventually, the author of the thesis overcame the computational issues by constructing an optimization method based on the combination of metaheuristics with surrogate modeling using machine learning models. This led to the emergence of several new research questions, such as:

1. Which machine learning models are the best surrogate models for microscopic traffic simulations?

2. Which metaheuristics will produce the best results and are the most efficient in solving the Traffic Signal Setting problem when combined with surrogate models based on machine learning?

3. Is this technique universal enough to be applied to the optimization of other complex processes? If so, what are the best surrogate models and metaheuristics for a given complex process? Does it depend on the nature of the complex process?

Therefore, the author of this thesis tried to answer these questions and conducted many series of experiments that demonstrated the performance of various surrogate models and metaheuristics in solving the Traffic Signal Setting problem, and later later validated the introduced methodology for optimizing another complex process: cancer evolution under radiotherapy treatment.

1.2 Research objectives

The main objective of this research was to gain a deeper understanding of the nature of complex processes and to make a step toward developing uni-

versal, scalable methods for the analysis and optimization of complex processes using metaheuristics. This was done on the basis of considering two complex processes, but some more general and theoretical deliberations regarding complex processes are also included in this thesis (cf. Chapters 3 and 7). The first and primary complex process studied was urban road traffic controlled by traffic signals, and the aim was to optimize some important traffic characteristics (mainly the total waiting time of vehicles at red signals in a given area and time period) using metaheuristics and investigate which of them give the best performance in terms of the quality of solutions and scalability, including time of computations. The second studied complex process was the cancer evolution under radiotherapy treatment, where the goal was to minimize the number of cancer cells after treatment by controlling the size and timing of the radiotherapy doses.

The initial experiments showed that combining metaheuristics with simulation models can result in approaches that produce reasonably good settings but are computationally expensive due to the need to evaluate a large number of candidate solutions using simulation models. The time complexity made it impossible to control the traffic lights in real-time. In fact, it made it very difficult to run experiments and validate different metaheuristics with different settings, even when the computations were run on a cluster of many CPUs. The large computational complexity of realistic simulations, their validation, and their use as evaluators of fitness functions for metaheuristics became a serious challenge for the author of the thesis that had to be solved in order to test many optimization algorithms with different settings.

Therefore, the next research objective was to investigate whether metaheuristics can be applied to optimize complex processes in a reasonable computational time. The efforts to speed up computations led to the development of an optimization methodology based on the combination of metaheuristics and surrogate modeling using machine learning.

This method gave promising results in solving the Traffic Signal Setting problem, so the next research objective was to investigate which machine learning models are the best surrogate models for microscopic traffic simulations, and which metaheuristics can produce the best results and are the most efficient in solving the Traffic Signal Setting problem when combined with surrogate models based on machine learning.

The developed method seemed to be universal enough to be applied to the optimization of other complex processes, so the next research objective was to test it for another use case: optimizing cancer evolution under radiotherapy treatment, where the goal is to minimize the number of cancer cells after treatment by controlling the size and timing of the radiotherapy doses.

1.3 Research methodology

The methods for studying the optimization of complex processes developed within the presented research were investigated using simulation environments, the so-called "digital twins" of the considered complex processes.

For traffic signal control, the primary tool was the Traffic Simulation Framework (TSF) software developed by the author of this thesis. The underlying traffic model was also proposed by the author as an extension of the Nagel-Schreckenberg model [246] to the case of realistic road networks. Both the model and its implementation are described in [125, 123] and in Chapter 5 of this thesis.

For cancer growth, another simulation tool was developed based on a model described in [7]. Its efficient implementation was proposed, designed, also supervised by the author of this dissertation, but carried out by a group of students within their Bachelor thesis project [15].

Later in the study, these simulation tools for investigated complex processes were applied as evaluators of the objective functions: values of some simulation parameters were passed as the input, and the simulators computed the value of the objective function corresponding to the given metrics (e.g., the total times of waiting at red signals, the total times of travel, the total number of cancer cells, etc.). Then, the selected metrics were evaluated in the investigated optimization algorithms, including metaheuristics.

Several metaheuristics were studied in the research presented in this thesis, e.g., genetic algorithms, particle swarm optimization, simulated annealing, tabu search, covariance matrix adaptation evolution strategy (CMA-ES), memetic algorithms, and Bayesian optimization. All these algorithms are described in Chapter 6.

Due to significant computational problems and the long time needed to evaluate the quality of control settings of complex processes, it was necessary to find a way to speed up the computations. This was done using surrogate models based on machine learning techniques, mostly neural networks and gradient-boosted decision trees. However, some other surrogate models based on other machine learning techniques (like support vector machines [64]) and classical approaches like mesoscopic models (intermediate models between microscopic and macroscopic models, cf. Section 4.2.1.1.2) were also considered in preliminary experiments.

Overall, the combination of simulation tools, their surrogate models based on machine learning, and metaheuristics resulted in the approach that produced the best and most promising (in terms of future research and potential applications) results and performance so far. This methodology is described

in detail in Chapter 7. However, as discussed in Chapter 9, some other approaches such as reinforcement learning have been and are still being studied by the author of this thesis.

1.4 Research outcomes and original contributions

The research carried out and presented in this dissertation resulted in the following scientific contributions:

1. Development over the years of the initial Traffic Simulation Framework software [125] into a tool evaluating the qualities of different traffic signal settings using a microscopic traffic model. It can be considered a "digital twin" of real-world city traffic evaluating the qualities of different traffic control settings.

2. The development of new software for simulating cancer growth and evaluating the qualities of radiotherapy protocols. The software was developed by a group of students coordinated by the author of the thesis [15] based on the model described in [7], and significantly outperforms the previous implementations in terms of speed. It can be considered a "digital twin" of living tissue with cancer cells and treatment using radiotherapy.

3. New definitions of complex systems, complex processes, and their models, built based on analysis of some other definitions existing in the scientific literature.

4. A general methodology for optimizing complex processes based on combining computer simulations, metaheuristics, and surrogate modeling (described in Chapter 7).

5. The application of machine learning models as surrogate models that approximate the results of time-consuming computer simulations. To the author's knowledge, this was the first application of such machine learning models as surrogate models to simulate road traffic with traffic signal control and cancer evolution under radiotherapy treatment. This methodology solved computational challenges encountered during the research and enabled exploring a broad spectrum of metaheuristics in various settings. The idea was invented by the author of this thesis, but the implementation and experiments were carried out in collaboration

with a group of researchers from the TensorCell research group founded and led by the author of this thesis [325]. A large number of different surrogate models have been studied to determine their performance depending on many factors.

6. The development of a new architecture of a sparse graph neural network in which the topology of connections is based on the topology of a considered road network. This architecture is expected to find applications in other tasks related to road traffic and graph problems. The architecture was initially proposed by one of the members of the TensorCell research group, Łukasz Skowronek, but the author of this thesis coordinated the entire research and experiments.

7. A comprehensive evaluation of several metaheuristics as optimizers of combinatorial optimization problems related to two complex processes: road traffic in cities and cancer evolution.

8. A number of large datasets generated using simulators of road traffic and cancer evolution were developed during the work on this thesis (cf. Appendix A). These datasets have been made publicly available to facilitate further research on these topics by the scientific community.

Also, one of the outcomes of the presented research works was the foundation of the research group TensorCell [325] - it is an independent research group founded by the author of this thesis in 2017 to facilitate and accelerate research on the optimization of complex processes. Although it is not a scientific result, it can be considered a valuable outcome, as several team members have got interested in the domain of optimizing complex processes using AI, the team has produced several valuable scientific publications, and there is a potential for further scientific contributions.

1.5 Outline of the thesis

The rest of the thesis is structured as follows:

- Chapter 2 contains a comprehensive review of the existing literature and knowledge in fields most relevant to the research methodologies and approaches presented in this thesis.

- Chapter 3 focuses on explaining the concept of complex systems and complex processes and examines their underlying principles, characteristics, and behaviors.

- Chapter 4 contains a comprehensive review of methods of modeling complex processes relevant from the point of view of the thesis.
- Chapter 5 contains a presentation of the Traffic Simulation Framework which was one of the tools that were developed during the research by the author of this thesis and is one of the outcomes of the presented study.
- Chapter 6 presents an explanation of the concept of metaheuristics and a review of some of the most popular metaheuristic algorithms that were later used in experiments described in the thesis.
- Chapter 7 contains the general methodology for optimizing complex processes using metaheuristics and surrogate models that are proposed in this thesis.
- Chapter 8 presents the main experiments carried out within the research with a discussion of their results.
- Chapter 9 contains a discussion of the identified limitations of the introduced method, possible ways to overcome them, as well as potential extensions and future research directions that may lead to real-world applications.
- Chapter 10 concludes the thesis.

These chapters are followed by the Bibliography and the Appendix listing the datasets (Appendix A) and computer programs (Appendix B) used in experiments that have been made publicly available.

Chapter 2

Related works

In this chapter, a review of the existing literature and knowledge in fields most relevant to the research methodologies and approaches presented in this thesis is provided. This synthesis aims to establish a foundation for understanding the context and relevance of the subsequent work and findings. This review is divided into 4 parts:

- Section 2.1 reviews approaches related to optimization of complex processes.
- Section 2.2 reviews approaches to surrogate modeling.
- Section 2.3 reviews approaches to traffic signal control.
- Section 2.4 reviews approaches to optimizing cancer treatment.

Finally, Section 2.5 discusses the identified limitations of the existing methods for optimizing complex processes, while Section 2.6 summarizes the author's contributions in the context of related works.

2.1 Optimization of complex processes

Optimizing complex processes is a critical aspect of many industries, including manufacturing, healthcare, transportation, and logistics. By streamlining operations, businesses and public institutions can increase productivity, reduce costs, and improve user satisfaction.

In order to apply some optimization techniques, it is necessary to have a mathematical model of a complex process. As explained in Chapter 4, there are many approaches to modeling complex processes and usually, there is no universal method that is always best. All models have advantages and

disadvantages, so selecting the best models should depend on the intended goal of modeling and the application of the model. Chapter 4 contains a review of the most important aspects of modeling complex processes and presents examples of models focusing on the complex processes studied in this thesis: road traffic in cities (Section 4.2) and cancer growth with the presence of radiotherapy (Section 4.3).

Once the model of the complex process is available, it can be used to estimate the values of the objective function that should be optimized. Optimization problems arising from complex processes are usually of non-convex nature and they often exhibit many local minima. There have already been many studies on the optimization of complex processes (e.g., [48, 272]). However, most of these studies focus on specific complex processes and it is still an open question whether there exist universal, efficient techniques to optimize all kinds of (or a broad range of) complex processes.

One of the popular techniques that can be considered rather universal are metaheuristics like genetic algorithms or particle swarm optimization. They do not guarantee to find the optimal solution, but they are capable of finding near-optimal solutions within a reasonable amount of time. In the context of complex processes, metaheuristics can be used to find good values of parameters controlling the evolution of a complex process, and there have already been some studies in this field, e.g., [79]. Many metaheuristics are already widely studied, and a review of the most popular metaheuristics that are also considered in this thesis is included in Chapter 6. Sections 2.3 and 2.4 review the applications of metaheuristics in the context of traffic signal control and cancer treatment, respectively.

With the recent advances in machine learning, it is also natural to try optimization techniques based on its subfield, reinforcement learning (RL) [321]. RL is a type of machine learning where an agent interacts with its environment by observing its current state, taking actions, and receiving feedback based on them in the form of a reward or penalty. The agent then uses this feedback to adjust its policy (a mapping from states of the environment to actions) and make better decisions in the future. Its goal is to learn a policy that maximizes the cumulative reward it receives over time. The agent learns this policy through trial and error, exploring the environment and adjusting its actions based on the feedback it receives.

RL has many real-world applications, especially in environments that are dynamic and uncertain. Thus, it has found applications in fields like robotics [137, 175, 187], playing games [306, 44, 340, 25], finance [212], and autonomous vehicle control [346, 56]. It is also used to train large language models for natural language processing [13].

RL algorithms can also be used to optimize complex processes by dy-

namically adjusting their control parameters and taking actions in real-time depending on the current state of the process. This can potentially help in solving many challenges of complex processes, like sensitive dependence on initial conditions or computational irreducibility.

However, there are also weaknesses in the RL approach. The training is relatively slow and requires huge computational power. Its reproducibility is often a challenge and it also usually requires a large amount of data. This is because the algorithm learns through trial and error, and requires many iterations to converge to a good policy. There is also the *exploration vs exploitation* tradeoff: too much exploration (exploring new options in the search space) can lead to inefficient learning, while too much exploitation (refinement of already known options) can lead to suboptimal policies. Nevertheless, there are already approaches to optimize and control complex processes using RL (e.g., [11, 84, 244]). It can be expected that with the further development of RL methods and increasing accessibility to large computational power, these techniques can become state-of-the-art in future traffic signal control systems. As in the case of metaheuristics, Sections 2.3 and 2.4 review the applications of RL in the context of traffic signal control and cancer treatment, respectively.

Another approach to modeling and optimization of complex processes is based on the *Interactive Granular Computing* (IGrC) [168] described in Section 4.4. It is an approach to knowledge discovery and decision-making that is based on the principles of granular computing [18], which is a branch of computer science that deals with the representation, processing, and manipulation of information at different levels of granularity (the levels of detail at which the system is modeled). IGrC aims to help humans interact with complex and uncertain data. One of the main advantages of IGrC is its ability to handle uncertainty and ambiguity in the data. By creating granular structures that can capture different levels of detail and abstraction, IGrC is able to represent and process data that is incomplete or imprecise.

IGrC can be also used to optimize complex processes, e.g., to create a granular structure that captures the important features of the process and allows for the exploration of different scenarios and trade-offs. For example, in a manufacturing process, there may be many factors that affect the quality of the product, such as temperature, humidity, and pressure. These factors may interact in complex ways, making it difficult to predict the optimal conditions for the process. IGrC can also incorporate human expertise into the optimization process. By allowing humans to interact with the data and provide feedback, IGrC can help to identify important factors and constraints that may not be immediately apparent from the data alone.

2.2 Surrogate models

Surrogate modeling (metamodeling) is a technique used to build a computationally efficient approximation of a complex function [280]. It is used when an outcome of interest is difficult to be obtained, e.g., due to high computational complexity. Such applications are especially common in the case of black-box optimization (optimization that takes advantage only of inputs and outputs of a computer simulation or physical experimentation and does not use any other knowledge or assumptions, in particular: is derivative-free), in which it is quite often necessary to run multiple simulations in order to evaluate many different input settings. Particularly, in the research presented in this thesis, surrogate models were applied to approximate the outcomes of time-consuming computer simulations. Popular surrogate modeling approaches are:

- Response surfaces;
- Kriging, gradient-enhanced kriging (GEK);
- Bayesian approaches;
- Machine learning models.

Response surface methodology is a statistical method introduced by George E.P. Box and K.B. Wilson in 1951 [37]. It involves creating a mathematical model of the relationship between the input variables and the output response of a process. The model can be represented as $Y = F(X) + \epsilon$, where Y is the response variable, X is a vector of input variables, F is a mathematical function that describes the relationship between the input variables and the response (usually a polynomial function is selected, e.g., it can be Chebyshev polynomials [89], B-spline Polynomials [88], or minimal polynomials [29]), and ϵ is the random error term. To build the response surface model, experiments are conducted to collect data on the process response for different values of the input variables. The data is used to estimate the coefficients of the function F using techniques such as the least squares method or maximum likelihood estimation.

Kriging, also known as Gaussian process regression, is also a statistical technique developed by D.G. Krige [192]. Its idea is to estimate the value of a variable at an unobserved location as a linear combination of observed values at nearby locations. The weights of the linear combination are determined using a statistical model of the spatial autocorrelation of the variable to minimize the variance of the prediction error, subject to the constraint that

the sum of the weights to 1. The kriging model assumes that the variable being modeled is a random function with a mean and a spatial covariance structure. The mean can be estimated using a simple average of the observed values, while the spatial covariance structure can be estimated using a covariance function. One of the extensions of kriging is *Gradient-enhanced kriging* which uses the gradient of the performance metric (estimating how good the surrogate model is, e.g., mean squared error) with respect to the input variables to enhance the kriging model [335]. The gradient provides additional information about the behavior of the performance metric in the vicinity of the observed data points, thanks to which it can improve the accuracy of predictions. This can be particularly beneficial in regions where the performance metric is highly nonlinear or exhibits sharp gradients. The gradient model can be represented as a linear combination of basis functions, with coefficients estimated using techniques such as the least squares method or maximum likelihood estimation.

Bayesian approaches are also statistical methods. They use probabilistic methods to build models that capture the underlying structure of the data and to represent uncertainty in the model, which is then updated based on new data using Bayesian inference to update the probability distributions and generate new predictions. This allows the model to become more accurate over time, as more data become available. An example is Bayesian Neural Networks (BNNs) [49] which use Bayesian inference to estimate the weights and biases of the network.

BNNs are also considered an example of surrogate modeling based on machine learning, in which the idea is to generate a large dataset of relations between the inputs and the outputs of the approximated functions and use this dataset to train machine learning models. Artificial neural networks are considered as one of the good approximators of continuous functions, which is a consequence of the *Universal Approximation Theorem* [70] explained in Section 7. Other popular models are Support Vector Machines [64] (and Support Vector Regressors for regression tasks [302]) and gradient boosting decision trees [6].

The aforementioned techniques are rather general approaches to surrogate modeling (they are also called "functional metamodels" [263]). Therefore, they may be useful for building universal methods for optimizing complex processes. It is also worth mentioning that in some cases it may be possible to develop surrogate models that are specific to a given complex process (such metamodels are also called "physical metamodels" [263]). For example, as presented in Section 4.2, in the case of road traffic in cities there are different models, ranging from macroscopic to microscopic or even nanoscopic. They may have different conformance to real traffic, different evaluation times,

and different granularity (the levels of detail at which they model a given system). Therefore, it may turn out that some models can be considered surrogate models for some other models. However, more universal surrogate models can also be applied to the phenomenon of road traffic. One of the first such surrogate models was presented in [263], and it combined a general-purpose component (a quadratic polynomial), which provided a detailed local approximation, with a physical component (the analytical queueing network model), which provided tractable analytical and global information.

A more comprehensive review of surrogate modeling and its application to optimization tasks can be found in [97] and in [363].

2.3 Optimization of road traffic in cities

As presented in Section 1.1, there are many approaches to optimizing road traffic in cities, ranging from the approaches aiming to reduce the demand for using the road infrastructure (e.g., organizing public transport, introducing fees for entering some areas, etc.) to the approaches aiming to improve the supply (e.g., building more roads or optimizing traffic signal settings). In this thesis, in the domain of optimization of road traffic in cities, the focus is only on the traffic signal control approach.

Some cities have already installed traffic signal control systems such as SCATS [217], SCOOT [42], RHODES [145], OPAC [102], MARLIN [84] (a comprehensive review of existing traffic management systems can be found in [332]). In many cases their quality is good and they are able to improve the traffic, at least in typical, repeatable, and predictable conditions, as well as in the case of low travel demand and a low number of cars. They are usually not as good in the case of heavy demand or atypical conditions, such as sudden road blockage (e.g., caused by a car accident), changes in traffic organization (e.g., because of mass events or roadworks), and bad weather conditions. The following weaknesses of existing traffic management systems were identified:

- reactiveness, but not proactiveness: reacting to past and present traffic conditions (which might be sufficient for regular, smooth, repeatable traffic, but may not be sufficient when major changes happen - the reaction might be inappropriate or just too late to prevent the occurrence of a large traffic jam), but not anticipating and not preventing undesired traffic states;
- lack of accurate evaluation of changes introduced to the traffic control system;

- limited adaptation to the changing environment, relatively small space of possible modifications to the traffic signal settings;
- questionable efficiency in the case of large road networks.

Similar approaches to the one proposed in this thesis are used in theoretical works [60] or are implemented in traffic management systems SCOOT [42] and MOTION [277]. They use simulation models or surrogate models to evaluate traffic signal control strategies. However, contrary to the approach presented in this thesis, these approaches do not use surrogate models based on machine learning.

On the other hand, there are ongoing research works on applications of machine learning techniques to traffic signal control, and with the further development of AI, it can be expected that more and more traffic control systems will have some elements of machine learning control included. Metaheuristics are particularly popular in research on traffic signal control. A review of applications of metaheuristics to traffic signal control can be found, e.g., in [165]. There are already works applying genetic algorithms [202, 324, 54], particle swarm optimization [215, 51, 54], ant colony optimization [283, 138, 54], and memetic algorithms [292]. These approaches may differ in many aspects, like the road network structure, traffic conditions, representation of traffic signal control parameters, methods for evaluating the quality of settings, etc. However, most of these works are quite recent indicating that the research work on applications of metaheuristics to traffic signal control is modern and progressive. The research presented in this thesis was one of the first such approaches to comprehensively investigating and comparing many different metaheuristics.

Another area of research on traffic signal control that has gained popularity in recent years and appears to be very promising is the application of reinforcement learning [321], which was introduced in Section 2.1. In the case of traffic signal control, the environment can be represented as a computer simulation (like the one described in Chapter 5). The states may correspond to the states of traffic (e.g., numbers of cars or average speeds on road segments), and the actions may correspond to the durations of the phases or to the decisions on whether to change the traffic signal control phase or not. The rewards may correspond to delays, throughputs, or other traffic characteristics. There are even frameworks for developing and comparing RL-based traffic signal controllers. For example, RESCO [11] includes the implementation of state-of-the-art deep-RL algorithms for signal control along with benchmark control problems that are based on realistic traffic scenarios. One of the first and most successful approaches to using reinforcement learning is the MARLIN-ATSC system, which stands for Multi-Agent Reinforcement

Learning for Integrated Network of Adaptive Traffic Signal Controllers [84]. The system uses a combination of microsimulation modeling and reinforcement learning to determine optimal traffic signal actions based on the state and actions of linked traffic signals. More recent approaches to applying reinforcement learning to traffic signal control employ some latest machine learning techniques and architectures of neural networks based on attention mechanisms [337]. For example, the CoLight system [350] uses graph attentional networks (GAT) [338] to facilitate communication. Specifically, for a target intersection in a network, CoLight incorporates the temporal and spatial influences of neighboring intersections to the target intersection and builds up index-free modeling of neighboring intersections.

A similar approach based on GAT was presented in [343]. By utilizing meta-learning techniques, these networks can learn to adapt to new scenarios quickly and effectively. These techniques have been gaining the attention of researchers in recent years because graph-based models (like GAT) can capture complex spatio-temporal relationships in data, thanks to which they have the potential to outperform other solutions. As mentioned in Chapter 8.6, the author of this thesis has also run with his research team initial experiments aiming to use GAT as surrogate models of a traffic simulator, but they did not show any advantage over sparse graph neural networks introduced in this these (cf. Section 8.3.1.9). However, the results were at the same level, which indicates that there might be potential to explore such GAT architectures further.

2.4 Optimization of cancer evolution under radiotherapy treatment

Cancer is a disease that results from the abnormal growth and division of cells in the body. These cells can invade nearby tissues and organs and spread to other parts of the body through the bloodstream or lymphatic system. Cancer can develop in any part of the body and can take many different forms depending on the type of cells involved. The process by which normal cells become cancerous is complex and multifactorial, involving genetic mutations, environmental exposures, and other factors [224].

Cancer can be treated in various ways, depending on its type and stage, as well as the patient's overall health and preferences. According to [224], there are several ways of cancer treatment:

- Surgery: It involves removing cancerous tumors and surrounding tissues. Surgery is often the primary treatment for solid tumors.

- Radiotherapy (RT): It uses high-energy radiation, such as X-rays or proton beams, to kill cancer cells. RT can be used to shrink tumors, prevent recurrence after surgery, or relieve symptoms.
- Chemotherapy: It uses drugs to kill cancer cells. It can be given orally or intravenously and is often used to treat cancers that have spread throughout the body.
- Immunotherapy: It uses drugs or other substances to stimulate the body's immune system to recognize and attack cancer cells.
- Targeted therapy: It involves drugs that specifically target the cancer cells' unique characteristics, such as gene mutations or proteins. Targeted therapy is often used to treat cancers that have specific mutations.
- Hormone therapy: It is used to treat cancers that are hormone-sensitive. Hormone therapy involves blocking or removing the hormones that fuel the cancer cells' growth.
- Stem cell transplant: It is a treatment that replaces damaged or destroyed bone marrow with healthy stem cells. Stem cell transplant is often used to treat blood cancers.
- Photodynamic therapy: It is a treatment that uses special drugs, called photosensitizing agents, along with light to kill cancer cells. The drugs only work after they have been activated by certain kinds of light.

Usually, these methods can be optimized in various ways. Combining therapies and using more than one cancer treatment method at the same time is one of the simplest and most frequently used strategies. Another method is personalized medicine: tailoring cancer treatment to a patient's individual needs based on their genetics and other factors. Advancements in technology, including next-generation sequencing, have made it feasible to detect particular genetic mutations in cancer cells, enabling the use of precision medicine to target these mutations accurately. The development of improved drug delivery systems (like nanoparticles or liposomes) can enhance the effectiveness of cancer treatment by targeting cancer cells more precisely and reducing side effects. Another method, that was also employed in the research presented in this thesis, is optimizing treatment plans. There are approaches to the optimization of cancer treatment using game theory [317], genetic algorithms [7], Markov decision processes [182], or reinforcement learning [81, 244]. Typically, the parameters that are optimized relate to radiotherapy and chemotherapy.

There are several objective functions that are frequently used in cancer treatment optimization [21, 136]:

- Tumor control probability (TCP): It is a measure of the probability that a given treatment will successfully control the growth and spread of the tumor. TCP is often used in radiation therapy optimization, where the goal is to deliver a sufficient dose of radiation to the tumor while minimizing damage to surrounding healthy tissues.

- Normal tissue complication probability (NTCP): It is a measure of the probability that a given treatment will cause damage to healthy tissues surrounding the tumor. NTCP is often used in radiation therapy optimization to minimize the risk of radiation-induced side effects such as nausea, fatigue, and skin irritation.

- Overall survival (OS): It is a measure of the length of time that a patient survives after receiving a given treatment. The objective of OS optimization is to maximize the patient's overall survival time while minimizing the risk of cancer recurrence and side effects.

- Quality of life (QoL): It is a measure of the patient's overall well-being and includes factors such as physical functioning, emotional well-being, and social functioning. The objective of QoL optimization is to maximize the patient's QoL while minimizing the risk of treatment-related side effects.

- Cost-effectiveness: It is a measure of the value of a given treatment in terms of its cost and its impact on patient outcomes. The objective of cost-effectiveness optimization is to find the treatment plan that provides the best possible outcomes for the patient at the lowest possible cost.

The control parameters that are typically optimized depend on the specific treatment approach being used [169]. In radiation therapy, the control parameters usually include the radiation dose, the size and shape of the radiation field, and the number and timing of treatment sessions. In chemotherapy, the control parameters usually include the type and dose of chemotherapy drugs, the timing and frequency of treatments, and the duration of treatment cycles [301]. In immunotherapy, the control parameters usually include the type of immunotherapy agent used, the dosage and timing of treatments, and the length of treatment cycles [211]. In targeted therapy, the control parameters usually include the choice of targeted therapy agent, the dosage and timing of treatments, and the duration of treatment cycles [370].

In the research described in this thesis, the focus is on optimizing protocols (doses and times of their administration) of RT, which is widely considered to be a cost-effective tool in the treatment of many types of cancer, with 40-50% of cancer patients receiving some form of RT during their treatment [327].

Despite the fact that there are numerous types of cancer with varying pathologies and locations in the human body, clinical treatment protocols for radiation therapy tend to be very similar in practice. The most commonly used protocol in recent decades has been to administer a radiation dose of 2 Gy per day [264]. Due to the significant cost, risk of testing new methods on patients, and long delays associated with obtaining reliable clinical evidence for potential treatment protocols, it is essential to prioritize preclinical screening and identification of promising candidate protocols to ensure that clinical trials focus on the most effective alternatives. Therefore, it is natural to explore how metaheuristics can help in finding promising RT protocols.

The research on this topic was carried out by S. Angus and M. Piotrowska in [7] where they applied genetic algorithms and their approach was further extended during the research presented in this thesis. They used a simulator of the stochastic EMT6/Ro model of cancer growth under the RT treatment (discussed in Section 4.3) to evaluate the quality of protocols.

The original approach was computational-demanding, so was difficult to scale, there was also limited adaptation to the changing environment. On the other hand, it was proactive as it was based on evaluating the long-term impact of RT protocols.

In the research presented in this thesis, it was possible to significantly accelerate the simulations and train surrogate models based on machine learning to approximate the simulation outcomes which gave the ability to find better RT protocols.

2.5 Identified limitations of the existing techniques

Section 2.3 outlined the identified weaknesses of existing traffic management systems, including reactivity (but not proactivity), lack of accurate evaluation of changes introduced, questionable efficiency on a large scale, limited adaptation to the changing environment, and relatively small space of possible modifications. Existing techniques for cancer treatment have similar limitations.

The identified methods for optimization of other complex processes usually have similar limitations, e.g., the method for finding good radiotherapy protocols for cancer treatment discussed in Section 2.4 was computationally intensive, thus difficult to scale, and there was also limited adaptation to the changing environment.

In general, metaheuristics are usually able to find near-optimal solutions to specific optimization problems related to complex processes or, in practice, to the mathematical models of complex processes. As discussed in Chapters 4 and 7, the results that are good according to the mathematical model do not have to be equally good in the real world, since the models (or surrogate models) of complex processes are only their simplified representations. However, even for metaheuristics, operating on a large scale can be challenging due to the large space of possible options. Also, evaluating the qualities of the candidate solutions considered can be computationally demanding. By default, they also have limited adaptation to the changing environment, and usually, the solutions found for certain settings had to be further adjusted once the conditions became too different.

Reinforcement learning methods can be used to find good strategies for responding to changing conditions, because by design they can be used to teach agents controlling complex processes what actions to take in certain states. However, they are also computationally expensive, especially when the space of possible actions is large. As discussed in Chapter 9, the author believes that methods based on reinforcement learning are very promising, and he has already started to investigate these techniques for the optimization of both complex processes considered in this thesis. However, the field of reinforcement learning was not as advanced as it is today when he started to work on the optimization of complex processes, so the focus of this thesis is on metaheuristics and their combination with surrogate models that approximate computationally expensive simulations to evaluate the qualities of control settings, which is also a relatively new and promising approach, as discussed in Section 2.6.

All techniques applied to the optimization of complex processes usually have some strengths and weaknesses and aim to find a balance between different aspects. For example, increasing the quality of evaluation of introduced changes or increasing the size of possible actions usually leads to higher computational costs. Therefore, there is still a need to study and develop these techniques further.

2.6 The dissertation's contributions in the context of related works

As presented in this chapter, there are already many approaches to the optimization of complex processes (including traffic signal control and the optimization of cancer treatment) as well as to surrogate modeling existing in the scientific literature. The methods used for the optimization of complex processes that can be considered rather universal range from metaheuristics [79, 165] and reinforcement learning [321, 11, 84] to Interactive Granular Computing [168], while in the case of surrogate modeling, the most popular techniques are response surfaces [37], kriging [193], Bayesian approaches [49], and machine learning models [64, 6].

Based on the review of the state of the art, the author of this thesis concluded that the problem of optimizing complex processes can still be considered an open problem for which new solutions are highly desired by the scientific and engineering community. There are still limitations of the existing techniques and the research presented in this thesis is a step toward developing better, efficient, scalable and universal methods.

To the best of the author's knowledge, the approach based on using machine learning models to approximate the outcomes of traffic simulations described in the thesis is the first such approach for the purpose of traffic signal control, especially in combination with metaheuristics. This idea was first presented in 2016 at the "NIPS 2016 Workshop on Nonconvex Optimization for Machine Learning: Theory and Practice" [129] and even the reviewers emphasized that it is worth exploring the application of neural networks as surrogate models in this context. It is possible that this was also one of the first such approaches in the whole field of traffic engineering. The sparse graph neural networks introduced as surrogate models of traffic simulations are also a completely new architecture in this context.

In terms of using metaheuristics, the research works on their applications to traffic signal control or cancer treatment are also modern and progressive. The research presented in this thesis was one of the first such approaches to comprehensively investigating and comparing many different metaheuristics for traffic signal control. A similar study (but with only 3 metaheuristics, with a very simple and less realistic traffic model) was presented in [54].

As presented in Chapter 4, there are already many approaches to modeling complex processes, including road traffic and cancer growth, but the Traffic Simulation Frawemwork which was one of the main tools used in this study, was also one of the first solutions applying microscopic traffic models on the scale of large cities. The author's extension of the Nagel-Schreckenberg

model [246] allowed to perform quite efficient traffic simulations on a realistic road network of Warsaw, which was a significant achievement at the time when the tool was created. This is also one of the reasons why this tool became popular and found applications in many research works, as discussed in Chapter 5.

The application of surrogate models to approximate the outcomes of cancer growth simulations for the purpose of optimizing the radiotherapy treatment using metaheuristics seems to be a novel approach too. In the case of the cancer growth simulation model used in this thesis (cf. Section 4.3), the development of a new simulation tool based on GPU also gave a significant advantage and allowed the investigation of more optimization algorithms and radiotherapy protocols.

Last but not least, the author's approach of studying 2 complex processes from (apparently) completely different fields (traffic engineering and medicine) and trying to build universal methods capable of optimizing complex processes from many different domains is also quite unique. Researchers working in transportation or medicine rarely have sufficient knowledge and perspective to view the processes they study as specific use cases of some more general phenomena. It is more natural for researchers working in the field of complex systems, but even they usually find it challenging to combine knowledge and techniques from several different domains (such as cellular automata, computer simulation, machine learning, and metaheuristics).

In the context of the identified limitations of the existing techniques for the optimization of complex processes, the methods presented in this thesis aim to reduce the computational cost while ensuring proactiveness and the ability to accurately evaluate the impact of the introduced settings. They are still not directly adaptive to the changing environment, so for a good adaptation some other techniques should be applied and this is also one of the reasons why the author of this thesis studies the applications of reinforcement learning in addition to metaheuristics and surrogate modeling.

Chapter 3

Complex systems and complexity science

"I think the next century will be the century of complexity", Stephen Hawking, January 2000, [143].

This chapter explores the area of complex systems, which is an interesting and relatively new area of science. Its goal is to fulfill one of the main objectives of the presented research (cf. Section 1.2, i.e., to gain a deeper understanding of the nature of complex processes).

The meanings of *complexity*, *complex systems*, and *complex processes* are explained. In addition, complex systems and processes that are particularly important for engineering purposes and in this dissertation are indicated.

It is often said in colloquial speech that some systems, structures, processes, or phenomena are complex, to express that they possess some kind of complexity. Usually, it means that something involves many related parts, or that something is difficult and non-trivial. But how can *complexity* be understood from the scientific point of view? Can we precisely and mathematically define what a complex process or system is and if so, can we give a computational procedure that measures complexity or answers whether something is complex or not? This chapter present an outline concerning these questions.

3.1 What is complexity?

According to the etymology dictionary [90], the English word *complex* means *composed of parts* and originates from the Latin word *complexus*, past participle of *complectere* (*to embrace*), which derives from *com* (*with*) and *plectere* (*to weave, to braid, to twine, to entwine*).

The concept of complexity is used in many disciplines and it is often precisely, mathematically defined. In computational complexity theory, scientists study the amount of resources (e.g., time and memory) required for the execution of an algorithm - the more resources the algorithm needs to solve a given task, the more complex it is [66]. There are computational classes, such as P, NP, or NPC, which, in some sense, measure the complexity of a problem, since they classify problems by the amount of resources required to solve their instances [66]. There is no absolute set of problems that are complex or not, but the complexity is comparable and is linked to the nature of considered problems. Nowadays, it is possible to measure the complexity of many algorithmic problems, or at least its lower or upper bounds. However, there are still some open questions in the computational complexity theory, such as the famous *P versus NP* problem [66].

In algorithmic information theory, *Kolmogorov complexity* of a string of characters is the length of the shortest program that outputs that string [188, 203]. It depends on the description language, but the effect of changing description languages is bounded by a constant (dependent only on the description languages), according to the *Invariance theorem* [203], so this complexity turns out to be absolute. However, there is no program that could compute Kolmogorov's complexity for every string, which is a consequence of the halting problem and Gödel's incompleteness theorem [68].

In mathematics, Krohn - Rhodes complexity is a topic in the study of finite semigroups and automata, for any finite semigroup S it measures the least number of groups in a wreath product of finite groups and finite aperiodic semigroups of which S is a divisor [193, 284].

In software engineering there exist different measures of programming complexity, e.g.:

- *Cyclomatic complexity* measures the number of linearly independent paths through the source code - program's control graph [225]. It is equal to e-n+p, where e is the number of edges, n is the number of vertices and p is the number of connected parts in the program's control graph.

- *Halstead complexity metrics* [139] measure different values, e.g., program's length, difficulty, effort, and time required to code it. All measures can be expressed in terms of the number of operators/operands and distinct operators/operands in the program.

- Software structure metrics based on information flow between system components [150]. Experiments showed those metrics are strongly correlated with the occurrence of changes in the code.

In finite model theory and computational complexity theory, descriptive complexity characterizes complexity classes by the type of logic needed to express the languages in them [159], e.g., nondeterministic polynomial class (NP) is a set of languages expressible by the existential second-order logic [93], second-order logic with a least fixed point operator gives problems solvable in exponential time - EXPTIME class [159].

It can be concluded that the concept of *complexity* is relative and domain-specific, i.e., it may have different meanings in different domains. In fact, S. Lloyd in his paper [214] provides a non-exhaustive list of 42 definitions or, more precisely, ways of measuring complexity. Apparently, there is no universal complexity into which every other complexity could be transformed. However, there are already some known correlations, e.g., descriptive complexity demonstrates that the computational complexity of a problem can be understood as the richness of a language needed to specify the problem. Furthermore, all approaches to defining complexity try to answer the following questions about the thing under study [214]:

1. How hard is it to describe?
2. How hard is it to create?
3. What is its degree of organization?

In all aforementioned cases, there exists a property that can be quantified to indicate the level of difficulty or intricacy regarding description, generation/computation, or organization. Therefore, it is reasonable to introduce the concept of complexity. It is often possible to compare degrees of complexity and determine if something is more complex, but there are rarely strict border points that determine if something is complex or not.

If there are so many different complexities with no single comprehensive and useful definition, is it meaningful to study a general complexity science instead of all different areas separately? It turns out that there is an interesting scientific area of complex systems in which systems from many totally different fields are studied in order to discover certain general laws and learn how interactions between many parts give rise to unexpected properties [16, 305].

3.2 Complex systems

The complexity studied in the area of *complex systems* refers to many entities (potentially abstract) which interact according to simple rules leading to

emergent properties, which are difficult to be derived from the properties of the interacting entities [171, 194, 16]. Complexity science is the study of such emergent system behavior and seeks to understand how the complex behavior of a whole system arises from its interacting parts. Scientists named such interacting entities a *complex system*.

Definition 3.1 *A **complex system** is a set of many components (physical or abstract) which interact leading to emergent properties which are difficult to be derived from the properties of the components.*

A complex system has, in every moment of time, a state, which is the configuration of its components. Based on that, the evolution of a complex system in time can be defined as a *complex process*:

Definition 3.2 *A **complex process** is the evolution of a complex system in time.*

The presented definition of a *complex system* is based on similar definitions that appear extensively in the literature [307, 237, 221, 104, 194, 171, 286], e.g.:

- "Roughly, by a complex system I mean one made up of a large number of parts that interact in a nonsimple way. In such systems, the whole is more than the sum of the parts, not in an ultimate, metaphysical sense, but in the important pragmatic sense that, given the properties of the parts and the laws of their interaction, it is not a trivial matter to infer the properties of the whole." [307]

- "A complex system is a group or organization which is made up of many interacting parts. (...) In such systems, the individual parts called "components" or "agents" and the interactions between them often lead to large-scale behaviors which are not easily predicted from a knowledge only of the behavior of the individual agents." [237]

- "Complex system - a system with numerous components and interconnections, interactions or interdependencies that are difficult to describe, understand, predict, manage, design, and/or change." [221]

- "Still, we can say that a system is complex if it consists of several interacting elements so that the behavior of the system will be difficult to deduce from the behavior of the parts. This occurs when there are many parts, and/or when there are many interactions between the parts." [104]

- "A complex system is an ensemble of many similar elements, which are interacting in a disordered way, resulting in robust organization and memory." [194]

- "Complexity Science can be seen as the study of the phenomena which emerge from a collection of interacting objects," [171]

- "A complex system is literally one in which there are multiple interactions between many different components." [286]

- "Complex system: the elements are difficult to separate. This difficulty arises from the interactions between elements. Without interactions, elements can be separated. But when interactions are relevant, elements co-determine their future states. Thus, the future state of an element cannot be determined in isolation, as it co-depends on the states of other elements, precisely of those interacting with it." [105]

There are also other, essentially different, definitions of complex systems, existing in scientific publications, e.g.:

- "In one characterization, a complex system is one whose evolution is very sensitive to initial conditions or to small perturbations, one in which the number of independent interacting components is large, or one in which there are multiple pathways by which the system can evolve. Analytical descriptions of such systems typically require nonlinear differential equations. A second characterization is more informal; that is, the system is "complicated" by some subjective judgment and is not amenable to exact description, analytical or otherwise." [352]

- "Complex system describes phenomena, structure, aggregates, organisms or problems that share some common theme: (i) They are inherently complicated or intricate [...]; (ii) they are rarely completely deterministic; (iii) mathematical models of the system are usually complex and involve non-linear, ill-posed, or chaotic behavior; (iv) the systems are predisposed to unexpected outcomes (so-called emergent behavior)." [96]

- "Common to all studies on complexity are systems with multiple elements adapting or reacting to the pattern these elements create." [10]

- "In a general sense, the adjective "complex" describes a system or component that by design or function or both is difficult to understand and verify. (...) complexity is determined by such factors as the number

of components and the intricacy of the interfaces between them, the number and intricacy of conditional branches, the degree of nesting, and the types of data structures." [351]

- "Whether living or nonliving, complex systems are open, ordered, nonequilibrated structures that acquire, store, and express energy." [52]

There are also works that try to give a general definition of *complexity*:

- "To us complexity means that we have structure with variations." [116]
- "That property of a language expression which makes it difficult to formulate its overall behavior, even when given almost complete information about its atomic components and their inter-relations." [82]
- "Complexity theory indicates that large populations of units can self-organize into aggregations that generate pattern, store information, and engage in collective decision making." [268]

Such a large number of different definitions and viewpoints shows that the area of complex systems and complexity science is still at an early stage of development and there are different perspectives and viewpoints.

The definition 3.1, adopted in the dissertation, is relatively simple and based on the most common definitions. However, such a simple definition needs more elaboration since it is still vague, e.g., it does not explain how many entities we need, how many of them should interact and how, what it means that emergent properties are "difficult to be derived", what exactly is *interaction*, should components be of the same type or not, etc. The reason for such vagueness is that complex systems occur in so many disciplines and the concept concerns so many systems, that it is still difficult to give a single, precise definition. In contrast to the complexities discussed in mathematics and theoretical computer science, we still know too little about complex systems, and the terminology is not mature and developed enough to provide a rigorous, non-vague, mathematical definition of complex systems, on which all scientists would agree [171, 194]. In fact, there are cases where it is difficult to answer whether a given system is a complex system or not (another interesting question is whether it is justified to classify systems as complex or not, but as presented in Section 3.2.1, complex systems usually share some important properties that can give insight into a considered system, potentially also help in its analysis and optimization). It is possible that progress in the discipline will lead to more precise definitions. It is also possible that there will be no strict definition, simply because the concept covers so many different systems that seem to share some common features and could be

commonly defined as complex but are essentially too different. This is natural in the real world and vagueness exists in science and is often acceptable and desired for communication purposes [179]. For example, it is difficult to define precisely what a "mountain" is and to give its borders, or to precisely define "traffic jams", or to say what the precise borders of being "small" or "tall" are, but everyone knows intuitively what those concepts mean. In order to simplify communication our language just developed and evolved in a way to give common names to concepts (e.g., objects, properties, actions) that share some common features, while disregarding details that are not necessarily common. In the case of complex systems, this may lead to major issues:

- Complex systems are differently understood by different people and the definition of complex systems is not straightforward and scientists often define it differently.
- Vague definitions of complex systems are usually not useful for engineering and analytical purposes for which it is required to apply precise, domain-specific definitions of particular complex systems.

Luckily, the studies of the area of complex systems are much better developed, and in the literature, the given definition 3.1 (or similar definitions) is often clarified, not applied directly. The clarification is usually done in two ways that are described in detail in the next two subsections (3.2.1 and 3.2.2):

1. By giving examples of systems that are considered to be complex [237, 14, 171].
2. By indicating additional features that should be common to most studied complex systems [171, 194].

Despite vagueness, Definition 3.1 already brings important scientific and philosophical consequences: it does not follow the "classical" or Cartesian mode of scientific thinking, which is expressed most explicitly in the Newtonian mechanics, that dominated the science before the 20th century. According to [104], classical science before the 20th century was characterized by the following properties:

- *Reductionism*: to fully understand a system, one should decompose it into its constituent elements and study their fundamental properties.
- *Determinism*: every change can be represented as a trajectory of the system through (state) space, i.e., a sequence of states, following fixed

laws of nature. These laws completely determine the trajectory towards the future (predictability) as well as towards the past (reversibility).

- *Dualism*: the ultimate constituents of any system are particles, i.e., pieces of matter. Since matter is already completely determined by mechanistic laws, leaving no freedom for intervention or interpretation, the only way we can include human agency in the theory is by introducing the independent category of mind.

- *Correspondence theory of knowledge*: through observation, an agent can in principle gather complete knowledge about any system, creating an internal representation whose components correspond to the components of the external system. This establishes a single, true, objective mapping from the realm of matter (the system) to the realm of mind (the representation).

- *Rationality*: given such complete knowledge, in its interaction with the system, an agent will always choose the option that maximizes its utility function. Thus, the actions of the mind become as determined or predictable as the movements of matter.

Thus, in classical science, the representation of systems is based on the paradigm that all changes can somehow be reduced to the motion of separate, rigid objects in space, following trajectories completely determined by physical laws [152]. The evolution of such systems is causal, deterministic, and reversible. Its representation is supposed to be objective, i.e., independent of the observer. According to Laplace, if we had complete knowledge about such systems, i.e., positions of all elements and forces acting on them, and enough power to analyze those data, perfect prediction would be possible:

> We may regard the present state of the universe as the effect of its past and the cause of its future. An intellect that at a certain moment would know all forces that set nature in motion, and all positions of all items of which nature is composed, if this intellect were also vast enough to submit these data to analysis, it would embrace in a single formula the movements of the greatest bodies of the universe and those of the tiniest atom; for such an intellect nothing would be uncertain and the future just like the past would be present before its eyes.[197]

However, even the definition of a complex system (Definition 3.1) is opposed to *reductionism*, since understanding the properties of the components of the system may not be sufficient to understand the whole system, i.e., its

emergent properties. Other theories which do not follow the classical scientific approach are for example *quantum mechanics* (for which the mathematical formalism contains non-determinism) or *relativity* (there is no objective mapping from the observed system to its representation).

The inconsistency with the classical scientific approach in the case of complex systems may suggest that we deal with some kind of a new scientific domain for which new tools are required. A similar scientific revolution happened at the beginning of the 20th century when fundamentals of quantum mechanics and relativity theory were established. Indeed, many scientists believe we are witnessing the birth of a new scientific domain and the studies of complexity and complex systems will be of utmost importance in the 21st century [357, 147, 171].

3.2.1 Examples of complex systems

The definition of *complex systems* is often clarified in the literature by giving examples of systems that are considered as complex. The term *complex systems* usually encompasses such structures as:

- Society: Interactions between people lead to many emerging phenomena that are often difficult to predict (e.g., wars and conflicts, epidemics, friendships) [14]; groups of people who interact by sharing knowledge and combining skills may achieve goals (e.g., invent new ideas) that could not be achieved by a smaller subgroup or even by the same group without interactions (just by summing knowledge and skills of all members, but without interactions) [349];

- Vehicular traffic: Cars and their drivers moving on the road network according to specific rules, their interaction may give rise to traffic jams and car accidents [14, 171]; CAVs can also interact by sharing knowledge about traffic conditions, which can result in safer and more efficient traffic;

- Traffic management system: Such systems are usually composed of many sensors, servers, and controllers which interact by communication and are therefore able to manage traffic efficiently [14];

- Human genome: It consists of 23 pairs of chromosomes containing thousands of genes interacting with other genes in a decentralized way through genetic regulatory networks. This interaction is believed to be a significant factor responsible for all human complexity [17, 195];

- Ant colony: Ants interact (cooperate) in order to achieve some common goals, e.g., to build an anthill or bring food to the anthill [221, 237];
- Financial market: people and organizations trade items at prices that reflect supply and demand, interactions may cause market crashes, speculative bubbles, etc; [14]
- Flock of birds or school of fishes: Interaction between animals (e.g., birds, fishes) leads to specific patterns of collective movement, they keep moving as a large, compact group, just by following a few simple rules of interaction [14];
- Living human organism: Cells, tissues, organs are connected and interact forming more sophisticated structures and sustaining life [221];
- Neurons in brain / nervous system: Neurons interact giving rise to intelligence, consciousness, emotions [221];
- Global climate: Different parts of the atmosphere and Earth's surface interact giving rise to different weather conditions, short-term and long-term climate changes [221], [237];
- Cellular automaton: Cells may have a state and transition rules that steer their behavior: each cell "perceives" states of other cells and sets the proper value for the next iteration (cells interact by passing information about their state) [357];
- Machines constructed by humans, e.g., cars, computers, airplanes: many parts are connected and interact resulting in many complex actions (cars drive, airplanes fly, computers perform computations, etc), such systems, developed by humans, are also considered as *Engineering systems* [221];
- Chemical substances: Chemical substances react and create new substances, sometimes also producing side effects (e.g., energy); it is presumed that life emerged (and may emerge) from interactions in complex chemical systems through the process of abiogenesis [232, 261]. Studies in the area of complex chemical systems resulted in the Nobel Prize in Chemistry in 2013 [176];
- Planetary system (n-bodies system): Planets, stars, black holes, and other celestial objects interact by gravitational forces which determine their motion [221];

- Entangled particles: In quantum mechanics, quantum entanglement of particles is a phenomenon that occurs when particles interact with each other in such a way that the state of a single particle cannot be described independently, instead, a system of particles exists as a whole [83].

The diverse structures encompassed by the definition of complex systems originate from a wide array of disciplines, making it challenging to anticipate any shared physical properties. Intriguingly, real-world systems conforming to the definition 3.1 and classified as complex systems often exhibit a remarkable degree of commonality, despite their seemingly unrelated fields. This observation suggests the existence of underlying, yet-to-be-uncovered natural laws governing these systems. The subsequent subsection introduces characteristics that researchers generally regard as prevalent among complex systems.

3.2.2 Features of complex systems

In the second type of clarification of the definition of complex systems, the following properties are universally considered as common features of most complex systems [194, 171]:

1. The system contains a collection of many components or "agents". These components and their interactions are usually similar and simple.

2. Emergent behavior: The system exhibits emergent phenomena which arise because of interaction between components, in the absence of any "invisible hand" or central controller. The emerging phenomena can be surprising and difficult to deduce from the behavior of individual components.

3. Sensitive dependence on initial conditions (known also as the "butterfly effect": A small change in one state of a (deterministic) system can result in large differences in a later state.

4. Nonlinearity: The superposition of two solutions of the equations that describe the system does not produce another solution. Cause and effect may not be proportional, causing counter-intuitive behaviors.

5. Hierarchical structure: Complexity manifests itself across multiple levels within the hierarchical structure of complex systems. In particle physics, the Standard Model elucidates the behavior of elementary particles, such as quarks and leptons, and their interactions via weak,

electromagnetic, and strong nuclear forces. These particles constitute atoms, which achieve stability through electromagnetic interactions, although unstable ions and isotopes may also exist. Under specific conditions, atoms combine to form chemical compounds, which are stabilized by the same interactions. At a higher level, these chemical compounds can assemble into more intricate structures, such as living cells. Simpler prokaryotic cells evolved into more complex eukaryotic cells. These complex cells constitute tissues, which in turn compose organs, eventually forming highly complex organisms, including plants, animals, and humans. Humans themselves are complex systems, and even a group of two individuals exhibits complex behavior, as described in the previous subsection. Furthermore, groups of people can form societies with intricate structures, organizations, and processes.

6. Order and disorder: Complex systems display a sophisticated interplay between ordered and disordered behaviors, whereby order can emerge from interactions among seemingly disordered components and subsequently vanish. Consequently, complex systems are regarded as an intermediate between simple, ordered systems—where phenomena can be effortlessly described or calculated (e.g., the motion of a few billiard balls)—and vast, unstructured, chaotic systems that can be examined using statistical approaches (e.g., the motion of gas particles).

7. Openness: The system is typically "open", it may be affected by the environment or there is no strict border between the system and the environment [53].

8. Computational difficulties / computational irreducibility: Inability to *abbreviate* a system (or program) or describe its behavior in a simpler way. This concept was introduced by Stephen Wolfram [357].

3.3 Complex adaptive systems

It is also worth mentioning a special class of complex systems called *complex adaptive systems* (CAS), in which the individual and collective behavior mutate and self-organize as a response to some events [233]. An example is road traffic - if there are some unexpected events, like road closures due to car accidents and road works, the traffic participants may adapt their routes and, as a consequence, traffic jams may appear in other areas of the city. Living organisms, cells, ecosystems, and societies are also considered CAS.

Besides the general features of complex systems presented in Section 3.2.2, in the case of CAS there might be additional typical features like adaptation to environment, evolution, and self-organization.

One of the first mathematical approaches to study CAS was a model of cultural evolution (developed in 1968 by Walter Buckley [46]) which regards psychological and socio-cultural systems as analogous to biological species. In the modern context, complex adaptive systems are sometimes linked to memetics [308]. An interesting approach to studying CAS was recently presented in [155], in which CAS is characterized by hierarchical arrangements of boundaries and signals. The author developed a framework for comparing and steering CAS through the mechanisms that generate their signal/boundary hierarchies.

3.4 Conclusions

This chapter has provided a comprehensive introduction to the concept of complex systems, exploring their defining characteristics, diverse manifestations across various disciplines, and the intricate interplay between order and disorder that underlies their behavior. It also introduced relatively simple definitions of a complex system (3.1) and a complex process (3.2) based on some common definitions and explanations found in the literature.

Complex systems emerge through the organization and interaction of numerous components, often giving rise to unexpected patterns, structures, and behaviors that challenge traditional scientific approaches. As we have seen, complex systems can be found at multiple levels of organization, from subatomic particles and molecular structures to ecological systems and human societies. This ubiquity highlights the importance of understanding the fundamental principles governing complex systems, as it holds the potential to unlock new insights and advances across a broad range of scientific fields.

As our knowledge of complex systems continues to grow, researchers are developing novel methodologies, models, and computational techniques to better understand, predict, and manage these systems. By building on the foundational concepts introduced in this chapter, future studies can delve deeper into the intricacies of complex systems, uncovering the hidden laws that govern their behavior and furthering our understanding of the world around us. The motivation of the author of this thesis to further explore the realm of complex systems and develop universal optimization methods for complex processes is rooted in this enhanced understanding. To achieve this objective, a thorough comprehension of how complex processes can be modeled is crucial, and a summary of this topic is provided in Chapter 4.

Chapter 4

Modeling of complex systems

"Everything should be made as simple as possible, but not simpler." A. Einstein

4.1 Introduction

A model of a complex system is its simplified representation. The goal of modeling is, in general, obtaining information about a system and its evolution. This information can be used for a variety of purposes, such as improving decision-making or optimizing a complex system and its associated complex process.

There are many approaches to modeling complex systems and usually, there is no universal method that is always best. All models usually have some advantages and disadvantages, so selecting the best models should depend on the intended goal of modeling and the application of the model. As G.E.P. Box said: "Essentially, all models are wrong, but some are useful" [36].

Various approaches to modeling complex processes exist. However, they can be categorized based on shared criteria or characteristics. For vehicular traffic, S. Hoogendoorn proposed a classification scheme for models [156] that appears to possess universal applicability and could potentially extend to other complex systems as well:

- approach to model generation (deductive, inductive, intermediate);
- level of detail (submicroscopic, microscopic, mesoscopic, macroscopic);
- scale of independent variables (continuous, discrete, semi-discrete);
- representation of the process (stochastic, deterministic);

- operationalization (analytical, simulation);
- scale of application (networks, stretches, links, and intersections).

For several centuries, significant advancements in mathematics and the physical sciences have been achieved by developing simplified models of intricate phenomena, extracting properties from these models, and subsequently validating those properties through experimental means [43].

In this chapter, the approaches to modeling complex processes are explained with two examples. Section 4.2.1 presents a description of modeling one of the popular complex processes which is also a primary object of study in this thesis - road traffic in cities. To ensure a comprehensive approach from the perspective of traffic engineering, it is essential to situate urban traffic modeling within the broader context of transportation modeling (Section 4.2). Section 4.3 provides a description of modeling another complex process investigated in this dissertation - cancer growth in the presence of radiotherapy. Section 4.4 summarizes one of the recent approaches to developing mathematical tools for modeling and dealing with complex processes - Interactive Granular Computing (IGrC).

4.2 Transport modeling

The goal of transport modeling is to simulate and analyze various aspects of transportation systems, such as land use, travel demand, network design, road traffic, and policy evaluation. Transport models are used to forecast future travel patterns and assess the impact of different transportation policies and investments.

Transport modeling research is an established field of science, with its roots in the 1920s. One of the first important contributions was made by Frank Knight in 1924 [186], who highlighted the complexities and potential pitfalls in assessing the social costs associated with transportation projects, emphasizing the need for a careful and thorough analysis of costs and benefits. This work contributed to the development of cost-benefit analysis as a tool for evaluating transportation projects, which later became an essential component of transportation planning. While Knight made important contributions to transport economics, it was John Glen Wardrop who introduced the concept of traffic equilibrium in 1952, formulating Wardrop's principles [347]. These principles are a transport-specific counterpart to the game-theoretic Nash equilibrium, which was proposed by John Nash around the same time, and are related to traffic assignments discussed later in this section.

Depending on the required level of detail, accuracy, the forecast period, and available input data, different mathematical approaches can be used to model transport. Generally, transport models consist of demand models (modeling the process of selecting origins, destinations, routes, departure times, and modes of transport by travelers) and traffic models (modeling the dynamics of traffic flow).

For demand modeling, historically, an aggregate methodology referred to as the *4-step model* has been mostly used [226]. It is an established methodology for urban, regional, and national demand modeling [262]. The model comprises four steps related to travel choices:

1. Trip generation.
2. Trip distribution.
3. Mode choice.
4. Traffic assignment.

Trip generation answers the question of how many trips originate in or are destined for a particular zone and determines the frequency of origins or destinations of trips in each zone. Zones usually correspond to areas that are homogeneous in terms of land use, population, occupation/employment characteristics, and travel patterns.

The trip distribution step matches trip origins with destinations and answers the question of where the trips go. This is done by weighting the attractiveness of the potential destination and the effort required to get there such as road distance, travel time, and toll/cost. Depending on the segmentation of the model, multiple distribution matrices may be generated, e.g., by trip purpose or mode of transport. The output of this step is the so-called origin-destination matrix, which is typically a two-dimensional table with rows and columns representing origins and destinations, respectively. The cells of the matrix contain the number of trips or the portion of traffic that originates from each row and travels to each column.

In the mode choice process, trips between zones are allocated to different transportation modes. Which mode of transport people are using depends on their preferences and aspects of their household or person such as car ownership. The output of this step can be represented as a set of origin-destination matrices, each matrix of which represents a particular mode.

Traffic assignment allocates trips between an origin and destination to specific routes in the transportation network. There might be different route assignment procedures for different transport modes, such as private cars,

public transit, walking, or cycling. Regardless of the mode, the traffic assignment process is guided by theoretical concepts such as the aforementioned Wardrop's principles [347], which provide a framework for understanding how travelers choose their routes. Wardrop's principles propose that travelers choose the routes that minimize their individual travel costs, such as travel time or distance. In Wardrop's equilibrium state, no traveler can reduce their travel cost by switching to another route, and the travel times on all used routes between an origin-destination pair are equal. Unused routes have travel times equal to or greater than the used routes.

The 4-step model is broadly used in practice, but recently, more detailed approaches referred to as activity-based models or agent-based models (ABM) have been implemented in many locations [174, 367].

There are several elements in ABM:

- Agents represent people who have characteristics, goals, and behavioral rules. The actions of agents depend on the environment they inhabit.
- The environment provides a space where agents live. The environment is shaped by the actions of agents.
- Interaction rules describe how agents and the environment interact.

ABM evolve on their own once these micro-level elements are specified, and macro-level properties emerge from such evolution.

4.2.1 Modeling of vehicular traffic

Traffic modeling is a subfield of transport modeling that focuses specifically on simulating the flow of vehicles (or transit, or pedestrians) on road networks. It can be said that the outputs derived from demand models, especially the traffic assignment step, serve as inputs to traffic models in transportation analysis. Traffic models take into account factors such as roadway capacity, travel times, and congestion levels, providing a more detailed representation of the transportation system's performance. Traffic models are used to predict traffic congestion, travel times, and the capacity of roads and intersections. They are also used to evaluate the impact of different traffic control measures and to design intelligent transportation systems.

4.2.1.1 History and state of the art

"Learn from yesterday, live for today, hope for tomorrow." A. Einstein

The first traffic models developed in the 1920s and 1930s were based on empirical observations of traffic. These models sought to describe the relationship between traffic density (the number of vehicles on the road) and speed. The goal was to understand how traffic flow changes as the number of vehicles on the road increases. One of the earliest models was the Greenshields model developed in 1935 [135]. The model made the underlying assumption that, in the absence of interruptions or congestion, the speed of vehicles and the density of traffic are linearly related to each other, i.e., $v = A - B * k$, where v represents the speed of vehicles, and k denotes the density of traffic. The constants A and B are determined through empirical observations.

In 1955, British mathematicians, Lighthill and Whitham, developed a model based on analogy to fluid dynamics [208]. In 1956, P.I. Richards extended the idea by introducing "shock-waves on the highway", completing the so-called LWR model [285].

Since then, the mathematical description of traffic flow has been the subject of extensive research, which resulted in a broad scope of models describing different aspects of traffic flow operations. According to [156], these models could be classified according to criteria presented in Section 4.1, which are discussed in the following subsections.

4.2.1.1.1 Classification according to the approach to model generation

M. Papageorgiou distinguishes the following approaches to traffic model generation ([266], [156]):

- deductive - known accurate physical laws are applied by analogy to describe traffic (based on analogy to fluid dynamics and kinetic gas theory, they are described in detail in Section 4.2.1.1.2)
- inductive - available input/output data from real systems are used to fit generic mathematical structures, e.g., ARIMA models, polynomial approximations, neural networks (examples are machine learning models applied to predict traffic characteristics [220]).
- intermediate - basic mathematical model-structures are developed and later fitted using real data (examples are mesoscopic models described in Section 4.2.1.1.2)

4.2.1.1.2 Classification according to the level of detail

This categorization distinguishes models according to the description level of entities (e.g., cars) in the respective flow models:

- *Macroscopic models*: describing traffic from the viewpoint of the collective vehicular flow;
- *Mesoscopic models*: describing traffic flow at an intermediate level of detail between microscopic and macroscopic models. Vehicles and driver behavior can be represented in groups, or they can still be described individually, but in more aggregate terms (in particular, they are indistinguishable);
- *Microscopic models*: considering the time-space behavior of individual drivers under the influence of vehicles in their proximity;
- *Nanoscopic models*: high-level description of vehicles' subunits, and their interaction (internal and with their surroundings).

Macroscopic models Macroscopic models do not distinguish between the constituent parts of the flow, i.e., individual vehicles and their behavior, but instead describe traffic in terms of aggregated values such as speed, flow rate (number of vehicles passing a given segment of road in a given time), and density (number of vehicles occupying a given segment of the road). These models are usually based on (or derived from) partial differential equations and analogies to other well-known physical phenomena, such as fluid dynamics and kinetic gas theory.

The first significant and scientifically recognized macroscopic model was the Lighthill-Whitham model based on an analogy to fluid dynamics [208]. It describes the relationship between density and flow rate (given by functions of space x and time t: $\rho(x,t)$, and $q(x,t)$, respectively) by an equation derived from the conservation law (represented using an equation $\partial_t \int_a^b \rho(x,t)dx = q(a,t) - q(b,t)$, which can be interpreted as the number of cars is conserved between any 2 points a, b - if there are no entrances or exits, the density in a small element of length changes at a rate equal to the difference between the inflow and the outflow):

$$\frac{\partial \rho(x,t)}{\partial t} + \frac{\partial q(x,t)}{\partial x} = 0. \tag{4.1}$$

The equation is an analogy to the continuity equation from fluid dynamics or the Maxwell equation from electromagnetism. Whitham and Lighthill additionally assumed that the flow rate is a function of density,

$$q(x,t) = q(\rho(x,t)), \tag{4.2}$$

which gives

$$\frac{\partial \rho}{\partial t} + \frac{\partial q}{\partial \rho} \times \frac{\partial \rho}{\partial x} = 0. \tag{4.3}$$

However, for a more realistic description, an additional equation, an analog of the Navier-Stokes equation for fluids, describing the time-dependence of speeds has to be considered instead of the simple Lighthill-Whitham assumption [293].

The model has been used to analyze a number of traffic flow problems, e.g., to explain the existence of "shock-waves" as in the work of Richards [285] which introduced "shock-waves on the highway", completing the so-called LWR model. However, it also has some limitations. It does not take into account inertial effects, which implies that vehicles adjust their speed instantaneously, it also does not contain diffusive effects, which would model the ability of drivers to look ahead and adjust to changes in traffic conditions. Another drawback is that the equation does not have a unique continuous solution [156]. Also, real traffic is not a single stream, but it is composed of several substreams (e.g., a few traffic lanes), and intersections with traffic signals, signs, and turns. All these differences imply that a standard Lighthill-Whitham approach may be useful in the case of one-lane, straight roads, without intersections, traffic lights, and disturbances (e.g., long segments of highways), but may not be appropriate in the case of more complex transportation systems, e.g., urban road network.

Another interesting class of macroscopic models are so-called Payne-type models [156, 270] in which Equation 4.2 is replaced by an equation describing dynamics of speed $V(x,t)$. One of the most important properties of a Payne-type model is that in certain density areas, the model is metastable, i.e., small variations in the traffic density will yield regions of increasing traffic densities, leading to occurrences of start-stop waves or localized traffic jams [156]. Indeed, this property can be observed in real traffic on straight lines (e.g., highways). However, similarly to the Lighthill-Whitham model, it is not sufficient to model traffic in urban areas.

Another class of macroscopic models are Helbing-type models [156, 148], which are extensions of Payne-type models. They are derived from gas-kinetic equations and consist of conservation of vehicles equation, speed dynamics equations (similarly as in the Payne-type models), and an equation describing the dynamics of the speed variance.

All these models are continuous, i.e., relations between density, speed, and flow are continuous functions. However, there exist also macroscopic models which are discrete or semi-discrete [156]. Most of these approaches are established by the application of a finite difference scheme to the continuous model equations, involving numerical approximation in the spatial direction, tem-

poral direction, or both, preserving the essential characteristics of the underlying continuous model. As a consequence, there are discrete versions of the LWR model as well as discrete and semi-discrete versions of the Payne-type models. The most popular and successful approach is the Cell Transmission Model (CTM) developed by C. Daganzo [71]. It is a numerical method to solve the wave equation proposed by Lighthill and Whitham (Equation 4.1). In the method, the road is divided into homogenous sections (cells), and flow and density are evaluated at a finite number of intermediate points between cells. The length of the cell is chosen such that it is equal to the distance traveled by free-flow traffic in one evaluation time step. CTM produces results consistent with the continuous kinematic wave equation (Equation 4.1) and reproduces many phenomena (e.g., shock waves) predicted by that equation (in fact, in 1996 Lebacque showed that CTM is the so-called *Godunov scheme* for the wave equation [156, 115, 200], which has a nice interpretation that the flow out of a cell is locally defined by the smallest of 2 quantities: local traffic demand and supply [156]). In 1999, Daganzo extended CTM and introduced Lagged Cell Transmission Model (LCTM), which is even more accurate [72]. Besides accuracy, an important property of CTM (and LCTM) is that simulation results do not depend on the order in which the cells are evaluated because the flow entering a cell is dependent only on the current conditions within the cell and is unrelated to the flow exiting the cell. Thus, CTM can be applied for the analysis of networks more complex than highways (e.g., urban networks). In such a case, an interesting extension is the Link Transmission Model (LTM), which just assumes that a given road network link is just a single cell, and thus it can speed up simulations [364].

One of the main advantages of macroscopic traffic models is their ability to capture overall traffic trends and patterns. These models are also computationally efficient, allowing for quick analysis and evaluation of different traffic scenarios. They typically have a small number of parameters and may be easier to calibrate than other types of models. However, macroscopic traffic models have some limitations and weaknesses. They are unable to capture individual vehicle behavior and interactions, which can result in inaccuracies in predicting traffic flow under certain scenarios.

It is also good to emphasize that macroscopic models can be used not only to model relations between traffic characteristics for road segments but also for urban areas [210, 170]. Examples of simulation tools implementing macroscopic models for urban areas are OTM-MPI [118] and VISUM [151].

Mesoscopic models Mesoscopic traffic models describe traffic flow at a medium detail level. Vehicles and driver behavior can be represented in

groups, or they can still be described individually, but in more aggregate terms (in particular, they are indistinguishable). Three well-known types of mesoscopic flow models are the so-called headway distribution models, cluster models, and the gas-kinetic continuous models [156].

Headway distribution models focus on directed to the statistical properties of time headways (time gaps between successive vehicles in a traffic stream), or, alternatively, of vehicle spacing. Starting from an empirical observation of the distribution of time headways and assuming that they are independent and identically distributed (i.i.d.) random variables, headway distribution models are based on the definition of suitable probability density functions for such distributions [94]. It is important to note that the assumption of independence and identical distribution of headways may not always be true in real-world traffic situations, as factors like traffic conditions, driver behavior, and vehicle types can lead to correlated or non-identical headway distributions. Despite this limitation, the i.i.d. assumption often serves as a useful starting point for modeling headway distributions in traffic flow studies. Examples of headway distribution models are presented in [45] and [41].

Cluster models aim to capture the dynamics of the traffic flow by representing the formation of vehicle groupings, or clusters, which exhibit shared attributes. Clusters are often treated as homogeneous entities, with the assumption that the internal characteristics of vehicles within a cluster, such as headways or speed differentials, are not explicitly considered. The size of a cluster is typically dynamic, subject to fluctuations as a result of factors such as changes in driving directions at intersections. Cluster models primarily focus on the principles governing cluster formation, the circumstances leading to the emergence of clusters, and the properties defining these clusters. Examples of cluster models are presented in [94].

Gas-kinetic traffic flow models characterize the dynamics of the traffic flow by focusing on the evolution of speed distribution functions rather than examining the behavior of individual vehicles. These models draw inspiration from gas-kinetic theory and provide a macroscopic perspective on traffic flow dynamics by considering the statistical properties of vehicle speed distributions. In these models, some concepts of statistical physics are introduced, such as the reduced phase-space density, which is related to the expected number of vehicles present in an infinitesimal region, traveling with a speed defined on the basis of a probability distribution function. Such a concept can be seen as the mesoscopic version of the macroscopic traffic density. The first such model was developed by I. Prigogine and R. Herman in 1971 [276]. It is based on the kinetic theory of gases and aims to describe the transition from molecular to macroscopic scales of gas behavior. The model incorporates the principles of non-equilibrium thermodynamics developed by Prigogine to de-

scribe the evolution of a system away from thermodynamic equilibrium. An improvement of this approach was introduced in the Paveri-Fontana model that considers the phase-space density [269]. Later, Helbing [149] presented a gas-kinetic model for multilane traffic flow operations, which is a similar approach to the Paveri-Fontana model, but lane-changing is explicitly considered thanks to adding three additional terms: the speed diffusion term (taking into account the individual fluctuations of the speed due to imperfect driving), the lane-changing term (modeling dynamic changes due to vehicles changing lanes), and the rate of vehicles entering and leaving the roadway.

Mesoscopic traffic models are computationally more efficient than microscopic models while providing a better representation of traffic dynamics and individual travel behavior than their macroscopic counterparts. The limited level of detail in mesoscopic models may result in less accurate predictions compared to microscopic models, so they may not be suitable for studying the behavior of individual vehicles and the interactions between them, which is better accounted for by microscopic models.

Examples of traffic simulation tools that use mesoscopic models are DynaMIT [23] and Aimsun [320, 257].

Microscopic model In microscopic models, each traffic participant is modeled as a separate agent. In each time step, the vehicle's position, speed, and acceleration are calculated based on the characteristics of vehicles (usually those that are in front of the given vehicle) in the previous time step. Such models can be also based on, e.g., differential equations or cellular automata.

One of the first important microscopic models is the Wiedemann car-following model developed in 1974 [353]. It uses thresholds to define different regimes of the car-following, distinguishes constrained and unconstrained driving by considering perception thresholds, and incorporates lane-changing and overtaking. In 1999, it was updated (by modifying the perceptual threshold, maximum acceleration, and deceleration rates) to better model freeway traffic [177].

The next popular car-following model is the Gipps model [108]. It analyzes the behavior and response of the following vehicle based on the preceding vehicle driver's actions. The model is defined by a set of constraints that include the driver's desired speed and the vehicle's acceleration constraints, assuming that drivers would estimate their speed based on the vehicle in front to be able to come to full speed and stop safely if needed. The Gipps model has been implemented as a microscopic model in one of the most popular traffic simulation tools - Aimsun [320, 257].

Another important microscopic model is the Krauss model [313]. It is based on the assumption that drivers always try to maintain a minimum safe distance behind the vehicle in front of them and adjust their speed accordingly to maintain a safe following distance between vehicles. The model was implemented in a popular open-source microscopic traffic simulation package, SUMO [313].

Another notable car-following model is the Intelligent Driver Model (IDM) [333]. It is based on differential equations in which the speed of a vehicle, its distance from the vehicle in front, and the relative speed between the two vehicles, as well as some parameters (like the desired speed and the minimum spacing), are used as an input to determine the vehicle's dynamics.

Among the microscopic simulation models based on cellular automata, one of the simplest yet popular models is the Rule 184 automaton, which is a one-dimensional binary cellular automaton [153]. In this model, each cell contains a binary value (0 or 1), and cells with the state 1 represent cells occupied by vehicles. The locations of vehicles evolve in discrete steps according to the automaton's rule. The vehicles move in a single direction, stopping and starting depending on the vehicles in front of them.

One of the most notable examples of a microscopic simulation model based on a cellular automaton is the Nagel-Schreckenberg (NaSch) model developed in 1992 [246]. The NaSch model emulates single-lane traffic. The road is represented as a tape divided into cells. Space, time, and speeds are discrete. All cells have the same size (set to $7.5m$ by default). At any time, each cell may be empty or occupied by a single vehicle. The state of a single cell is the speed of the car occupying the cell (or *null* if the cell is empty). The speed of the car i (denoted as v_i) is also considered as an internal state of car i and can take a value from the finite set $\{0, 1, ..., v_{max}\}$, where v_{max} is common to all cars (usually $v_{max} = 5$). The state of the automaton at time $t + 1$ is obtained from the state at time t by applying at the same time the following 4 rules to all cars:

1. Acceleration: $v_i(t + 1) := min(v_i(t) + 1, v_{max})$.
2. Braking: $v_i(t + 1) := min(v_i(t + 1), d_i(t))$, where $d_i(t)$ is the number of empty cells in front of the car i.
3. Randomness: with probability p: $v_i(t + 1) := max(0, v_i(t + 1) - 1)$.
4. Movement: car i moves forward $v_i(t + 1)$ cells.

The first step models the assumption that each driver wants to drive as fast as possible. The second step indicates that drivers reduce their speed to

avoid crashes (overtaking is impossible since the road has only one lane). The third step introduces randomization into the model. It simulates that drivers sometimes spontaneously reduce the vehicle's speed due to, e.g., weather conditions or psychological aspects. The fourth step is the actual movement of the car. All steps are crucial and cannot be removed in order to maintain correspondence with the real traffic on highways. Figure 4.1 visualizes an exemplary state of the Nagel-Schreckenberg model.

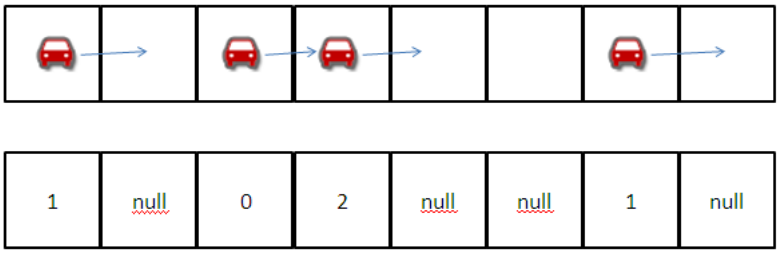


Figure 4.1: An exemplary cellular automaton corresponding to the situation on a single-lane road in the Nagel-Schreckenberg model.

One of the reasons why the NaSch model became popular is the fact that according to experiments, it is able to simulate real single-lane highway traffic with satisfactory accuracy [246]. Another reason is that the model is relatively simple, easy to implement, and implementation can be also computationally efficient. As in the case of most models based on cellular automata, there is a possibility to accelerate computations using GPU. Also, it has been already extensively studied [293] and implemented in professional traffic simulators [331] and transportation systems [339].

These were also the reasons why the NaSch model was selected by the author of this thesis for further investigation and extension to the case of realistic urban traffic scenarios, as discussed in Chapter 5. There had been other extensions of the NaSch model in the past, but to the best of the author's knowledge, this was the first extension of the NaSch model to the urban scenarios with realistic road networks (taken from the OpenStreetMap service [259]), traffic signals, multiple lanes, diverse road types, and various vehicle categories.

It is also worth mentioning one of the first extensions of the NaSch model - the Chowdhury-Schadschneider model developed in 1999 [63]. In this model, the road network is represented as a grid composed of N horizontal streets and N vertical streets. Every horizontal street intersects every vertical street, so there are N^2 intersections. On the streets, traffic moves in one direction.

Without loss of generality, it can be assumed that traffic moves to the right on horizontal streets and upwards on vertical streets. There are also traffic lights at every intersection. The lights for horizontal traffic are synchronized with each other (they have the same state at all times), similarly the lights in the vertical direction. In road sections between intersections, traffic follows the NaSch model.

Another notable microscopic model based on cellular automaton is the Biham–Middleton–Levine (BML) traffic model [27]. In this model, cars are represented by points on a lattice with random starting positions. Each car may move only downwards or only to the right (in this sense, it is similar to the Chowdhury-Schadschneider model). These two types of cars move in turns. On each turn, all cars of the corresponding type move one step forward if they are not blocked by another car. The BML model is considered to be the simplest system exhibiting phase transitions and self-organization.

Nanoscopic models Nanoscopic traffic models are even more detailed than microscopic models and can consider the cognitive processes within the driver or the mechanics of the vehicle [253]. In terms of the cognitive processes, they can take into account, e.g., the personal reaction time of the driver, the speed desired by the driver, as well as their changes in time. In terms of the mechanics of the vehicle, they can consider the mechanical structure of the vehicle (motor, gear, damping, steering), variation of vehicle's weight due to reduced weight of the tank, etc. They are especially useful for modeling the drive of autonomous or semi-autonomous vehicles but less for traffic signal control. However, optimizing control parameters of drivers or autonomous programs steering vehicles may also lead to traffic optimization [124].

4.2.1.1.3 Classification according to the scale of the independent variables

[156] distinguishes two time scales: continuous and discrete. In continuous models, the state of the transportation system changes continuously over time in response to continuous stimuli. An example is the Lighthill-Whitham model [208]. Discrete models assume that state changes occur at discrete time instants. An example is the Nagel-Schreckenberg model [246]. Besides time, other independent variables (e.g., position, speed, desired speed) can be also described in a continuous or discrete way. There also exist mixed models, e.g., based on the PDDL+ approach introduced in [336].

4.2.1.1.4 Classification according to the representation of the processes

This classification corresponds to the determinism of the model. Mathematical models of complex processes could be either deterministic or stochastic. An example of a deterministic model is the Lighthill-Whitham model [208], while an example of a stochastic model is the Nagel-Schreckenberg model [246].

4.2.1.1.5 Classification according to the operationalization

Models can be operationalized as analytical solutions of sets of equations, or as simulation models. Most traffic models are quite advanced and usually, simulation models are used in practice. However, in some cases and simplified versions, analytical solutions can be obtained. The examples are the Greenshield's model [135] and simplified versions of the LWR model [285].

4.2.1.1.6 Classification according to the scale of application

Models can describe traffic on a single road segment, a corridor, an intersection, a highway, an entire city, etc.

4.3 Modeling of cancer growth

Another complex process that was investigated in this thesis was the process of cancer evolution under radiotherapy treatment. In this case, living cells interact according to specific rules, and some of them may become cancer cells. Radiotherapy can affect this process and, eventually, reduce the number of cancer cells.

The cancer growth itself, even without radiotherapy or other external factors, can be considered a complex process as well. Numerous methods exist for modeling cancer growth, and they can be broadly classified according to their representation of tumor tissues as either discrete (cell-based) models or continuous models [216].

In discrete modeling, each individual cell is monitored and modified based on a distinct set of biophysical principles, and there are 2 main types of discrete models: lattice-based and lattice-free. The first approach represents the behavior of distinct tumor cells as automata on a grid, with their states controlled by deterministic or probabilistic rules, so cellular automata are natural mathematical tools for these models. The second type characterizes the activities of individual cells located anywhere and their corresponding

interactions, they are usually modeled using agent-based modeling. Discrete methods rely on a set of rules for each cell, allowing for the translation of intricate biological processes (e.g., mutation pathways, cell-cycle events) into model rules. However, the computational expense escalates quickly as the number of cells in the model increases, constraining the spatial and temporal scales that can be achieved. Moreover, while these models can describe biophysical processes in considerable detail, obtaining reliable measurements of model parameters through experiments that can capture the necessary detail at the cellular scale may not be straightforward [216].

In the context of larger-scale systems, employing continuous methods that conceptualize tumors as aggregations of tissues, with cellular densities or volume fractions as descriptive measures, presents a viable modeling alternative [216]. These models utilize reaction-diffusion equations to represent tumor cell density, the extracellular matrix, matrix-degrading enzymes, and the concentration of cellular substrates like glucose, oxygen, as well as growth factors and inhibitors. They are usually based on ordinary and partial differential equations.

There also exist hybrid, discrete-continuous methods that can be divided into two groups: composite models in which tumors are depicted as an aggregation of discrete elements, such as cells, while cellular substrates, including nutrients and growth or other chemical factors, are represented as continuous variables, and hybrid models that incorporate different modeling approaches to capture multiple scales and aspects of cancer growth [5].

Besides the general models of cancer growth, some models take into account the impact of various treatment methods, like radiotherapy [274, 287, 85], which can lead to direct cell killing, damage to the vasculature, and alterations in the tumor microenvironment. Depending on the chosen model, including radiotherapy can be done by adding terms that represent radiation-induced cell death, modifying cell proliferation and migration rates, or altering parameters related to nutrient and oxygen availability.

In the research described in this thesis, a model of breast cancer introduced in [7] was employed to evaluate the quality of radiotherapy strategies. It is a model of EMT6/Ro cell line derived from the EMT6 (Experimental Mammary Tumour-6) cell line that was isolated from the breast of a mouse with a mammary tumor. The model is an extension of the MCS stochastic asynchronous cellular automata model of EMT6/Ro dynamics that was earlier introduced in [273] by adding a calibrated multi-fraction irradiation module. Thanks to that, the model can be used not only to study various aspects of tumor biology, such as tumor growth and formation of the necrotic core but also as a response to radiotherapy. The model was calibrated and its performance was validated by taking into account various tumor charac-

teristics such as the number of cells, saturation size, tumor volume, doubling time, the thickness of the proliferating rim, cell phase population fraction, onset and progression of necrosis, and the effective dose resulting from multi-fraction irradiation. The details can be found in [264, 7] and in Section 8.2.2.

4.4 Interactive Granular Computing

The classical models of complex processes can already be very useful in various applications ranging from analysis and prediction to optimization and real-time management of a complex process. As we have already cited (cf. Section 1.1), they work especially well when the complexities ignored in the models (in order to make simplifications) are not the essential properties of the phenomena, but do not work when the ignored complexities are the essence [43, 105].

One of the notable examples of the recent approaches to developing mathematical tools for dealing with complex processes is the *Interactive Granular Computing* (IGrC) [168, 80, 167]. It is an approach to knowledge discovery and decision-making by intelligent systems that aim to help humans interact with complex and uncertain data. IGrC is based on the principles of granular computing, which is a branch of computer science that deals with the representation, processing, and manipulation of information at different levels of granularity [18]. The key idea behind IGrC is to create a collaborative environment where humans and machines (as well as other physical objects like robots, sensors, actuators, and cellular networks) can work together to solve complex problems. In this environment, humans provide the context and the domain knowledge, while physical objects provide the computational power and the ability to process large amounts of data and perceive them. The basic objects in the IGrC-based modeling are the so-called informational granules (ic-granules) linking abstract and physical objects. Control of granules transforms the current configuration of such ic-granules (used for perceiving the current situation) into a new one. IGrC involves several stages of knowledge discovery, starting with the creation of a granular structure to represent the data. This structure is then used to extract patterns and relationships from the data and to create a knowledge base that can be used for decision-making.

One of the main advantages of IGrC is its ability to handle uncertainty and ambiguity in the data. By creating granular structures that can capture different levels of detail and abstraction, IGrC is able to represent and process data that is incomplete or imprecise.

As presented in Section 2.1, IGrC can be also used to optimize complex processes.

4.5 Conclusions

There are numerous approaches to modeling complex processes, reflecting the ongoing and evolving nature of this field. As new tools, techniques, and data sources emerge, it becomes increasingly important to represent real-world phenomena accurately and efficiently, while calibrating and validating the developed models. However, it is crucial to acknowledge that no model is perfect - they merely serve as approximations of the studied real-world phenomena. Consequently, the selection of a particular model should be driven by the specific goals of the complex process modeling endeavor.

Chapter 5

Traffic Simulation Framework

"New directions in science are launched by new tools much more often than by new concepts. The effect of a concept-driven revolution is to explain old things in new ways. The effect of a tool-driven revolution is to discover new things that have to be explained." F. Dyson

5.1 Introduction

This chapter introduces the Traffic Simulation Framework software (TSF) - a comprehensive tool for simulating traffic in urban areas which was developed over many years by the author of this thesis. It can be considered as a "digital twin" of real-world traffic in cities evaluating the qualities of different traffic control settings. In particular, this framework facilitates the evaluation of traffic signal settings and the generation of datasets for further traffic optimization purposes.

The work on this tool was initiated by the author of this thesis in 2008 but is still ongoing and many new features were added during the work on this dissertation (it is also worth mentioning that the work on TSF was preceded in 2007 by the work on the Urban Transport Control System [38] which was intended to be an intelligent navigation system taking into account real-time traffic conditions). TSF already found numerous scientific applications, some of which are also discussed within this chapter (Section 5.7). As it is one of the outcomes of the research presented in this thesis and played a crucial role in producing other outcomes, it is essential to describe it in detail.

TSF includes implementations of the classical 4-step travel demand model (described in Section 4.2) as well as 2 traffic models:

- a microscopic traffic model extending the well-known Nagel-Schreckenberg model (NS model) described in Section 4.2.1,

- a mesoscopic model based on simple assumptions.

The travel demand model implemented in TSF is described in Section 5.2. Section 5.3 presents the underlying microscopic model, while Section 5.4 outlines the mesoscopic model. Then, 5.5 presents the implementation of TSF tool and its functionalities. Section 5.6 discusses how the software was calibrated. Finally, Section 5.7 describes some of the existing and potential future applications of TSF.

5.2 Travel demand model

TSF implements the classical 4-step travel demand model (described in Section 4.2) composed of trip generation, trip distribution, mode choice, and route assignment.

Trip generation determines the frequency of origins or destinations of trips in each zone. By default, TSF uses a map of Warsaw taken from the OpenStreetMap (OSM) service [259], covering an area of about 30 km per 30 km (from longitude 20.830078125 to 21.26953125 and from latitude 52.106505190756316 to 52.375599176659101) that is divided into 40×40 grid representing 1600 zones. However, it is also possible to use 2 other divisions into zones designed based on a traffic study in Warsaw done in 2005. They contain 399 and 774 zones, respectively, and were kindly shared by the administration of urban roads in Warsaw (*Biuro Drogownictwa i Komunikacji*) in 2015 [58]. In these cases, the zones are not regular but correspond to more natural borders between districts (like roads, parks, etc.).

Regardless of the type of division, the frequency of origins or destinations can be determined for each zone separately by the user, they can be also read from configuration files. The road network graph contains nodes and edges inherited from OSM, so each zone contains some number of nodes. Nodes are points at which trips can start or end. In TSF, it is assumed that within each zone, the locations of the starts of trips are uniformly distributed between all nodes in the given zone. Similarly, the locations of destination points are uniformly distributed between all nodes in a given zone. Therefore, in the trip generation step, TSF first selects the origin zone based on the defined distribution (frequencies of origins for each zone) and then it selects the origin point from the chosen origin zone with a uniform distribution.

Then, trip distribution matches origins with destinations. For each origin point, TSF first selects the destination zone. In TSF, it can be done in 2 ways. The first option is a random selection based on the distributions of frequencies of destination zones. The second option is using the origin-destination matrix

(OD matrix) loaded from a configuration file. The main difference is that in the first option, for each origin zone the distribution of destination zones is the same, while with the OD matrix, there might be different distributions of destination zones for each origin zone. After the destination zone is selected, TSF selects a node from that zone based on a uniform distribution.

The mode choice step is currently trivial, as there is only one available option now: private cars. However, there are plans to include other modes of transport too. For example, as part of the research carried out by the author of this thesis, microscopic traffic models for connected and autonomous vehicles (CAVs) were extensively studied, which resulted in 2 scientific publications [132], [128], and several popular-science presentations. Also, the author carried out research on public transportation, as well as shared mobility options (like van-pooling, bike-sharing, scooters, cargo bike sharing, and pedestrian traffic), during his work in some external research projects. However, at the moment of writing this thesis, no other modes of transportation are implemented in TSF. When such modes become available in the future, it is planned to implement the mode choice step using the logit model [2].

Route assignment allocates trips between origin and destination by a particular mode to a route. In TSF, a single route is a path (sequence of nodes) in the road network graph. The first element on the path is always an origin point and the last element is a destination point. The paths can be read from an input file, so they can be specified by the user arbitrarily. However, TSF makes it possible to calculate any number of routes before the start of the simulation and save them to a file to read them later as input.

The routes calculated by TSF are, by default, the optimal routes in terms of minimizing the sum of weights assigned to edges. Optimality of routes can be defined in many ways, e.g., in some cases, it is necessary to calculate the shortest path, in other cases - the fastest path or the cheapest path (in terms of fuel consumption), etc. Therefore, the weights can be assigned differently, depending on the specific application. One of the interesting features of TSF is that it is possible to specify different types of road segments (edges), such as highways, primary roads, secondary roads, or residential roads, and assign to them specific attributes that can be used to calculate the weights.

By default, TSF assumes that drivers choose routes that are optimal in terms of the time required to reach a destination point. The corresponding weights of edges can be calculated based on the length of the edge and a default speed according to the type of road segment or traffic condition data collected from real-world traffic. The optimal path (in terms of the sum of weights assigned to edges) can be computed in several ways, but in TSF, the default algorithm is the A* algorithm [142].

After the routes are generated, it is possible to run simulations according

to the traffic model.

5.3 Microscopic traffic model

The microscopic model in TSF is a probabilistic cellular automaton that inherits some foundations from the well-known Nagel-Schreckenberg model (NaSch model) described in Section 4.2.1.1.2, but introduces important extensions to emulate real road traffic in large urban areas. For example, it takes into account traffic signals, multiple lanes on a road, distinguishability of drivers, different starting and destination points, and the possibility of turning at an intersection and overtaking another vehicle. The main properties of the TSF model:

- Time is discrete. The model evolves in time steps.
- The model is multi-agent and microscopic: each vehicle is represented as a separate agent taking decisions independently.
- The road network is represented as a directed graph $G = (V, E)$, where V is the set of vertices (nodes) and E is the set of edges connecting nodes. Vehicles move between the graph's nodes and the actual movement takes place on the edges of the graph.
- Some intersections have traffic signals at their entries. The traffic signals located at the same intersection are synchronized. Note that intersections are not the same as nodes of the road network graph. Some large intersections can be composed of many nodes and the traffic signals can be located only in the nodes. On the other hand, nodes can be also located on straight road segments without intersections.
- Every vehicle has its own starting point and destination point. Starting points and destination points are randomly selected from a specific distribution based on an origin-destination matrix which may be derived from real-world data (the details are described in Section 5.2).
- For each vehicle, the route is selected by routing algorithms (e.g., Dijkstra algorithm [77] or A* algorithm [142]) aiming to find the best route based on weights assigned to edges. The weights can correspond to specific parameters that can be specified by the user, e.g., the edge's length or estimated time of travel.
- Every vehicle has a parameter that indicates a time step (from the start of the simulation) in which the vehicle should start its drive.

- Vehicles that reach their destination points no longer take part in the simulation.
- The set of roads (edges) is divided into several classes. Roads from the same class are indistinguishable in terms of the number of lanes and distribution of the vehicles' speed. There are currently 4 classes of road segments: freeways, primary roads, secondary roads, and residential roads, but it is possible to define more types (in OSM, there are more available types, but in TSF, they have been grouped together for simplicity).
- Every edge of the graph G may consist of several lanes. The number of lanes depends on the class of a road segment.
- Each lane is divided into cells just like in the NaSch model. All cells on the same edge have the same size.
- Every cell of the graph G may be occupied by at most one vehicle, just like in the NaSch model.
- Vehicle's speed does not need to be discrete, unlike in the NaSch model. This means that vehicles can also have positions within a cell.
- Vehicles are distinguishable. Every driver has their own profile which influences their behavior on the road, unlike in the NaSch model.
- Drivers always tend to increase their speed (but not to exceed maximal speed), just like in the first step of the NaSch model.
- Drivers reduce vehicle's speed before an intersection. The reduction may depend on the action the driver intends to take (turn at the intersection or not).
- Vehicles can change lanes if there is more than one lane on the road.
- When there is no possibility to change lanes, drivers have to decelerate due to other vehicles in front of them, just like in the second step of the NaSch model.
- Drivers may randomly reduce their speed, just like in the third step of the NaSch model (however, this option can be disabled to simplify computations).
- When the vehicle's speed is established, it moves just like in the fourth step of the NaSch model.

A random reduction of the speed could be potentially disabled in some experiments to simplify and accelerate computations. The reason is that in the case of stochastic models, in order to obtain statistically significant results, it is often necessary to run multiple simulations and calculate statistical values, like the average, median, or standard deviation of the considered values. In the case of a deterministic model, it is sufficient to run an evaluation just once. Of course, a natural question is whether removing the stochastic component is reasonable and will still lead to valid results. It may depend on the goal of using such a model, but as discussed in Section 8.2.1, in the case of traffic signal control such a simplification is reasonable.

5.3.1 Transition between states in the TSF model

This section presents a procedure for transitioning between states of the cellular automaton at time t and time $t+1$.

Let $t \in T$ be a given step, $CARS$ is the set of all cars participating in the simulation, and $CARS_t$ is the set of all cars involved in the traffic at time t (the cars that started moving before step t but have not finished yet). The parameters $turnParameter$ and $intersectionParameter$ are given, which affect the behavior of the driver when turning at an intersection and when passing an intersection, respectively. There is also an additional parameter *prob*, which introduces randomization into the driver's behavior, just like in the $NaSch$ model.

The general procedure for transitioning between states of the cellular automaton at time t and time $t+1$ is presented in Algorithm 5.3.1.

The algorithm can be explained as follows:

- Every driver tends to increase the vehicle's speed up to the speed limit. It is modeled in the procedure $increaseSpeed(car, t)$. The speed limit depends on the driver's profile and type of the road.

- Check if the vehicle reaches the intersection with the traffic signal with the *red* phase (it is done in the procedure $stopAtSignal(car, t)$). If the vehicle has to stop, then the procedure $reduceSpeedAtSignal(car, t)$ reduces the driver's speed in order to stop just before that intersection.

- If the vehicle does not have to stop at the red signal, but reaches the intersection ($intersection(car, t)$ returns $True$) and turns on it ($turnAtIntersection(car, t)$ returns $True$), then the vehicle's speed is reduced in the $reduceSpeed(car, t, turnParameter)$ procedure according to the value of the $turnParameter$.

Algorithm 1 Algorithm TSF-move: Move of a single *car* at time t

Require: $t \in T$, $car \in CARS_t$, $turnParameter$, $intersectionParameter$, $prob$

```
increaseSpeed(car, t);
if stopAtSignal(car, t) then
  reduceSpeedAtSignal(car, t)
else
  if intersection(car, t) then
    if turnAtIntersection(car, t) then
      reduceSpeed(car, t, turnParameter);
    else
      reduceSpeed(car, t, intersectionParameter);
    end if
  end if
end if
if shouldChangeLane(car, t) then
  changeLane(car, t);
end if
safeReduceSpeed(car, t)
with probability prob: reduceSpeed(car, t);
makeMove(car, t);
```

- If the vehicle reaches the intersection ($intersection(car, t)$ returns $True$), but does not turn at it, then its speed is reduced in the procedure $reduceSpeed(car, t, intersectionParameter)$.

- Until now, the vehicle's speed was calculated when there were no other vehicles on the road. However, the interaction between vehicles should be also considered. If there is another vehicle in front of the considered vehicle at a close distance, then the driver may decide to change lanes. The procedure $shouldChangeLane(car, t)$ checks if it is possible to change lanes. If yes, then the vehicle changes lanes in the procedure $changeLane(car, t)$.

- It is still possible that the vehicle is too close to another vehicle that is in front of it, so the procedure $safeReduceSpeed(car, t)$ checks this and possibly reduces the speed to maintain a safe distance between cars.

- The algorithm takes into account that some random factors (e.g., weather, psychological aspects) may result in a slight reduction of the vehicle's

speed, so with probability *prob* the vehicle's speed is reduced by a constant value in the procedure $reduceSpeed(car, t)$.

- Finally, the vehicle moves forward in the procedure $makeMove(car, t)$.

It is important to notice that the presented procedure is still general. Some of its internal subprocedures (i.e., $shouldChangeLane$, $safeReduceSpeed$) can be implemented in many ways and with various assumptions. In addition, input parameters can also influence the model. The only way to check the model's correctness is to compare the results (data) produced by a specific implementation with data collected from real traffic. However, in the case of a microscopic simulation model, it requires collecting data about the locations of many cars with a high frequency. Also, since driver behavior and traffic regulations vary from country to country (and sometimes even from city to city within a given country), the calibration process may need to be performed independently for each study area.

More details about the TSF and its microscopic traffic model were presented by the author of this thesis in [123] and [125].

5.3.2 Strengths and weaknesses of the model

One of the strengths of the presented microscopic model is that it is based on the well-known and comprehensively studied NaSch model. It just extends the NaSch model to the case of large-scale urban areas. Another strength is its simplicity. As in the case of the NaSch model, its extension to large urban areas is also relatively simple as well as easy to understand and implement compared to many other models. One of the consequences of simplicity and using a cellular automaton is a relatively low computational cost. Moreover, it is well known that the simulation of cellular automata can be also accelerated using Graphical Processing Units (GPUs), which gives the possibility of further accelerations in the future.

However, as presented in Chapter 7, the approach for optimizing traffic signal settings using metaheuristics requires running a large number of evaluations of traffic signal setting qualities (and similarly for optimizing other complex processes). One of the consequences is that in order to find near-optimal settings, the time required to run a single evaluation should be reasonably small. Even though evaluation of the presented microscopic model is relatively fast, it may still be too slow for real-time applications of metaheuristics or for running multiple experiments (simulating 10 minutes of traffic on the whole road network of Warsaw with 42000 vehicles takes about

30 seconds on standard machines [1]). Consequently, an avenue worth exploring was the development of a model that balances computational efficiency and accuracy. This approach culminated in the creation of a mesoscopic model described in Section 5.4.

5.4 Mesoscopic traffic model

The mesoscopic model in TSF was designed and implemented for the purpose of building a surrogate model approximating outcomes of the microscopic model presented in Section 5.3. The model is not time-based, but event-based, which means that it does not specify the positions and speeds of cars at each time step. Instead, it estimates the time of drive (T_{EST}) on the road between 2 neighboring nodes on the path, based on the geographical distance between nodes (D) and the default maximum speed on this road segment (V_{MAX}): $T_{EST} = a \cdot \frac{D}{V_{MAX}}$, where a is a parameter that should be set based on comparison to real traffic data or outcomes of another reference model. Then, if a car has to wait at a red signal, the total time of drive is increased by the time to the next switch from the red signal state to the green signal state. Based on the description in Section 4.2.1.1.2, the model is closest to cluster models, as all cars traveling through the same road segment have the same speed.

This is a relatively simple mesoscopic model, it does not take into account a speed reduction caused by large traffic density (in fact, it is not straightforward to estimate traffic density on every road segment at an arbitrary moment of time) or queuing models (queue formation and dissolution). As explained in Section 8.3.1.2 and in [131], this mesoscopic model is not able to provide sufficiently accurate estimations of outcomes of the microscopic model 5.3 to consider it as a sufficient surrogate model, but there is a potential for further improvements and developing somewhat more complex and more accurate (but probably slower to evaluate) mesoscopic models.

5.5 Implementation of the models

The traffic simulation model described in Sections 5.3 and 5.4 were implemented in the tool named Traffic Simulation Framework (TSF).

It allows the simulation of the motion of about 10^6 vehicles on realistic urban networks in real-time using standard computing machines.

[1] Processor Intel(R) Core(TM) i7-4702MQ CPU @ 2.20GHz, 32GB RAM

The system was developed in *C#* language using .NET technology and can be run on computers with installed .NET Framework [229] in version 2.0 or higher (on Windows systems) or Mono.NET [242] in version 2.0 or higher (on Linux systems or macOS). Maps and road network descriptions come from the OpenStreetMap project [259]. By default, TSF allows to simulate traffic on the road network of Warsaw, but it is relatively easy to adapt the system to any other real road network, it requires substituting the maps and road network description, as well as calibrating the traffic model using real traffic data (the traffic model or TSF's implementation do not have to be changed).

5.5.1 Functionalities of the TSF

In this subsection, the following features of the TSF system are briefly presented:

- Graphical User Interface (Section 5.5.1.1);
- Simulating vehicular traffic (Section 5.5.1.2);
- Generating routes for drivers (Section 5.5.1.3);
- Editing settings of traffic signals (Section 5.5.1.4);
- Editing distributions of start points and destination points (Section 5.5.1.5);
- Specifying monitored streets and areas (Section 5.5.1.6);
- Saving simulation data (Section 5.5.1.7).

5.5.1.1 Graphical User Interface

One of the most characteristic features of the TSF system is a comprehensive Graphical User Interface (GUI). The main window of the application contains a city map that can be zoomed in, zoomed out, and moved around (Figure 5.1).

In addition, the user may specify elements displayed on the map. It is possible to display the following information:

- Locations and speeds of cars;
- Locations and states of traffic signals;
- All streets and all types of roads;

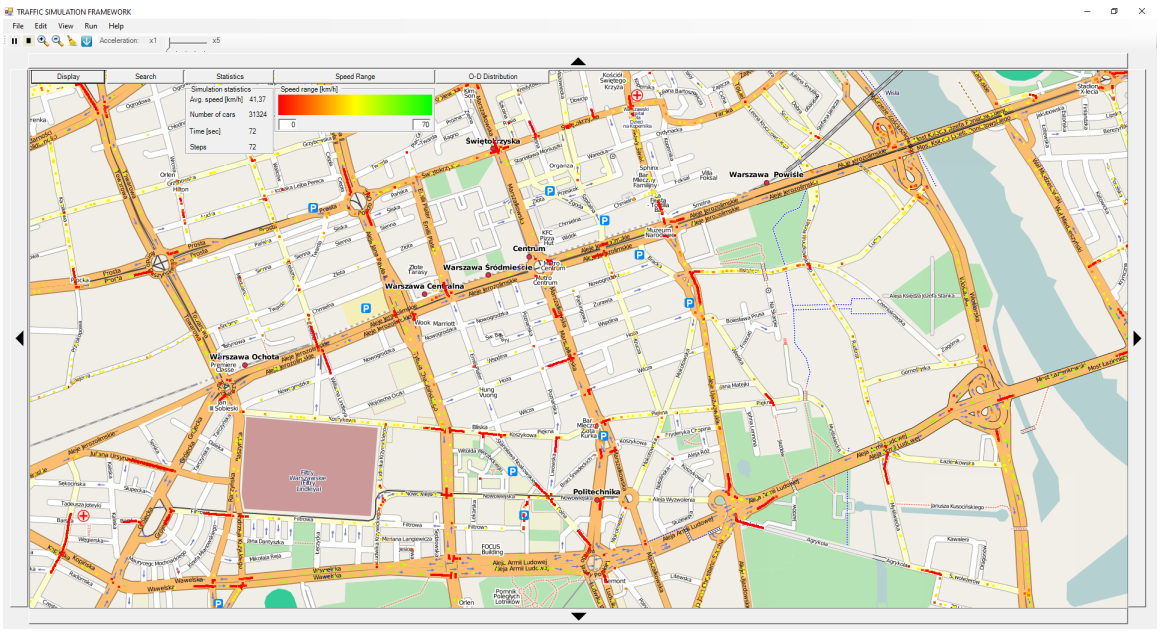


Figure 5.1: The main window of the TSF system with monitoring traffic simulation.

- All nodes of the road network;
- Average speeds of cars on every street;
- Distributions of start and destination points;
- Streets and regions monitored during simulations.

5.5.1.2 Simulating vehicular traffic

The main functionality of TSF is simulating realistic vehicular traffic. A simulation process is based on the TSF model described in section 5.3. Currently, the system is capable of simulating the movement of approximately 10^6 vehicles in real-time on standard machines.

Before the start of the simulation, the user may specify the initial number of cars, the length of a single time step, the number of cars that start driving after a specified number of steps, and some additional parameters related to the driver's behavior (i.e., $turningParameter$ and $intersectionParameter$ which are used in the TSF model) and other parameters. These values can be set using a dedicated window (Figure 5.2).

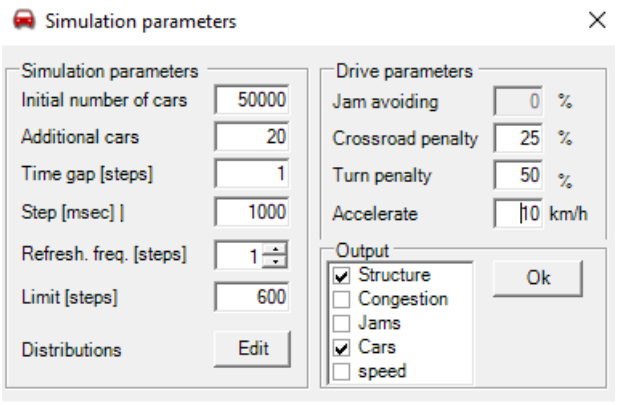


Figure 5.2: The window for specifying values of simulation parameters.

During the simulation, it is possible to monitor traffic conditions and the locations of all cars. Vehicles are represented as dots on the map and their colors correspond to their current speeds (Figure 5.1). It is also possible to monitor the current statistics like average speed for all road segments. More detailed statistics can be saved to external files that can be later analyzed after the simulation is finished.

5.5.1.3 Generating routes for drivers

All vehicles that participate in the simulation have specified default routes. A single route is a path (represented as a sequence of nodes) in the road network represented as a directed graph $G = (V, E)$. The first element of that sequence is always a start point and the last element is a destination point. The process of selecting these points is explained in the trip generation and trip distribution steps of the transport model described in Section 5.2.

The user is able to calculate any number of routes before the start of the simulation. It is also possible to save calculated routes and read them later from external files.

TSF assumes by default that drivers generally choose routes that are optimal in terms of the time required to reach a destination point. Every edge in graph G represents some real-world street (or part of a street) which is characterized by a distance and type. Each type of road has its own default maximal speed (which can also be modified by the users), so it is possible to calculate the default time required to cover any street. Thanks to that, calculating routes can be reduced to finding a default fastest path between 2 nodes in the graph and it could be calculated in many ways. In TSF, the A^* algorithm with a heuristic related to the real-world distance between nodes

is implemented.

It is also worth noting that TSF has an implemented option to take into account not default travel times, but realistic travel times that are the outcomes of simulations. This can be particularly useful in cases where routes are generated dynamically, on the fly, during the simulation. However, this option was not used in the experiments performed for the purpose of this thesis.

5.5.1.4 Editing settings of traffic signals

Traffic signals are very important in the TSF model and have also a great impact on real traffic. The TSF's input files contain information about the locations of traffic signals and their settings. The locations can be obtained from the OSM data. The default settings are loaded from a configuration file (prepared by a user) at the start of the simulation but can be later modified from the GUI's level.

In TSF, all traffic signals are characterized by the following attributes:

- duration of the green phase (in simulation steps),
- duration of the red phase (in simulation steps),
- initial phase (red/green),
- offset - number of simulation steps to switch a phase.

TSF provides tools for adding, modifying, and deleting traffic signals on the city map. All these actions can be performed just by clicking on a map and providing attributes, like in Figure 5.3.

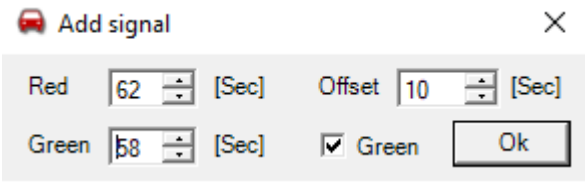


Figure 5.3: A form to provide traffic signal settings.

5.5.1.5 Editing distributions of start points and destination points

As discussed in Section 5.2, TSF divides the entire city map into zones. The trip generation and trip distribution steps can be performed using the OD

matrices read from the configuration files, but the zones can also be defined differently. For example, in the case of Warsaw, these are by default 40×40 square regions, each representing a $500m \times 500m$ real-world square. In the default setting, each zone has attributes $rank_{start}$ and $rank_{destination}$ that take values from the set $\{0, 1, 2, 3, 4\}$. The $rank_{start}$ parameter determines the probability that the zone will be selected as the start zone for a newly generated trip. Similarly, values of the parameter $rank_{destination}$ implicate the probability that the zone will be selected as the destination zone.

The probability that a zone z will be selected as a start zone is equal to $\frac{rank_{start}(z)}{\sum_{w \in Z} rank_{start}(w)}$, where Z is a set of zones. By analogy, the probability that a zone z will be selected as a destination zone is equal to $\frac{rank_{destination}(z)}{\sum_{w \in Z} rank_{destination}(w)}$.

The values of these parameters can be read from configuration files, but it is also possible to edit them manually from the GUI level by marking chosen zones on a map, as presented in Figure 5.4.

When generating new routes for cars, after selecting a zone as an origin zone, TSF selects a node from that zone, giving all nodes in that zone the same probability of being selected. The same procedure is used to select a destination node from the selected destination zone.

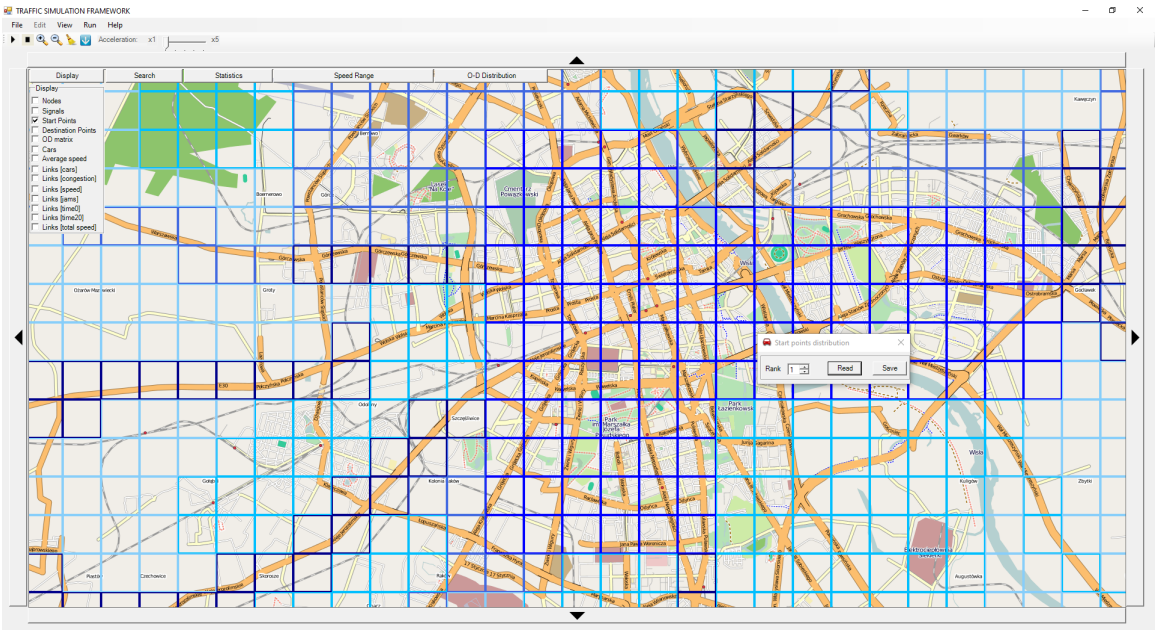


Figure 5.4: A module for specifying the distribution of start zones in TSF. The darker the color of the zone border, the higher the probability of selecting a point from that zone as the starting point.

5.5.1.6 Specifying monitored streets and areas

TSF provides the ability to select edges of a road network to be monitored during the simulation. Then, in each simulation step, some data from these edges can be collected and potentially saved to external files.

It is possible to select any single edge in the graph by marking its end nodes on a map. The user can also select all edges from a larger area by marking an appropriate rectangle on a map. The simulator's GUI allows the users to modify selected areas, give them names, display them on a map, or delete them from the set of monitored regions. Figure 5.5 shows an example set of monitored edges specified using TSF's GUI.

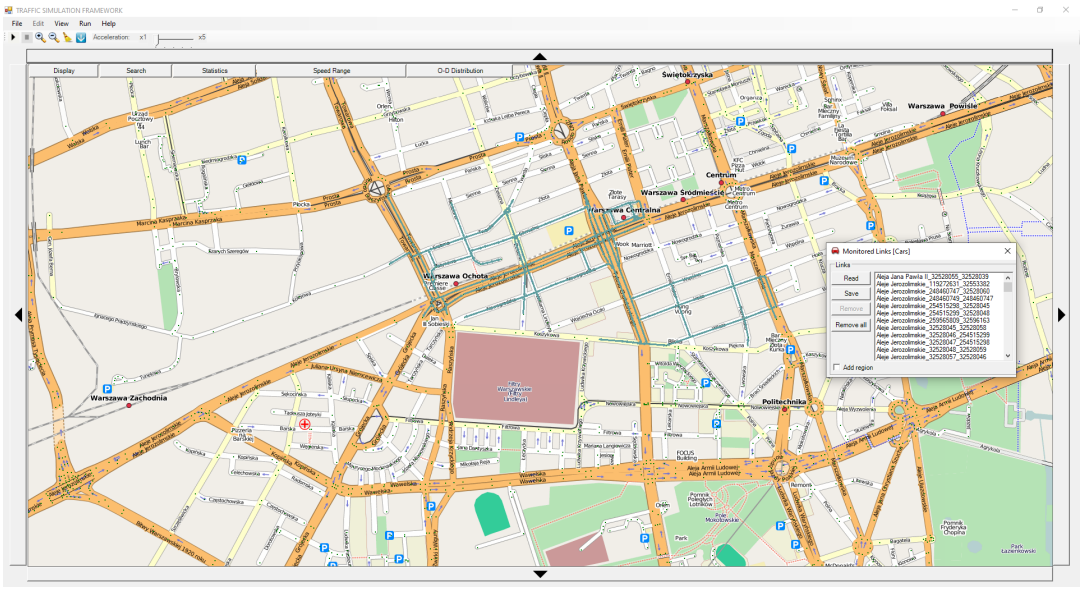


Figure 5.5: An example set of monitored edges visible in the TSF's GUI.

5.5.1.7 Saving simulation data

The main purpose of monitoring parts of the road network is to collect data from simulated traffic. Specified data is stored in files and can be processed later by external tools.

TSF can output information about "static" elements of the road network (graph description, initial synchronization of traffic signals) as well as information about the motion of all cars within the monitored regions. In every simulation step and for every car from selected areas, TSF can save the following information to external files:

- $Time$: the current simulation step;
- Car_{ID}: the vehicle's identifier;
- $Edge_{ID}$: identifier of the edge (road) where the car is located;
- $Lane_{ID}$: identifier of the lane of edge $Edge_{ID}$ where the car is located;
- $Location$: distance between the car and the beginning of the edge $Edge_{ID}$;
- $Speed$: the current speed of the car.

5.6 Compatibility with real-world traffic

In order to ensure compatibility with real-world traffic, both the travel demand model and the traffic model need to be built based on real-world data and need to be calibrated in order to correspond well to real-world conditions. TSF expects to load road network descriptions from the OpenStreetMap [259], which provides a geographic database of the Earth (including road networks) that can be road networks) that can be edited through open collaboration. Even though it is not fully consistent with the real road network, and in some areas (especially underdeveloped areas) a lot of information is missing, it gives a fairly good approximation of the real road network, especially in urban areas of developed countries. The current road network of Warsaw in OpenStreetMap is already well-developed, but the road network version that was used in experiments was downloaded and prepared in 2009, so some inconsistencies exist.

The default trip generation configuration was built in 2009 for the division into 40×40 zones and was based on statistical data collected from GUS [114]. In the initial trip distribution it was assumed that the number of trips between a pair of zones is proportional to the frequency scores assigned to both zones based on the trip generation output. For trip assignment (implemented using the A* algorithm) and the traffic model, there was no analysis of compatibility with real-world traffic performed, as there was no such data (like traffic volume measurements and real trajectories of drive in Warsaw) available to the author of the thesis at the time of building the initial model. It cannot be definitively asserted that the initial version of TSF generates realistic traffic data, as there is no empirical evidence to support this claim. However, feedback from Warsaw residents who frequently drive in the city indicated that the TSF accurately replicates traffic congestion patterns and is consistent with observed traffic conditions in real-world locations [355].

The later works on TSF carried out in collaboration with W. Chmiel ([58]) introduced zones compatible with the zones used in traffic analysis by road authorities in Warsaw that were designed based on the results of the traffic study carried out in Warsaw in 2005. These designs were shared by urban road authorities in Warsaw with the author of TSF along with OD matrices from 2005 and 2010 and traffic volume measurements from 2012 and 2013. Thanks to that, it was possible to carry out calibration of the OD matrices to mimic traffic conditions based on traffic volume measurements from 2012 and 2013. As a result, it was possible to achieve values of the GEH statistic (which is a measure used in the field of transportation planning and traffic engineering to compare observed traffic counts with model-estimated traffic flows [334]) below 5 for the 85th percentile of measurement points[58]. It was also concluded that the main sources of differences were the outdated OSM data and the lack of the trip matrix and traffic volume data from the same year.

It is important to acknowledge that the results obtained from research conducted using TSF may not be directly applicable to real-world situations. However, this is the case for all existing traffic models as none of them can reproduce real traffic with 100% accuracy. All results of experiments performed with traffic models need to be carefully analyzed before implications for real-world traffic scenarios can be drawn.

However, the primary objective of this thesis is to develop novel, universal methodologies for optimizing complex processes, such as urban road traffic, rather than identifying specific traffic signal settings or algorithms for immediate real-world implementation. It is essential that the models employed in these experiments exhibit a reasonable degree of accuracy, allowing the developed methods to be applied to real-world scenarios following appropriate calibration and availability of high-quality data. The current stage of TSF's development appears to fulfill these criteria, as the underlying models are informed by existing knowledge in transportation and traffic modeling and engineering. Additionally, the fundamental traffic model is based on a cellular automaton, which is also a complex system. Moreover, in TSF, it is possible to edit the road network structure or traffic signals on order to simulate different traffic conditions and changes in the road network (e.g., caused by car accidents or road works). Consequently, it was assumed that the state of TSF was adequate for conducting experiments aimed at devising new algorithms for optimizing complex processes.

5.7 Applications of TSF

It is important to note that despite TSF's limited compatibility with real-world traffic data, it has found applications beyond the scope of the research conducted for this thesis. One contributing factor is that existing microscopic or mesoscopic simulators often exhibit high complexity, are challenging to use, and are designed for general traffic engineering purposes, requiring calibration due to the multitude of parameters involved (and usually, the more complex the tool is, the more parameters are involved and the more difficult it is to calibrate the model). These tools are suitable for typical, engineering applications, but adapting these tools for atypical applications can be difficult.

In contrast, TSF's code was readily available to its author, making it relatively straightforward to modify the tool to meet specific requirements in various research projects aiming to investigate atypical scenarios and innovative ideas. This is also one of the reasons why TSF has found several applications in research works carried out by the TSF's author and other scientists.

This section highlights a selection of TSF's past applications in which the author was directly involved and can provide an accurate description of the use cases (it is important to mention that a demo version of TSF is publicly available, which could potentially result in its application to other use cases beyond the knowledge of TSF's author, cf. Appendix B). Given that the development of TSF is a by-product of the research associated with this dissertation, it is relevant to discuss its current applications and potential future uses.

5.7.1 Traffic Prediction Contest

"The best way to predict the future is to create it" A. Kay

One of the first applications of TSF was generating large data sets for the contest on traffic prediction: *IEEE ICDM 2010 Contest TomTom Traffic Prediction for Intelligent GPS Navigation* [355]. The contest was organized in 2010 by researchers from the University of Warsaw and TunedIT Solutions (with the support of TomTom International BV) as a side event of the International Conference on Data Mining 2010 [162]. The challenge was organized in the form of an interactive online competition consisting of three distinct tasks:

1. Traffic congestion prediction, in an elementary setup of time series fore-

casting: a series of measurements from 10 selected road segments were given and the goal was to make short-term predictions of future values based on historical ones.

2. Modeling the process of traffic jam formation during the morning peak in the presence of roadworks, based on initial information about jams broadcasted by radio stations. Input data contained identifiers of road segments closed due to roadwork, accompanied by a sequence of segments where the first jams occurred. The algorithms had to predict a sequence of segments where the next jams will occur in the nearest future.

3. Traffic reconstruction and prediction based on real-time information from individual drivers. Input data consisted of a stream of notifications from 1% of vehicles about their current locations in the city road network, sent every 10 seconds. The algorithm received this stream and had to predict the traffic congestion on selected road segments for the next 30 minutes. Large volumes of data were involved in this task, requiring the use of scalable data mining methods.

Competition datasets were generated by TSF which was run for many hours to generate gigabytes of traffic data.

The contest attracted 575 participants (both teams and individuals), of whom over 100 submitted solutions, most of them several times: the total number of solutions was nearly 5000. Best algorithms achieved nearly 3-fold improvement over baseline solutions in predicting traffic congestion and jams. As tasks were independent, anyone could participate in all of them or in a chosen one. The winning solutions were summarized in research papers and presented at a workshop organized during the ICDM 2010 conference in Sydney [162]. The contest was summarized in [355].

5.7.2 Acquisition of traffic-related knowledge by interaction with domain experts

In 2013-2015, TSF was used to acquire traffic-related knowledge by interaction with domain experts. The goal of this research was to learn conceptual levels of traffic congestion and a concept of a traffic jam at a single intersection by means of a dialog with experts.

First, 51 traffic simulations corresponding to different traffic situations close to the intersection of streets “Banacha”, “Grójecka”, “Bitwy Warszawskiej 1920 r.” were prepared using TSF (the area under investigation is

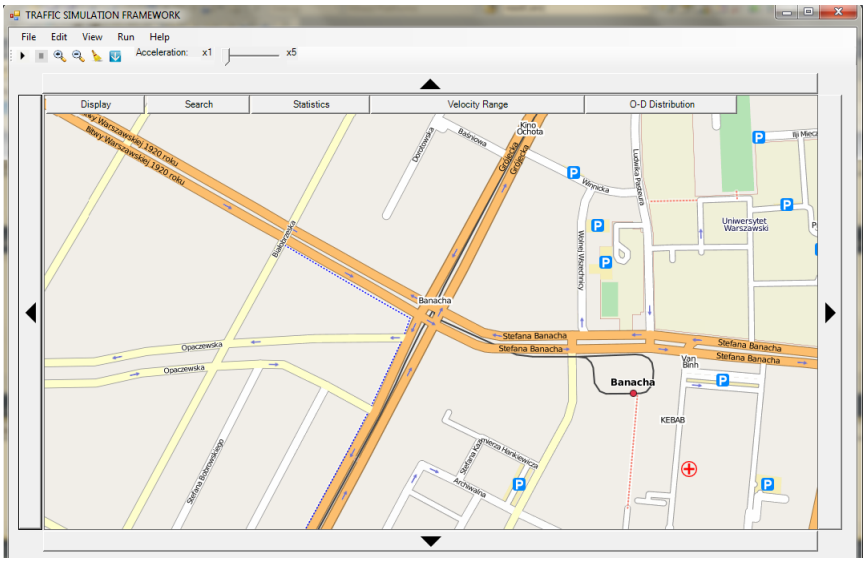


Figure 5.6: Visualization of the investigated intersection in the TSF system.

presented in Figure 5.6, it is a place where large traffic congestion occurs very often).

All scenarios represented 10-minute traffic simulations, with some parameters held constant across all scenarios. However, the scenarios varied in terms of the following parameters:

1. Initial number of cars (*NrOfCars* parameter);

2. Number of new cars that start driving in each simulation step (*NewCars* parameter);

3. Start and destination points distributions.

Five different distributions of start points ("From East", "From West", "From North", "From South", "Uniform"), 5 different distributions of destination points ("To East", "To West", "To North", "To South", "Uniform"), and 3 different values of the NewCars parameter (1, 3, 5) gave 75 possible simulation scenarios. 48 of them were selected for experiments assuming $NrOfCars = 100$. In addition, for $NrOfCars = 1000$ and a uniform distribution of start points and destination points, 3 more situations were generated with 3 different values of the NewCars parameter. It gave in total 51 traffic situations that were later simulated using TSF.

Every simulation was "recorded" - TSF logged to the output files the following data:

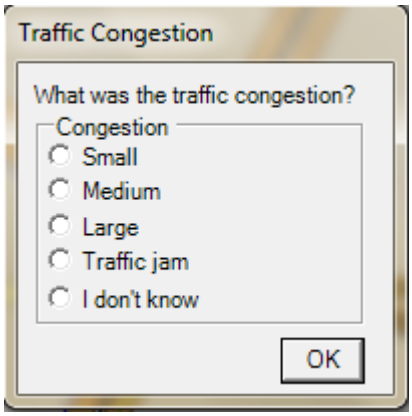


Figure 5.7: Window shown to experts after every video.

- Timestamp (simulation step);
- Current car's position (link in the road network, position within the link, geographical longitude and latitude);
- Current car's speed (in km/h).

The acquired information enabled the reconstruction of traffic scenarios, which were then presented as short "visualizations" (videos) to experts assessing traffic conditions. It was assumed that a single traffic signal cycle had a constant duration of 2 minutes for all signals. Therefore, each 10-minute scenario was composed of five parts, each lasting 2 minutes and corresponding to a single traffic signal cycle. In total, it gave 255 traffic cases. Each of them was evaluated by domain experts and their task was to provide information about the traffic state in the area close to the considered intersection (in this context, a domain expert could be anyone with experience in observing or participating in Warsaw city traffic). 1 of 51 situations was analyzed by all experts, while every situation from the remaining 50 was analyzed by 3 experts. After the presentation of a particular video, TSF displayed the question: *What was the traffic congestion?*. Experts were prompted to respond to the question by selecting one of five available options: *Small*, *Medium*, *Large*, *Traffic jam*, *I don't know*. The answer was given by experts using the window presented in Figure 5.7. If the experts selected *I don't know* response in the first window, the system asked for checking the closest options by displaying the window presented in Figure 5.8. Subsequently, the system prompted experts to provide a justification for their response in natural language. Upon verifying the chosen answer and submitting the justification, the next video was presented to the expert.

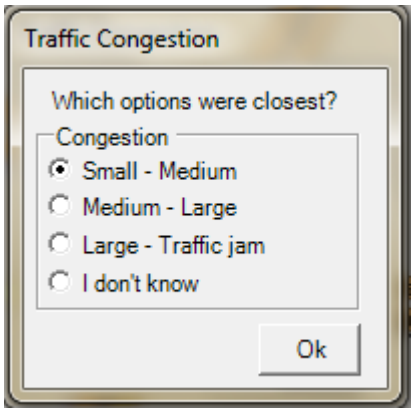


Figure 5.8: Window for submitting two closest options.

The acquired answers from experts were later evaluated in 2 ways: expert-oriented and case-oriented. In the expert-oriented evaluation, the consistency of decisions made by a given expert was checked: from every situation, 2 cases were selected to be labeled by an expert twice. In the case-oriented evaluation, it was analyzed how a given case is labeled by different experts. For this purpose, each case was labeled by 3 different experts. Also, justifications of experts provided in natural language were analyzed and some additional traffic properties (used by experts to make decisions) were identified. These properties were later used to design a new series of experiments.

The experts' answers as well as the results of the evaluation of their consistency could be later used to assign proper labels regarding how different traffic situations are perceived by travelers, which can find application in, e.g., traffic information systems [354]. In addition, machine learning methods detecting high-level traffic states (e.g., *traffic jam*) based on low-level traffic data can also be used to trigger the process of reconfiguring traffic signal settings (e.g., using metaheuristics). The details of this research can be found in [133] and [345].

5.7.3 Modeling mobility and visualizing people's flow patterns in rural areas for future infrastructure development

TSF was also used in a cross-border mobility study conducted in support of the infrastructure development efforts of local authorities and NGOs in the area over the Kayanga-Geba River, at the border between Senegal and Guinea Bissau (visualized in Figure 5.9). A mobility model based on the 4-

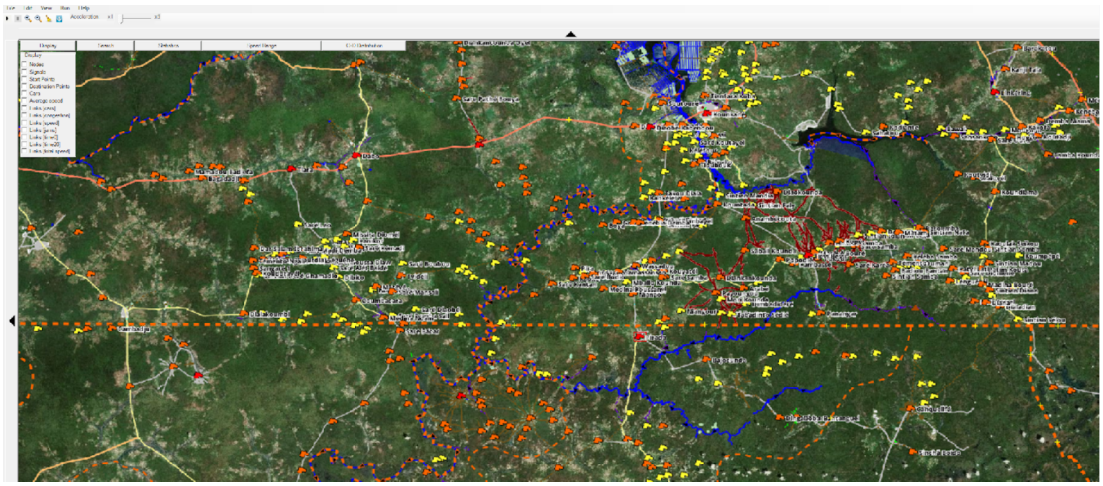


Figure 5.9: GUI of TSF applied to modeling mobility in the studies area at the border between Senegal and Guinea Bissau.

step model was built for the considered area and implemented in TSF, which was later used to calculate origin-destination matrices for the studied regions in two cases: with and without cross-border mobility.

During the research, there was no access to information about possible modes of transport or the exact routes (there was no such data even in OpenStreetMap or Google Maps). Consequently, the decision was made to model mobility using only the initial two steps of the four-step model, which facilitated the construction of origin-destination matrices for the study area utilizing the gravity model [87, 55]. Nonetheless, with information on potential routes available, it would be feasible to calculate travel distances considering both scenarios, with and without cross-border mobility. This could aid in identifying optimal locations for new infrastructure development, such as schools, hospitals, or bridges.

The details of this research can be found in [34].

5.7.4 Alternative and Prospective Future Applications

Even though TSF is still under development and its correspondence with real traffic can be still improved, it has already found applications in various domains.

Besides the aforementioned application of TSF, the tool has been already used in several other research works carried out by other scientists. For example, datasets generated by TSF were used to evaluate various traffic

prediction algorithms [291] or to detect traffic states [330]. Previous attempts have been made to apply TSF to logistics optimization tasks, such as solving the Vehicle Routing Problem [73] and its variants. Furthermore, there have been efforts to simulate traffic involving connected and autonomous vehicles (CAVs) and to determine optimal driving strategies for such vehicles. However, the results of these research works have not been published yet.

In future applications, TSF could be utilized to examine the impact of CAVs on real-world traffic or to identify optimal locations for parking or electric vehicle charging stations. Broadly, TSF has the potential to contribute to urban design, as well as address planning and management challenges that urban and traffic engineers currently face or will need to tackle in the near future, particularly in the era of connected, autonomous, electric, and shared transportation [122, 124].

The relevance of TSF is expected to increase, especially on a large scale and in the context of CAVs, as it is easier to ensure high compatibility with real-world traffic when vehicles are controlled by computer programs that can be accurately simulated. Additionally, the efficient implementation of a relatively simple traffic model in TSF allows for effective large-scale traffic simulations using a microscopic model. Given the anticipated rise in demand for new traffic analyses to optimize traffic management and transportation infrastructure in the era of CAVs and electric/shared mobility, it is expected that there might be more real-world applications of TSF. There are also ongoing discussions and design works aiming to incorporate TSF as one of the tools within the Polish laboratory for intelligent transportation systems, SmartCity Lab, established in Chełm (in Poland) in March 2023 [312].

It is also worth mentioning that the research work aimed at developing the Traffic Simulation Framework and its underlying microscopic model was awarded the "LIDER ITS 2015" prize for the best R&D work in the field of Intelligent Transportation Systems in Poland in 2015 [205].

Chapter 6

Metaheuristics and other optimization algorithms

This chapter presents an overview of metaheuristics and other popular optimization algorithms, specifically emphasizing the ones employed in the experiments described in this thesis. The focus is on the categorization of metaheuristics and the description of specific algorithms, including their pseudocode, hyperparameters, and applications, especially in optimizing complex real-world processes.

6.1 Introduction

There already exist many approaches to solving optimization problems, ranging from algorithms finding the exact solutions to approaches focusing on finding near-optimal solutions. Some of the existing optimization techniques are linear programming, quadratic programming, interior-point method, trust-region method, conjugate-gradient methods, AI-based techniques, heuristics, and metaheuristics [92]. All of them possess advantages and disadvantages and the suitability of particular techniques usually depends on the particular optimization problem, its complexity, the size of its instances, as well as constraints related to computational power and other resources. As explained in Section 1.1, in the case of complex processes, the evolution of qualities of possible solutions can be time-consuming, due to the complex systems property of computational irreducibility, so finding the exact solution is usually time-consuming. In addition, due to the limited accuracy of mathematical models of real-world processes, sometimes it does not make sense to search for the exact solution, as the solution that is optimal in a mathematical model does not have to be optimal in the real-world setting. Therefore, applying

heuristics to optimize complex processes seems to be a natural approach.

As stated in [267], heuristics bring a balance of *good* solutions (relatively close to global optimum) and affordable time and cost and have proven to be a comprehensive tool to solve hard optimization problems. However, heuristics are usually based on specific characteristics of the considered problem, which means that they cannot be considered a universal tool, and their adaptability to other problems is limited. Metaheuristics were designed to overcome this drawback. They are problem-agnostic algorithms but can be adapted to incorporate problem-specific knowledge.

Metaheuristics are optimization algorithms used to find near-optimal solutions to complex optimization problems [218, 103]. Unlike traditional optimization algorithms, which rely on mathematical formulas to find the optimal solution, metaheuristics use heuristics to guide their search for solutions. This makes them well-suited for solving problems in which the optimal solution is difficult or impossible to determine using classical mathematical models. Metaheuristics are also often used when the optimization problem has too large size or is too complex to be solved by exact methods in a reasonable time.

In order to formally formulate a metaheuristic, it is necessary to consider a fitness (objective) function F evaluating the quality of solutions, a stopping criterion C that determines when the algorithm should stop searching for a better solution (e.g., a maximum number of iterations, a minimum improvement threshold) and a transition rule mechanism T that guides the search from the current solution to a new solution, e.g., using randomness or adaptive techniques to balance exploration and exploitation.

The aforementioned exploration and exploitation are two fundamental concepts in the context of metaheuristics [218]. Exploration refers to the process of searching the solution space in a global and broad manner, focusing on discovering new regions that have not been visited before. Exploitation, on the other hand, refers to the process of intensively searching within a specific region of the solution space, focusing on refining and improving the current best solutions. Too much exploration may lead to an inefficient search, as the algorithm may not sufficiently focus on promising regions. On the other hand, too much exploitation may cause the algorithm to get stuck in local optima, preventing it from finding better solutions in other regions of the search space. Therefore, a good metaheuristic algorithm should strike a balance between exploration and exploitation to ensure that it can effectively search the solution space and find high-quality solutions.

Algorithm 2 presents a general description of a metaheuristic algorithm.

There is a wide variety of metaheuristics and a number of properties by which they can be classified. The following subsections contain examples

Algorithm 2 General Metaheuristic Algorithm

Require: Fitness function F; stopping criterion C; transition rule T;
1: Initialize the search space S and the objective function F;
2: Generate the set of initial solutions $X \subset S$;
3: Evaluate the objective function F on elements from X;
4: Set x_{best} to the best solution (with respect to F) from X;
5: **while** stopping criterion C is not met **do**
6: Generate a set of new candidate solutions X' using the transition rule T from the current set of solution X;
7: Evaluate the objective function F on elements from X';
8: Replace the current set of solutions X with X';
9: **if** X contains a better solution than x_{best} **then**
10: Update x_{best};
11: **end if**
12: **end while**
13: **return** x_{best};

of popular classifications of metaheuristics (based on [30], [362]). They are then followed by sections summarizing some important metaheuristics and optimization algorithms that were applied in the research presented in this thesis. Each section contains a description of the algorithm and presents its pseudocode and hyperparameters (tunable parameters used to control the algorithms and set before their execution). A comprehensive overview of metaheuristics can be also found in, e.g., [92, 267, 218, 61].

6.1.1 Local search vs. global search

Local search algorithms aim to improve a solution incrementally by making small changes to it. The algorithm moves from one solution to another in the vicinity of the current solution in the hope of finding a better one. An example of local search algorithms is Hill Climbing (Section 6.8).

On the other hand, global search algorithms aim to find the global optimum by exploring a large region of the search space. Genetic Algorithms (Section 6.2), CMA-ES (Section 6.6) and Particle Swarm Optimization (Section 6.3) are examples of global search algorithms.

The choice between local search and global search algorithms depends on the nature of the optimization problem being solved. In general, local search algorithms are well suited for problems where the solution space is smooth and unimodal (i.e., there is only one solution that provides the best possible outcome for the given problem, and all other solutions are relatively inferior),

while global search algorithms are better for problems with multimodal solution spaces (in which there are multiple optimal solutions) or a high degree of randomness.

6.1.2 Individual-based vs. population-based

Individual-based metaheuristics work with a single candidate solution at a time and update this solution to improve it [30]. The algorithms typically use local search methods to incrementally improve the candidate solution. These approaches can be more efficient in finding the optimal solution (compared to population-based approaches), but they may be limited in their ability to explore the solution space and find multiple, potentially optimal solutions. Examples of individual-based metaheuristics are Hill Climbing (Section 6.8) and Simulated Annealing (Section 6.5).

On the other hand, population-based metaheuristics, maintain a set of candidate solutions (population) and use this population to guide the search for the optimal solution [30]. The population can be evolved over time through various operations, such as selection, crossover, and mutation, to generate new candidate solutions. The population-based approach allows for the exploration of the solution space and enables the algorithms to find multiple, potentially optimal solutions. Examples of population-based metaheuristics are Genetic Algorithms (Section 6.2), CMA-ES (Section 6.6), and Particle Swarm Optimization (Section 6.3).

Population-based metaheuristics are more suitable for problems with complex, multimodal solution spaces, while individual-based metaheuristics are more appropriate for problems with simple, smooth solution spaces.

6.1.3 Deterministic vs. stochastic

Deterministic metaheuristics are algorithms that always produce the same result given the same input and conditions, whereas stochastic metaheuristics are algorithms that involve a random component and may produce different results each time they are run with the same input and conditions.

Stochastic metaheuristics are often preferred for solving optimization problems because they are able to escape local optima and find the global optimum more efficiently than deterministic metaheuristics. Some commonly used stochastic metaheuristics are Genetic Algorithms (Section 6.2) and Simulated Annealing (Section 6.5).

On the other hand, deterministic metaheuristics are generally simpler and easier to understand, and they can often be used as building blocks for more complex algorithms. A commonly used deterministic metaheuristic is

Hill Climbing (Section 6.8) assuming that the initial state is given. However, in this case, the initial states are often generated randomly which can significantly impact the outcome.

6.1.4 Dynamic vs. static objective function

Usually, metaheuristics keep the objective function given in the problem representation static ("as it is"), but some algorithms, like Guided Local Search [342] modify the objective function during the search [30]. The motivation behind this approach is to escape from local minima and plateaus by modifying the search landscape. Accordingly, during the search, the objective function can be dynamically altered by trying to incorporate information collected during the search process.

6.1.5 Single-objective vs. multi-objective

Single-objective optimization involves finding the optimal solution for a single objective or criterion, such as minimizing the cost of a project, maximizing the profit, and minimizing the time of travel [362].

On the other hand, multi-objective optimization involves finding the optimal solution for multiple, often conflicting, objectives [362]. The goal is to find a set of solutions that are optimal with respect to multiple criteria, such as minimizing cost and maximizing profit. In multi-objective optimization, there is usually no single solution that is optimal for all objectives, and the optimization process results in a set of trade-off solutions that represent different compromises between the objectives.

Multi-objective optimization algorithms can be further classified into two categories: Pareto-based and non-Pareto-based [12]. Pareto-based algorithms aim to find the Pareto front, which is a set of non-dominated solutions in which no objective can be improved without negatively affecting other objectives. Non-Pareto-based algorithms aim to find a single solution that is a compromise between the objectives, e.g., by using a function that maps multiple objectives to a single scalar value.

6.1.6 Continuous optimization vs. discrete optimization

Continuous optimization and discrete optimization refer to the type of variables being optimized in a problem [103]. In continuous optimization, the variables are real numbers and the solution space is continuous. In discrete

optimization, the variables are discrete, meaning they can only take certain values from a finite or countable set [103].

For continuous optimization, some popular metaheuristics include Particle Swarm Optimization (Section 6.3) and CMA-ES (Section 6.6).

For discrete optimization, some popular metaheuristics are Simulated Annealing (Section 6.5) and Genetic Algorithms (Section 6.2).

In the case of complex systems considered in this dissertation, optimization problems are usually defined as discrete optimization problems, but due to applications of surrogate models (which are continuous), continuous optimization is applied as well.

6.2 Genetic algorithms

6.2.1 Algorithm's description

Genetic algorithms (GAs) are metaheuristics based on the mechanisms of natural selection and genetics [236]. They are often used for solving discrete optimization problems in different fields of computer science and engineering, from scheduling and routing problems to design and parameter tuning problems.

The basic idea behind GAs is to encode potential solutions to a problem as chromosomes in a population and to evolve this population over time through the application of genetic operators: selection, crossover, and mutation. These operators (and their hyperparameters), together with the size of the population and stopping criterion, are considered as hyperparameters of GAs.

6.2.2 Algorithm's hyperparameters

The key hyperparameters of GAs are [236, 74]:

- The size of the initial population: It determines how many chromosomes (encoding potential solutions) should be considered in a single population/iteration of the algorithm.
- Stopping criterion: It determines when the genetic algorithm should be stopped. It can be based on, e.g., the number of iterations, the quality of the found solution, or the pace of convergence.
- Selection operator: It specifies the process of choosing the best chromosomes from the current population to serve as parents for the next generation.

- Crossover operator: It specifies the process of combining the genetic information of two parent chromosomes (chosen from the chromosomes selected from the previous population using a selection operator) to create new offspring chromosomes.
- Mutation operator: It specifies the process of randomly changing the genetic information of a chromosome.
- Elitism: The number of top-performing solutions that are preserved from one generation to the next. This hyperparameter controls the trade-off between exploration and exploitation in GAs. By preserving the best solutions, elitism ensures that the algorithm does not lose sight of promising regions of the solution space, while still allowing for the exploration and improvement of candidate solutions.

Selection is the process of choosing the best chromosomes from the current population to serve as parents for the next generation. The best chromosomes are typically chosen based on their fitness, which is a measure of how well they solve the problem at hand. The most popular selection operators used in GAs and in this dissertation are:

- Roulette wheel selection (or fitness proportional selection): This operator assigns a probability to each chromosome based on its fitness, and a chromosome is then selected based on this probability. The chromosomes with higher fitness have a higher probability of being selected.
- Tournament selection: This operator involves running several "tournaments" among a few chromosomes chosen at random from the population. The winner of each tournament is selected for crossover. There might be several ways of selecting the winner. It can be the one with the best value of the fitness function or the decision can be non-deterministic, e.g., based on the roulette wheel selection. Another option is to order the chromosomes into a ranking according to their fitness function and select the chromosome from the position $rank$ as a winner with a probability $(1 - P)^{rank-1} \times P$, for a given probability $P \in (0, 1)$.
- Rank selection: The chromosomes in the population are sorted according to their fitness, and assigned a rank based on their position in the sorted list. Chromosomes are then selected based on their rank, with higher-ranked chromosomes having a greater probability of selection.
- N-best selection: selecting N best chromosomes sorted according to their fitness.

Crossover is the process of combining the genetic information of two parent chromosomes (chosen from the chromosomes selected from the previous population using a selection operator) to create new offspring chromosomes. The most popular crossover operators that are also used in this dissertation are:

- Single point crossover: This operator selects a single point in the chromosomes and exchanges the genetic information of the parent chromosomes beyond that point to create two offspring chromosomes.
- Two-point crossover: This operator selects two points in the chromosomes and exchanges the genetic information of the parent chromosomes between those two points to create two offspring chromosomes.
- Uniform crossover: This operator randomly selects bits or elements from two parent chromosomes to create the offspring chromosomes (typically, each bit is chosen from either parent with equal probability and the operator produces two child chromosomes). This results in a mix of genetic information from both parent chromosomes in the offspring.

Mutation is the process of randomly changing the genetic information of a chromosome. This allows for the exploration of new solutions and helps prevent the population from becoming stuck in a local optimum. Here are some common mutation operators used in genetic algorithms:

- Uniform mutation: Randomly selecting an element in a chromosome and replacing it with a random value.
- Gaussian mutation: Adding a random value generated from a Gaussian distribution to an element in a chromosome.
- Scramble mutation: Randomly rearranging the elements in a portion of a chromosome.
- Inversion mutation: Reversing the order of elements in a portion of a chromosome.
- Swap mutation: Swapping the positions of two randomly selected elements in a chromosome.

6.2.3 Algorithm's pseudocode

Algorithm 3 presents a pseudocode of the Genetic Algorithm.

Algorithm 3 Pseudocode of the Genetic Algorithm

Require: Fitness function F; stopping criterion C; selection, crossover, mutation operators;

1: Initialize the population of candidate solutions P;
2: Evaluate the fitness $F(x)$ of each candidate solution $x \in P$;
3: **while** C is not met **do**
4: Select a subset of the current population using a selection method for further reproduction (optionally, preserve some of the best-performing chromosomes);
5: Apply crossover and mutation to generate a new population R;
6: Replace the old population with R;
7: Evaluate the fitness $F(x)$ of each new candidate solution x from the current population;
8: **end while**
9: **return** the best candidate solution from the final population;

6.3 Particle Swarm Optimization

6.3.1 Algorithm's description

Particle Swarm Optimization (PSO) is a heuristic optimization algorithm inspired by the social behavior of bird flocking or fish schooling [180]. It is a population-based optimization algorithm that seeks to find the global optimum of a search space by updating the positions of a set of particles. Each particle represents a candidate solution to the optimization problem.

In PSO, each particle has a position in the search space and a velocity that reflects its movement from one position to another. The velocity of each particle is updated based on its own best-known position (pbest) and the best-known position of all particles in the swarm (gbest). The new velocity and position of a particle are then updated using a weighted combination of its current velocity, pbest, and gbest. This process is repeated iteratively until a stopping criterion is met.

There are 2 types of updating the positions and velocities of particles [86]:

- Global-best (each particle updates its velocity and position based on the best position found by any particle in the swarm so far);
- Local-best (each particle updates its velocity and position based on the best position it has found so far as well as the global best position found in a group of particles that are local for that particle).

In the case of the global-best update, the velocity of each particle i at time $t+1$ is given by:

$$v_i(t+1) = w \cdot v_i(t) + c1 \cdot r1 \cdot (best_i - x_i(t)) + c2 \cdot r2 \cdot (g_{best} - x_i(t)), \quad (6.1)$$

where:

- $v_i(t)$ is the velocity of particle i at time t;
- w is the inertia weight, which determines how much the particle's velocity is affected by its previous velocity;
- $c1$ is the cognitive acceleration coefficient;
- $c2$ is the social acceleration coefficient;
- $r1$ and $r2$ are random numbers generated between 0 and 1;
- $best_i$ is the particle's own best position found so far;
- $x_i(t)$ is the current position of particle i at time t;
- g_{best} is the global best position found by the entire swarm.

In the case of the local-best update, the update formula for the velocity of each particle i at time $t+1$ is given by:

$$v_i(t+1) = w \cdot v_i(t) + c1 \cdot r1 \cdot (best_i - x_i(t)) + c2 \cdot r2 \cdot (l_{best} - x_i(t)), \quad (6.2)$$

where l_{best} is the local best position found by a group of particles, while the other symbols have the same meaning as for the global-best update.

In both cases, the updated position of each particle i at time $t+1$ is given by:

$$x_i(t+1) = x_i(t) + v_i(t+1). \quad (6.3)$$

PSO is simple to implement, computationally efficient, and has been shown to be effective for a wide range of optimization problems, including continuous optimization, combinatorial optimization, and multi-objective optimization. However, PSO can be sensitive to the choice of parameters and may converge to suboptimal solutions if the parameters are not properly set. Selecting PSO parameters that yield good performance has therefore been the subject of extensive research [304].

6.3.2 Algorithm's hyperparameters

The key hyperparameters of the PSO algorithm are [180, 304, 368]:

- Swarm size (denoted in this thesis as n): The number of particles in the swarm.
- Termination criterion: It defines when the optimization process should stop.
- Cognitive acceleration coefficient (denoted in this thesis as $c1$): It controls the particle's tendency to follow its best historical position.
- Social acceleration coefficient (denoted in this thesis as $c2$): It controls the particle's tendency to follow the swarm's global best position.
- Inertia weight (denoted in this thesis as w): It determines how much the particle's velocity is affected by its previous velocity.
- Way of updating a particle's position in the search space: Global-best or local-best PSO [86].
- Number of neighbors to be considered (denoted in this thesis as k) in the case of the $local - best$ approach.
- A method for computing distance between particles (used only for local-best): It can be, e.g., L2 (Euclidean distance) or L1 (Manhattan distance).
- Maximum velocity: It sets an upper limit on the particle's velocity.

6.3.3 Algorithm's pseudocode

Algorithm 4 presents a pseudocode of the PSO algorithm.

6.4 Tabu search

6.4.1 Algorithm's description

Tabu Search is a metaheuristic optimization algorithm that is used to find the global optimum solution for combinatorial optimization problems. It was first proposed by Fred W. Glover in 1986 [111] and formalized in 1989 [112].

The algorithm works by maintaining a list of "tabu" or forbidden solutions, which helps to prevent the algorithm from getting trapped in a local

Algorithm 4 Pseudocode of the Particle Swarm Optimization algorithm

Require: Fitness function F; stopping criterion C;
Initialize the particle positions and velocities x_i and v_i;
Initialize the best individual positions $best_i$;
Initialize the global best position g_{best};
while C is not met **do**
  for each particle i **do**
    Update the velocity v_i using equation 6.1 (or 6.2);
    Update the position x_i using equation 6.3;
    if $F(x_i)$ is better than $F(best_i)$ **then**
      Update the best individual position $best_i \leftarrow x_i$;
    end if
    if $F(x_i)$ is better than $F(g_{best})$ **then**
      Update the global best position $g_{best} \leftarrow x_i$;
    end if
  end for
end while
return the global best position g_{best};

optimum. The idea is to encourage the algorithm to explore new solutions and move toward the global optimum.

Tabu Search operates in iterations, where at each iteration a set of new solutions is generated based on the current solution and the tabu list. The new solutions are then evaluated and the best of them that was not in the tabu list becomes the new current solution and is added to the tabu list. Also, if it is better than the best solution found so far, the best solution is updated too. Even if the new solution is not better than the current solution, it still becomes the new current solution, allowing the algorithm to escape from a local optimum. Tabu lists are usually implemented as finite, but they can also be considered infinite. Also, in addition to the points added to the tabu list, the points from their neighborhood can also be added to the tabu list. If the tabu list is too large, the oldest candidate solution is removed and it is no longer tabu and can be reconsidered.

The key elements of Tabu Search are the method of generating new candidate solutions, the way of updating the tabu list, and the mechanism for accepting new solutions. The performance of Tabu Search depends on these elements and their implementation, as well as the size of the tabu list and the criteria for updating it.

There are many variants of the basic tabu search algorithm that have been developed over the years. Some of the most commonly used variants:

- Short-Term Memory Tabu Search (STM-TS) [243]: In this variant, the tabu list is updated after each iteration, with the length of the tabu list being kept short to allow the algorithm to adapt quickly to changing conditions in the search space.

- Long-Term Memory Tabu Search (LTM-TS) [344]: In this variant, the tabu list is updated less frequently, and the length of the tabu list is longer than in STM-TS. Thanks to that, the algorithm can better exploit the knowledge gained during the search process, enhancing its ability to escape local optima and explore different areas of the solution space. Long-term memory can be also combined with short-term memory (as in STM-TS). This variant is typically used for optimization problems where the search space is large and the optimum solution is likely to be far from the starting point.

- Scatter Tabu Search (STS) [241]: In this variant, Tabu Search is combined with scatter search [223], a population-based optimization algorithm, to create a hybrid optimization algorithm that balances exploration and exploitation. The basic idea behind STS is to maintain multiple solution candidates called "elite solutions" throughout the search process. These elite solutions represent diverse areas of the search space and help in exploring different regions of the problem.

- Reactive Tabu Search (RTS) [19]: In this variant, the length of the tabu list is updated dynamically during the optimization process, allowing the algorithm to adapt to changing conditions in the search space.

- Adaptive Memory Tabu Search (AMTS) [113]: In this variant, the size and content of the tabu list are adapted dynamically during the optimization process, allowing the algorithm to balance exploration and exploitation.

6.4.2 Algorithm's hyperparameters

Based on [111, 112], the key hyperparameters of the Tabu Search algorithm are:

- Tabu list length: determines the number of previously visited solutions that are marked as "tabu".

- Tabu tenure: the number of iterations that a move stays in the Tabu List.

- The method for generating new candidate solutions: determines the set of candidate solutions that are generated from the current solution.
- Aspiration criteria: it is an optional hyperparameter which allows specific tabu moves to be accepted if they lead to improvements that exceed a certain threshold.
- Maximum number of iterations: determines the maximum number of iterations that the algorithm should run before terminating.

6.4.3 Algorithm's pseudocode

Algorithm 5 presents a pseudocode of the Tabu Search algorithm.

Algorithm 5 Pseudocode of the Tabu Search algorithm

Require: Fitness function F; stopping criterion C; method for generating a set of candidate solutions G; Desired maximum tabu list length L;

Initialize the current solution x;
Initialize the best solution *best* as x;
Initialize the tabu list T as a list containing x;
while C is not met **do**
    Generate a set of candidate solutions N from x using G;
    Evaluate the fitness function F for each candidate solution in N;
    Identify the best candidate solution S that is not in the tabu list T;
    if S is better than the best solution *best* **then**
        $best \leftarrow S$;
    end if
    Update the tabu list T by adding S and removing the oldest element if the maximum length was achieved;
end while
return the best solution *best*;

6.5 Simulated annealing

6.5.1 Algorithm's description

Simulated Annealing (SA) is a probabilistic optimization technique inspired by the annealing process in metallurgy, where a metal is slowly cooled to reduce its defects and increase its stability. Similarly, in SA the algorithm slowly reduces the temperature to allow the solution to converge to an optimum [185].

SA is commonly used when the solution space is large or when the function being optimized has many local minima. Unlike other optimization techniques that can get stuck in local minima, SA has a chance of jumping out of a local minimum and finding the global minimum.

The algorithm works by randomly perturbing the current solution and evaluating the cost of the new solution. If the cost of the new solution is better than the current solution, the new solution becomes the current solution. If the cost of the new solution is worse, it is still accepted with a certain probability that depends on the temperature. As the temperature decreases, the probability of accepting worse solutions also decreases.

The temperature schedule used in SA is crucial to the success of the optimization. If the temperature decreases too quickly, the algorithm may converge to a local minimum before reaching the optimum. If the temperature decreases too slowly, the algorithm may not converge at all.

SA is widely used in various fields, such as machine learning, computer graphics, logistics, and engineering design.

6.5.2 Algorithm's hyperparameters

The key hyperparameters of SA are [185, 319]:

- Initial temperature: It determines the initial level of exploration. A high initial temperature results in a higher acceptance rate of worse solutions, allowing the algorithm to explore more of the solution space. A low initial temperature results in a lower acceptance rate of worse solutions, leading to faster convergence.

- Stopping criterion: It determines when the algorithm should stop. Common stopping criteria include reaching a certain temperature, reaching a maximum number of iterations, or reaching a desired level of accuracy.

- Final temperature (if used in a stopping criterion): It determines the final level of exploration, i.e., how much the optimization process will converge to a minimum solution. A lower final temperature leads to faster convergence. However, setting the final temperature too low may result in premature convergence to a suboptimal solution.

- Cooling schedule: It determines how the temperature decreases over time. A fast cooling schedule results in faster convergence but can cause the algorithm to get stuck in a local minimum. A slow cooling schedule results in a higher probability of finding the global minimum but can be computationally expensive. The cooling schedule can be, e.g., a geometric cooling ($T_{k+1} = \alpha * T_k$, where α is a cooling rate and T_k, T_{k+1} are temperatures in consecutive steps), or a linear cooling ($T_{k+1} = T_k - \delta T$, where δT is a positive constant value).

- Acceptance function: It determines the probability of accepting a worse solution. The most common acceptance function used in SA is the Boltzmann acceptance function [185], which takes into account both the current temperature (T) and the difference in cost between the current solution and the new solution (ΔE): $P(\text{accept}) = \exp\left(-\frac{\Delta E}{T}\right)$.

- Neighborhood function: The neighborhood function defines how the solution space is sampled.

- Number of different starting points: different starting points may lead to totally different solutions, so it is good to run simulated annealing several times starting from different points.

6.5.3 Algorithm's pseudocode

Algorithm 6 presents a pseudocode of the Simulated Annealing:

Algorithm 6 Pseudocode of the Simulated Annealing algorithm

Require: Initial solution x; initial temperature T; cooling schedule S; stopping criterion C; number of iterations N; fitness function F; function G returning a neighbourhood $G(x)$ of a given solution x;

```
1: best ← x;
2: k ← 0;
3: while C is not met and k < N do
4:    Select a new solution x' from (G(x));
5:    Calculate ΔE = F(x') − F(x);
6:    if ΔE < 0 then
7:       x ← x';
8:       if F(x) < F(best) then
9:          best ← x;
10:      end if
11:   else
12:      p ← exp(−ΔE/T);
13:      Randomly generate r ∈ [0, 1];
14:      if r < p then
15:         x ← x';
16:      end if
17:   end if
18:   update T according to the cooling schedule S;
19:   k ← k + 1;
20: end while
21: return best;
```

6.6 Covariance Matrix Adaptation Evolution Strategy

6.6.1 Algorithm's description

Covariance Matrix Adaptation Evolution Strategy (CMA-ES) is a population-based optimization algorithm that is commonly used for solving high-dimensional, non-linear, and non-convex optimization problems [141]. It is a type of evolutionary algorithm that is based on the principles of evolution and natural selection, similar to genetic algorithms (Section 6.2). The algorithm is well suited for problems where the objective function is expensive to evaluate, as it requires relatively few function evaluations to find a good solution.

CMA-ES works by generating a population of candidate solutions and

iteratively improving them based on their fitness. The algorithm maintains a covariance matrix that describes the distribution of the solutions in the population and uses this matrix to generate new solutions in each iteration. The covariance matrix is updated in each iteration based on the results of the objective function evaluations, allowing the algorithm to adapt to the underlying structure of the problem and effectively search for a global optimum.

CMA-ES has several key features that make it well-suited for many optimization problems. One of the key advantages of the algorithm is that it can effectively handle problems in which the parameter space has a high degree of correlation between the parameters. The algorithm can also handle problems with noisy objective functions, as it is robust to the presence of measurement noise. Additionally, CMA-ES is highly scalable, making it well-suited for large-scale optimization problems.

A more detailed description of CMA-ES can be found in, e.g., [141, 140].

6.6.2 Algorithm's hyperparameters

The key hyperparameters of the CMA-ES algorithm are:

- Population Size: It determines the number of candidate solutions that are generated in each iteration of the optimization. Larger population size can lead to better exploration of the search space but also requires more computational resources.
- Stopping Criterion: It specifies the conditions under which the optimization process will terminate. Common termination criteria include reaching a certain tolerance, achieving a specific value for the cost function, or reaching a maximum number of iterations.
- Initial Covariance Matrix: It is a user-defined matrix that determines the initial distribution of candidate solutions in the population. The choice of the initial covariance matrix can have a significant impact on the performance of the optimization.
- Initial Mean Vector: Represents the starting point of the search in the solution space.
- Offspring Size: The number of selected points from the population that will be used to update the mean and covariance matrix.
- Step Size: It determines the scale of the mutations that are applied to the candidate solutions in each iteration. A larger step size can lead

to faster convergence but may also result in premature convergence to suboptimal solutions.

- c_1: Learning rate for the rank-one update of the covariance matrix update - It is used to control the step size of the rank-one update [204] of the covariance matrix and determines how much the rank-one update contributes to the overall covariance matrix update.

- c_μ: Learning rate for the $rank-\mu$ update [191] of the covariance matrix update.

6.6.3 Algorithm's pseudocode

Algorithm 7 presents a pseudocode of the CMA-ES algorithm.

Algorithm 7 Pseudocode of the CMA-ES algorithm

Require: Fitness function F; stopping criterion C; population size λ; selection size μ; initial mean vector m; initial covariance matrix CM; initial step size σ;

while C is not met **do**
  for $i = 1, 2, \ldots, \lambda$ **do**
    Sample $z_i \sim \mathcal{N}(0, CM)$;
    Compute $x_i = m + \sigma z_i$;
  end for
  for $i = 1, 2, \ldots, \lambda$ **do**
    Evaluate fitness $F(x_i)$;
  end for
  Sort offspring by fitness and select top μ points $x_{1:\mu}$;
  Update mean $m = \frac{1}{\mu} \sum_{i=1}^{\mu} x_i$;
  Update covariance matrix C;
  Update step size σ;
end while

6.7 Memetic Algorithms

6.7.1 Algorithm's description

Memetic Algorithms are a class of optimization algorithms that combine genetic algorithms and local search methods [67]. They are called "memetic" because the algorithms work by maintaining a population of solutions, similar to Genetic Algorithms, while also allowing individual solutions to be refined through local search methods, similar to a cultural evolution process where new knowledge and techniques are passed down through generations.

A Memetic Algorithm typically consists of the following steps:

- Initialization: A population of possible solutions is generated, typically using random sampling or a heuristic method.
- Fitness Evaluation: The "fitness" of each solution in the population is evaluated.
- Selection: A subset of the fittest solutions is selected for further refinement.
- Local Search: The selected solutions are refined using a local search method, such as gradient descent or simulated annealing.
- Mutation: New solutions are generated by mutating the refined solutions or by combining solutions from the population through a crossover.
- Repeat: The process of fitness evaluation, selection, local search, and mutation is repeated until a stopping criterion is met, such as a maximum number of iterations or a desired level of performance.

The key advantage of Memetic Algorithms over traditional Genetic Algorithms is the ability to perform local searches on individual solutions, allowing for more effective exploration and exploitation of the search space. This can result in faster convergence to the optimal solution and improved solution quality.

6.7.2 Algorithm's hyperparameters

The key hyperparameters of the Memetic Algorithm are [67, 248]:

- Population size: The number of potential solutions in the population.

- Selection mechanism: The method used to choose individuals for reproduction, such as tournament selection, roulette wheel selection, etc.
- Crossover operator: The method used to combine parent solutions to generate offspring, such as one-point crossover, uniform crossover, etc.
- Mutation operator: The method used to modify offspring solutions (similar to genetic algorithms), such as uniform mutation, or swap mutation.
- Mutation probability: The probability of applying the mutation operator to modify an offspring solution.
- Local search heuristic: The specific local search method used to improve individual solutions, such as hill climbing, simulated annealing, or tabu search.
- Elitism: The number of top-performing solutions that are preserved from one generation to the next. This hyperparameter controls the trade-off between exploration and exploitation in memetic algorithms. By preserving the best solutions, elitism ensures that the algorithm does not lose sight of promising regions of the solution space, while still allowing for the exploration and improvement of candidate solutions.
- Stopping criterion: The conditions determine when the algorithm should stop.

6.7.3 Algorithm's pseudocode

Algorithm 8 presents a pseudocode of the Memetic Algorithm.

6.8 Hill Climbing

6.8.1 Algorithm's description

The Hill Climbing algorithm belongs to the family of local search algorithms. It starts with an arbitrary solution to a problem and then attempts to find a better solution by making an incremental change to the solution. The algorithm employs a greedy approach, which means that the movement through the space of solutions always occurs in the sense of improving the value of the objective function [290].

Algorithm 8 Pseudocode of the Memetic Algorithm

Require: Fitness function F; stopping criterion C; selection, crossover, mutation operators, local search heuristic;

1: Initialize the population of candidate solutions P;
2: Evaluate the fitness $F(x)$ of each candidate solution $x \in P$;
3: **while** C is not met **do**
4: Select a subset of the current population using a selection method for further reproduction (optionally, preserve some of the best-performing chromosomes);
5: Apply crossover and mutation operators to generate a new population;
6: Perform local search on representatives of the new population to improve their quality and obtain a new population R;
7: Evaluate the fitness $F(x)$ of each new candidate solution x from the current population R;
8: **end while**
9: **return** the best solution found;

6.8.2 Algorithm's hyperparameters

The performance of the Hill Climbing algorithm can be affected by the choice of the starting point of the algorithm. That is why it is often recommended to repeat the algorithm multiple times with different starting points. One can also impose a limit on the number of iterations in order to find a suboptimal solution within a certain time limit.

6.8.3 Algorithm's pseudocode

Algorithm 9 presents a pseudocode of the Hill Climbing algorithm.

6.9 Quantum metaheuristics

Besides classical metaheuristics, it is also good to mention the emerging field of quantum metaheuristics that combines principles from quantum computing with metaheuristic algorithms. In order to process information, quantum computing uses qubits which can exist in multiple states simultaneously, and computations can be performed thanks to taking advantage of quantum mechanical phenomena such as superposition or quantum entanglement. Even though contemporary quantum computers are still relatively small and noisy and their current and potential advantage over classical computers is still questionable, it is expected that they could solve some classes of problems

Algorithm 9 Pseudocode of the Hill Climbing algorithm

```
Require: Stopping criterion C; fitness function F that should be optimized;
  initial candidate solution x;
  while C is not met do
    x' ← neighbor of x with the best value of F;
    if F(x') is better than F(x) then
      x ← x';
    else
      return x;
    end if
  end while
  return x;
```

faster or better than classical computers. Among examples of such applications, one can distinguish factorization, simulating chemical reactions and quantum mechanical phenomena, sampling from certain distributions, and solving combinatorial optimization problems. The last class of problems is the one in which quantum metaheuristics are being especially explored. There are already quantum (or hybrid: quantum-classical) variants of many classical algorithms, like Quantum Particle Swarm Optimization [213], Quantum Ant Colony Optimization [107], Quantum Genetic Algorithms [360].

There is also the Quantum Annealing approach which can be also considered a metaheuristic [173]. The core idea is to map an optimization problem onto the physical architecture, which consists of physical qubits arranged in a lattice and interacting through couplers. The optimization problem must be encoded in the form of an Ising Hamiltonian, which describes the energy landscape of the optimization problem, and the goal is to find the ground state of this Hamiltonian, corresponding to the optimal solution. The Ising Hamiltonian corresponds to the mathematical formulation called QUBO (Quadratic Unconstrained Binary Optimization), which expresses the objective function of the optimization problem as a quadratic polynomial in the binary variables. When an optimization problem can be represented as a QUBO, it is possible to map it onto the physical architecture of a quantum annealing machine, where the energy of the quantum system composed of qubits corresponds to an Ising Hamiltonian formulation.

The Quantum Annealing process starts with the qubits initialized in a simple quantum state (e.g., superposition of all possible states), which can be considered an initial solution. Then, the Hamiltonian of the quantum system is gradually changed (by modifying the strength of the magnetic field) from an initial, simple Hamiltonian, to the Ising Hamiltonian encoding

the optimization problem. This gradual change is controlled by a parameter called the annealing schedule, which starts at a large value favoring the initial Hamiltonian and gradually decreases to a smaller value favoring the Ising Hamiltonian.

During the annealing process, the algorithm leverages quantum tunneling, superposition, and entanglement to explore the energy landscape of the optimization problem. Quantum tunneling enables the system to escape local minima and explore different regions of the energy landscape. Through superposition, Quantum Annealing can explore multiple potential solutions simultaneously. Entanglement further enhances this exploration by establishing correlations between qubits, enabling them to collectively contribute to the search process.

Upon completion of the annealing process, the solution can be read out by measuring the qubits in their final state. In theory, the read solution should correspond to the ground state (optimal solution) of the Ising Hamiltonian and the solution of the optimization problem. However, in practice, achieving the ground state is challenging due to specific requirements on the annealing process that are difficult to fulfill. For example, the annealing process needs to be slow enough to allow the system to adiabatically evolve and stay close to the ground state. Nevertheless, there are time constraints and the annealing process is often faster than ideal, which can result in the system getting trapped in local minima or excited states. Also, quantum annealing machines are highly sensitive to external noise and imperfections, which can introduce errors in the computation and affect the coherence of the qubits. Decoherence can cause the system to deviate from the ground state, leading to suboptimal solutions.

As a consequence, Quantum Annealing often results in suboptimal solutions, which is why it is considered a metaheuristic rather than an exact algorithm. Nonetheless, Quantum Annealing can still provide valuable insights and solutions for various optimization problems, especially when combined with other optimization methods.

Even though there are still open discussions regarding the capabilities of the existing quantum annealers (like the D-Wave's machine [196]), it is good to emphasize that there are already theoretical works and experiments showing how it could be applied to solve combinatorial optimization problems related to complex processes, e.g., in transportation. For example, in 2017, researchers from D-Wave and Volkswagen showed how to apply quantum annealing to optimize the route assignment for a fleet of taxis and reduce traffic jams, as a consequence [250]. There are also many research works on applying Quantum Annealing to solve the Travelling Salesman Problem, the Vehicle Routing Problem and its variants, and a team of researchers led by the au-

thor of this thesis also contributed to that field by proposing new quantum and hybrid (quantum-classical) algorithms based on Quantum Annealing to solve those optimization problems [32]. There is also a publication from 2021 showing how Quantum Annealing can be used to optimize traffic signal settings [160], which is the main study problem in this thesis. However, this technique is applied to simplified models of road networks, traffic signals, and traffic flow, so it cannot be directly applied in practice, but it shows that Quantum Annealing can be already considered an interesting metaheuristic that is worth further exploration.

6.10 Bayesian Optimization

6.10.1 Algorithm's description

Bayesian Optimization (BO) is a probabilistic model-based method for optimizing a function that is expensive or difficult to evaluate [101]. It is suitable for the global optimization of black-box functions that do not assume any functional forms. It is also effective for optimizing functions with complex structures, such as functions with multiple local optima. BO is widely used in hyperparameter tuning and global optimization problems [239]. Additionally, BO is computationally efficient, and it can quickly converge to the best solutions, even when dealing with high-dimensional parameter spaces. One important note is that Bayesian optimization is not considered a metaheuristic, but it was used for comparison in some experiments presented in this thesis, so it is also explained in this section.

The key idea behind BO is to build a probabilistic model (PM) of the objective function (OF), which can be used to make informed decisions about where to evaluate the OF next. This allows the algorithm to efficiently explore the search space and avoid evaluating the function in regions where the function value is likely to be poor.

The process of BO can be divided into 2 main steps: model building and acquisition function optimization.

In the model-building step, the PM of the OF is constructed based on the available OF evaluations. PM takes into account the uncertainty in the OF value and can be updated as more OF evaluations are obtained.

In the acquisition function optimization step, an acquisition function (AF) is used to balance the trade-off between exploration and exploitation. AF is a measure of the expected improvement in the OF value at a given point, and it is optimized to determine the next point at which to evaluate the OF. In BO, AF is used in combination with a PM to guide the search for the next

set of sample points to evaluate. The PM is updated after each evaluation of the sample points, and the next set of points to evaluate is chosen by maximizing AF based on the updated PM. This process is repeated until a stopping criterion is met.

BO can be computationally expensive, and the choice of PM and AF can have a significant impact on the performance of the algorithm.

6.10.2 Algorithm's hyperparameters

The key hyperparameters of the BO algorithm are [101, 98]:

- PM: It influences the accuracy of the predictions and the speed of convergence. Gaussian Process (GP) is a popular choice, but other models, such as random forests, may be used as well.
- AF: it determines the trade-off between exploration and exploitation in the search for the best solutions. Common choices include Expected Improvement (EI), Probability of Improvement (PI), and Upper Confidence Bound (UCB).
- Initial design: The number of initial points and their selection method (e.g., random sampling, Latin Hypercube Sampling, or Sobol sequences). These initial points are used to build the initial PM.
- Number of evaluations: It determines the total number of evaluations of the sample points that will be performed during the BO process, and the computational cost of the optimization process, as a consequence.
- Stopping criterion: It determines when the optimization process should stop, either because a satisfactory set of candidate solutions has been found, or because a maximum number of evaluations has been reached.
- Batch size: It determines the number of sample points that are evaluated at each iteration of the optimization process. Larger batch size can reduce the computational cost but may also lead to reduced exploration.

6.10.2.1 Model Building

Popular models used in Bayesian Optimization include Gaussian Process, Random Forest, Extra Trees, and Gradient Boosted Trees [101, 98, 239].

The Gaussian Process (GP) model takes the previous observations of the OF as input and predicts the mean and variance of the OF at any new input

point. Then, this information can be used by AF to decide where to sample the OF next.

The GP model provides a probabilistic estimate of the OF, which allows the AF to balance the trade-off between exploration and exploitation. By exploring regions where the OF value is uncertain, the algorithm can efficiently search the search space and identify the global optimum.

One of the strengths of GP in BO is its ability to handle noisy and heterogeneous data, which is common in many optimization problems. GP models the noise as an additional source of uncertainty, which allows the algorithm to take into account the noise level when making decisions about where to evaluate the OF next. Additionally, GP provides a natural way to incorporate prior knowledge about the OF into the model. This can be useful in cases where the OF has known constraints or other structural properties.

Random Forest is a widely-used machine learning algorithm that is used for both regression and classification tasks. It is an ensemble learning method that creates a set of decision trees and combines their predictions to make a final prediction. The key idea behind Random Forest is to create multiple trees, each of which is trained on a random subset of the data and a random subset of the features, and then average the predictions of all trees to get a final prediction.

Extra Trees (Extremely Randomized Trees) is an extension of the Random Forest approach. The primary difference between the two is that Extra Trees adds more randomization to the process of building decision trees, resulting in a more diverse and robust model. Extra Trees approach works by creating an ensemble of decision trees, each of which is grown using a random subset of the features and a random subset of the training data. The final prediction is made by averaging the predictions of all trees in the ensemble. The randomized nature of Extra Trees helps to mitigate overfitting, a common problem in traditional decision trees.

Gradient Boosted Trees (GBTs) is also an ensemble learning method that creates a set of decision trees and combines their predictions to make a final prediction. The key idea behind GBTs is to iteratively add decision trees to the model, where each new tree is trained to correct the errors made by the previous trees. GBTs work by fitting simple decision trees to the negative gradient of the loss function (which represents the discrepancy between the predicted values and the true values of the OF) with respect to the predictions. This means that each new tree focuses on the errors made by the previous trees, and tries to correct them. The final prediction is made by adding up the predictions of all trees in the model. One of the main advantages of GBTs is that they can handle complex non-linear relationships between the inputs and outputs of the OF. They also have the ability to handle missing

data and outliers and can be used with a variety of loss functions.

When using GBTs (or Extra Trees or Random Forest) in BO, the model is trained using different sample points, and the performance of the PM is evaluated using a performance metric, such as mean squared error. The PM is then updated based on the results of these evaluations, and the process is repeated until the best points are found.

6.10.2.2 Acquisition Function

Common AFs include Probability of Improvement (PoI), Expected Improvement (EI), and Upper Confidence Bound (UCB) [101, 98, 239].

PoI is defined as the probability that the next set of sample points will contain a better point than the current best solution. Using the trained PM, the mean and standard deviation of the predicted OF values are computed for each sample point in the batch, and then the PoI value for each point is calculated by considering the difference between the current best solution and the predicted mean, normalized by the standard deviation. The next set of sample points to evaluate is chosen by maximizing the PoI.

EI is defined as the expected improvement over the current best solution. Its idea is similar to PoI, but EI considers both the probability and the magnitude of improvement. It quantifies the expected improvement over the current best solution, taking into account the uncertainty of the PM. EI calculates the weighted average improvement by incorporating the predicted mean and standard deviation of the PM. It assigns higher values to points that not only have a higher probability of improvement but also exhibit larger potential improvements.

The key idea behind using EI as the AF is to favor candidate solutions that have the potential to make a significant improvement over the current best solution. This helps to avoid getting stuck in a local minimum and to ensure that the search for the best solution is comprehensive and efficient.

Compared to PoI, EI can lead to more exploitation and faster convergence to the best solution. However, it may also lead to over-exploitation and a lack of exploration, especially in high-dimensional parameter spaces.

The UCB acquisition function is based on the idea of balancing exploration and exploitation by adding a term that represents the uncertainty of the PM's predictions. The UCB for a given point is calculated as the mean of the PM's predictions plus a term that depends on the uncertainty of the prediction (e.g., standard deviation). It can be written as $UCB = Mean + (\beta \cdot Uncertainty)$, where β is a parameter that controls the trade-off between exploration and exploitation. The higher the value of β, the more emphasis is placed on exploration. The term representing the uncertainty

of the PM's predictions is designed to increase as the uncertainty of the prediction increases. This encourages exploration in areas where the model is uncertain, helping to avoid over-exploitation and ensure a comprehensive search for the best candidate solutions. Similar to PoI and EI, the next set of points to evaluate is selected by maximizing the UCB.

6.10.3 Algorithm's pseudocode

Algorithm 10 presents a pseudocode of the BO algorithm.

Algorithm 10 Pseudocode of the Bayesian Optimization algorithm

Require: Objective function OF; stopping criterion C; probabilistic model PM; acquisition function AF;
1: Initialize PM with an initial design of points;
2: Evaluate the performance of the points using OF;
3: Update PM with the new information;
4: **while** C is not met **do**
5: Select the next point using the acquisition function AF;
6: Evaluate the performance of the selected point using OF;
7: Update PM with the new information;
8: **end while**
9: **return** the set of points with the best performance;

6.11 Gradient Descent

6.11.1 Algorithm's description

Gradient Descent is an optimization method in which the search directions are defined by the gradient (or approximate gradient) of the differentiable objective function at the current point. The idea is to take repeated steps in the opposite direction of the gradient (or approximate gradient) of the objective function at the current point (or the average of gradients at many points in a batch) since this is the direction of the steepest descent. There might be different variants of the Gradient Descent algorithm, e.g.: Stochastic Gradient Descent (SGD) [35], Momentum [289], AdaGrad [289], Nesterov [249], RMSProp [289], Adam [183]. The Gradient Descent algorithm relies on an explicit formulation of the optimization function and its gradients, so it is generally not considered a metaheuristic, but since it is one of the most popular optimization algorithms, it was decided to investigate it in several experiments for comparison.

6.11.2 Algorithm's hyperparameters

Hyperparameters of the Gradient Descent algorithm depend on its variant. Some of the most common hyperparameters:

- Number of iterations: number of steps of the gradient descent algorithm.
- Learning rate: It controls how much the model's parameters are updated during each iteration and determines how much the parameters should be adjusted at each iteration of the optimization process [289]. A smaller learning rate may lead to slower convergence but higher accuracy, while a larger learning rate may result in faster convergence but lower accuracy or even divergence.
- Momentum: In some gradient descent algorithms (e.g., Adam) it controls the extent to which the previous parameter update influences the current update.
- Batch size: In some variants (like SGD), the batch size determines the number of sample points used to estimate the gradient at each iteration. A larger batch size may lead to more accurate gradient estimates and smoother convergence, while a smaller batch size may result in faster training and better exploration of the optimization landscape.
- ϵ: Used for numerical stability and to prevent division by zero for some algorithms like RMSProp or Adam [289].
- γ: The decay rate of the average of the squared gradient, used in some algorithms like RMSProp [289].
- β_1 and β_2: Initial decay rates used in Adam ([183]) for estimating the first and second moments of the gradient.

It is also important to note that outcome of the Gradient Descent algorithm may strongly depend on the selected initial point, so very often this algorithm is repeated many times starting from different initial points.

6.11.3 Algorithm's pseudocode

Algorithm 11 presents a pseudocode of the Gradient Descent algorithm.

Algorithm 11 Pseudocode of the Gradient Descent algorithm

Require: Objective function OF; learning rate α; number of iterations T; initial candidate solution x;

1: $i \leftarrow 0$;
2: **while** $i < T$ **do**
3: Compute the gradient of the cost function $\nabla_x OF(x)$;
4: Update the candidate solution: $x \leftarrow x - \alpha \nabla_x OF(x)$;
5: $i \leftarrow i + 1$;
6: **end while**
7: **return** x;

6.12 Limitations and drawbacks of metaheuristics

Although metaheuristics are a powerful tool for solving complex optimization problems, they also have some limitations and drawbacks that need to be taken into account when using them [61]:

- No guarantee of finding the optimal solution: Metaheuristics are designed to find a good solution, but there is no guarantee that they will find the optimal solution. The quality of the found solution depends on the specific parameters, the problem complexity, and the characteristics of the solution space. However, as explained in Section 6.1, this limitation is less severe in the case of dynamically evolving complex processes, for which only approximate mathematical models can be built, so sometimes it might be even difficult to correctly define what optimality actually means.

- High computational cost: Metaheuristics typically require many iterations to explore the solution space, which can be computationally expensive. Therefore, metaheuristics are generally slow to converge to an optimal solution. An additional difficulty arises when the time required to evolve the quality of the solutions is significant, and this can be the case for many complex processes, as discussed in Chapter 4.

- Dependence on parameters: Metaheuristics usually have many hyperparameters that need to be tuned, so finding good values can add computational overhead.

- Dependence on initial settings: The performance of metaheuristics may also depend on the selection of initial settings, like the initial popula-

tion in Genetic Algorithms. Therefore, in some cases, it is necessary to ensure the correct selection of those initial settings, e.g., through randomness or taking into account the characteristics of the solution space. Sometimes, it might be necessary to run multiple experiments starting from different initial settings, which can add computational overhead.

Chapter 7

Methodology

Let us consider a complex system CS and its evolution in time as a complex process CP (cf. Definitions 3.1 and 3.2). In general, the process CP might exist only for a specific time period T. In this context, T can be considered as the *time domain* of the complex process CP. For the purpose of this thesis, relativistic effects are neglected so time can be considered as a classical, scalar quantity. Therefore, it can be assumed that T is a connected subset of $\mathbf{R}$ containing 0, which serves as a reference point in time for a given observer.

Let $CP(t)$ be the state of CP at time $t \in T$. Here we abstract from a formal definition of what exactly the state of CP is and how to define it, because for different complex processes, it may be something completely different. For example, in the case of the evolution of abstract complex processes like a cellular automaton, it may be just a set of states of all the cells that make up that automaton. In the case of physical complex processes, it can be more complicated because it should be considered more as a state of physical reality. For road traffic in cities, it can be the current locations, speeds, and other important characteristics of vehicles, but also the "states" of drivers (including their minds making decisions about how to drive), the states of traffic signals (including the states of their physical components), etc. An important note is that the state of a complex process is also not the same as the result of observing it, since the result of observation may be different for different observers/detectors - for some observers some features of $CP(t)$ may not be measurable, for other observers the features may be observable but with limited accuracy. Nevertheless, building a mathematical model of a complex process requires modeling its "states" at different points in time.

Let us call M a model of the complex process CP defined on a time domain $T' \subset \mathbf{R}$ if for each $t \in T' \cap T$, M can provide a mathematical representation of the state $CP(t)$. Note that T and T' do not have to be the same sets: there might be a reasonable model that is defined only for some points

in time (e.g., it could be a discrete model of a continuous process). On the other hand, it is possible that the model might be able to provide a representation of the process CP even outside of its time domain T. Usually, in order for the model to be useful, it should be able to provide an accurate representation of CP for many points in time. However, different processes and applications may require different levels of accuracy for the model to be considered useful, and not always the most accurate models are the most useful ones (and usually some other factors, like the time of the model's evaluation, are also important to consider the model as useful). Therefore, the requirement regarding the model's accuracy is rather intuitive and not formal. Also, calculating the accuracy of a model requires comparing the model outcomes with the results of observing the complex process CP, and these results may depend on the observer and the quality of detectors. Therefore, the concept of accuracy is formally not included in the definition of the model (Definition 7.1), but intuitively, useful models should be of satisfactory quality.

The model can be deterministic or stochastic, so, in general, it can be assumed that for a timestep $t \in T'$, M can provide a distribution over possible states of the model. If the model is deterministic, the distribution would be just a Dirac delta [78].

In the case of optimizing the complex processes CP, it is necessary to define the objective function that should be optimized as well as the control parameters that can be used to steer the evolution of CP. The optimization of a complex process can be performed directly on that process, controlling its behavior in real-time, but usually, in order to design proper optimization strategies and algorithms, it is necessary to investigate the optimization of a realistic, accurate model of a complex process, and this is the approach and assumption for the rest of this chapter.

Therefore, control parameters can be considered as variables of the model M (being a mathematical representation of the complex process CP) that influence its evolution. Examples of control parameters are settings of traffic signals (in the case of optimizing road traffic), protocols (doses and times of their administration) of radiotherapy, etc.

A model of a complex process can be therefore defined as follows:

Definition 7.1 *A model M of a complex process CP is a tuple $< S, C, V, T, F >$, where:*

- *S is a set of states of the model being mathematical representations of possible states of the process CP,*
- *C is a set of control parameters (in practical applications, C is always a finite set),*

- *V is a set of possible assignments of values of control parameters C (i.e., for $v \in V$ and $c \in C$, $v(c)$ can be considered as a value of the control parameter c for assignment v),*
- *T is a time domain,*
- *F is a function that for a given $v \in V$ and a timestep $t \in T$ returns a distribution over states S, so $F : V \times T \to (\mathcal{P}(S))$, where $\mathcal{P}(S)$ is the set of probability measures over S.*

The objective function that has to be optimized, denoted as OF, should be defined on the set of possible assignments of values of the control parameters (V) and provide a value from a set Y in which an order is defined (hence, $OF : V \to Y$). For simplicity, it will be assumed that this is a linear order, but partial orders could be also considered in some cases. In the case of complex processes which are usually computationally irreducible [357], calculating the exact value of the objective function usually requires evaluating the model M over many time steps (cf. Section 3.2.2).

Let us now give some examples:

- For vehicular traffic in cities in which the control parameters are settings of traffic signals, the model M could be based on any of the traffic simulation models producing a representation of traffic for each possible traffic state (cf. Section 4.2.1 for a review of traffic models) and a considered time. For discrete traffic models, T can be represented as $\mathbb{N}$, for continuous traffic models, T can be represented as $\mathbb{R}$. The set S could be a set of possible representations of traffic states that could be generated by the given traffic model. For example, in the case of microscopic traffic simulation models, S could be a set of all possible positions and speeds of all vehicles that can take part in the simulation, as well as potential values of control parameters, at any possible timestep from the set T. The set of control parameters C could be composed of modifiable settings of traffic signals (e.g., offsets) in a given simulation area (e.g., a city district), so the assignments of their values (elements from V) can be represented as vectors with integer values representing, e.g., values of offset and durations of signal phases at all considered intersections. The function F can be considered as a function that for the given assignment of traffic signal settings and a given time step returns the distribution over possible states (e.g., positions and speeds of all vehicles). The values of F can be computed, e.g., by evaluating the selected traffic simulation model. The objective function OF could take as an input an assignment of values of signal settings

(elements from the set V) and return some traffic characteristics, e.g., the total time of waiting at red signals (or the total delay) of all vehicles in a given area at time $T_{MAX} \in T$, aggregated from the reference point in time, T_0. The considered traffic characteristics (values of OF) can be calculated by the function F based on the traffic model and its control parameters.

- For cancer growth and the model described in Section 4.3, S is the set of possible states of the cellular automaton modeling cancer. In this model, time is discrete, so T is equivalent to the steps in which cancer can be simulated (the time domain can be represented as $\mathbb{N}$). The control parameters C could be variables indicating doses at all possible time steps of the radiation administration. Therefore, values of V can be represented as functions that return doses for each time step (of potential dose administration). Elements of V can also be represented as sets of pairs (d, t), where d are administered doses and t are times of their application. F is a function that for a given assignment of control parameters (interpreted as a radiotherapy protocol) and time step returns the distribution over possible cancer states, calculated based on the considered simulation model, which is a stochastic cellular automaton. The objective function OF could take as an input an assignment of control parameters (interpreted as a radiotherapy protocol) and return the estimated expected value of the number of cancer cells after 10 days of the treatment process, corresponding to 144000 simulation steps (each simulation step corresponds to 6 seconds, as described in [15] and Section 4.3). This estimation could be done by running a sufficient number of simulations and averaging the outcomes.

The essential characteristic of an objective function is its computational efficiency, meaning that it should be possible to calculate the function's value within a reasonable time frame. For example, in the case of stochastic models, like stochastic cellular automata that are a base of the model for cancer growth that is used in this thesis (Section 4.3), it might be intractable to accurately calculate the expected value. In such cases, the objective function should rather estimate the exact value. Also, even in the case of deterministic models, it may turn out that their evaluations using simulations take a very long time due to computational irreducibility. In such cases, it might be also better to choose as the objective function an approximation (e.g., a surrogate model) of the model's output that can be evaluated fast.

Nevertheless, even if the objective function is computationally efficient, discovering the optimal value may still prove to be time-consuming or computationally challenging. For example, in the case of road traffic, even for a

very simple mathematical model based on cellular automata, in which traffic signal settings are control parameters, the problem of finding the optimal setting is proven to be NP-hard, highlighting the complexity of the task [54].

For some NP-hard problems, it is very important to find the exact, optimal solutions, even if finding such solutions is computationally demanding. For example, one can think about the Traveling Salesman Problem (TSP) [198] in which the goal is to find the route visiting all specified locations in the order that minimizes the cost. Similarly, its extension, the Vehicle Routing Problem (VRP) [73], involves multiple vehicles that must originate and terminate at a depot while visiting all locations at the lowest possible cost. In the case of such problems, on a grand scale, with thousands of vehicles operating for a long time and visiting many locations, the ability to find the optimal, exact solutions becomes critical, because it may save significant costs for logistics companies.

However, the definitions of TSP and VRP assume simplifications of real-world conditions, e.g., they do not consider varying travel times or dynamic changes in logistics requirements, they assume having perfect knowledge about road conditions, etc. That is why the solutions that are optimal in simplified theoretical models do not have to lead to the best possible solutions in the real world. As a consequence, in practice, it is usually sufficient to find near-optimal solutions that might be slightly worse than the optimal solutions in the theoretical model but can be found relatively fast. Moreover, it may not be reasonable to search at all costs for the exact solutions to problems defined with simplified models, because the near-optimal solutions (which can be found much faster) may even turn out to be better in the real world.

In such cases, when the goal is not to find the optimal solution (which might be computationally intractable), but a near-optimal solution, the effective and universal tools are *metaheuristics*, explained in detail in Chapter 6. They are algorithms that guide the process of searching for near-optimal solutions (usually in very large spaces) and are not problem-specific which means that they can be applied to all sorts of optimization problems (however, for some optimization problems some better techniques may exist).

One of the properties of metaheuristics is that as they guide the process of searching for good solutions, they need to have the ability to efficiently assess the values of the objective function for different points in the search space. The ideal situation is when the values of the objective function can be calculated explicitly relatively fast. However, in the case of stochastic models or even deterministic models, it can be intractable to calculate the exact outcomes of the model for given control parameters, so it can be intractable to calculate the values of the objective function OF using that model. A

natural question is whether it is possible to calculate, or at least estimate, the outcomes of OF faster, without evaluating the simulation model.

In such cases, one can think about using another model, the so-called *surrogate model* [6], as a substitution for the original model in the process of calculating the values of the objective function. Therefore, instead of calculating the exact values of OF, another function can be used that employs a surrogate model to approximate the exact values of OF. Let's denote such an approximator based on a surrogate model as SM.

In this context, several aspects should be considered:

1. The accuracy of approximation should be good enough to consider SM as a good substitution for OF in further calculations aiming to optimize OF. It means that the values $SM(v)$ should be relatively close to $OF(v)$ for $v \in V$.

2. The time of computing $SM(v)$ should be short enough to ensure that the surrogate model could be used to evaluate a large number of possible assignments using metaheuristics.

3. The benefits of obtaining SM should be significant to compensate for the process of building the surrogate model.

A noteworthy observation is that the first condition can be somewhat relaxed. In fact, in order to ensure that SM can be used as a substitute for OF in optimization algorithms, it is sufficient to ensure that SM approximates OF monotonically, i.e., if $OF(v_1) > OF(v_2)$ for $v_1, v_2 \in V$, then $SM(v_1) > SM(v_2)$. For numerous optimization algorithms, particularly metaheuristics, understanding the relations between objective function values across all possible pairs of points within the domain (rather than knowing the precise values at these points) is sufficient for discovering near-optimal solutions.

If a model satisfying the aforementioned conditions can be built, it could be used as the objective function (assessing values of the original objective function) for different points in the search space. However, in many cases, it might be very difficult to build such a surrogate model.

For instance, one could consider employing a simplified model that omits certain computationally intensive details but meets the necessary conditions to consider the model good enough (e.g., is able to approximate the original model or the real-world process with satisfactory accuracy). In the context of traffic simulation models, macroscopic or mesoscopic models could be computationally more efficient than many microscopic models. However, experiments conducted in 2015 demonstrated that the differences in outcomes

between a newly designed mesoscopic model and a microscopic model implemented in the TSF software (described in Chapter 5) can be substantial (cf. Section 8.3.1.2 and [131]). As a result, a mesoscopic model cannot directly replace TSF.

Also, in general, it is difficult to construct accurate enough surrogate models (like mesoscopic or macroscopic models, as in the case of road traffic) manually, just by analyzing the complex process. Therefore, it is desirable to develop another universal methodology.

Research advancements in artificial intelligence, especially machine learning, led the author of this thesis to consider that machine learning models, like neural networks, can be potentially used as surrogate models for complex processes. They can be trained to approximate the outcomes of models of complex processes on a dataset composed of pairs $(v, OF(v))$ for $v \in V$, where the goal of training is to approximate $OF(v)$ for a given v. There is even a theoretical premise suggesting that neural networks could be good surrogate models in such cases, at least for some complex processes. In 1989, Cybenko proved that any continuous function defined on a compact subset of $\mathbb{R}^n$ can be approximated with an arbitrary accuracy using neural networks with only 1 hidden layer with sigmoid activation function [70]. Currently, this result is known as the *Universal Approximation Theorem*, and there are also results extending it to, e.g., non-Euclidean spaces [190] or other architectures of neural networks, like convolutional neural networks [369], neural networks with ReLU activation function ([146]) or neural networks with a bounded number of hidden layers and a limited number of neurons in each layer [222].

The precise number of neurons, the best architecture, the optimal dataset size, the best training method, and the training time required may vary across different functions. Identifying suitable architectures and training methods to achieve adequate accuracy and efficiency is a critical step (notably, the training set size should be relatively small to ensure a relatively short and cost-effective generation process, and the surrogate model training process should also be relatively brief). However, the *Universal Approximation Theorem* implies that neural networks have the potential to be effective approximators of the objective function OF, so they are worth exploring in this context.

The experiments conducted in Chapter 8 proved the hypothesis that machine learning models can be efficient approximators of objective functions that are computationally demanding and, after training, can be used to evaluate the qualities of points from the domain of the objective function that are explored by metaheuristics. Of course, it is important to validate that the candidate solutions found in this process are really good not only according to the objective function using a surrogate model but also according

to the original objective function. The conducted experiments demonstrated that some machine learning models perform better in this context, while other machine learning models give worse results. Also, some metaheuristics proved to be more efficient.

Figure 7.1 visualizes the whole methodology for optimizing complex processes using metaheuristics that was developed and validated within the research presented in this thesis.

Experiments conducted to validate this methodology were carried out on two complex processes: optimizing urban road traffic through traffic signal control and optimizing cancer treatment via radiotherapy. While the method still exhibits some limitations and requires further validation (cf. Section 9), these initial results have demonstrated its merit for further investigation.

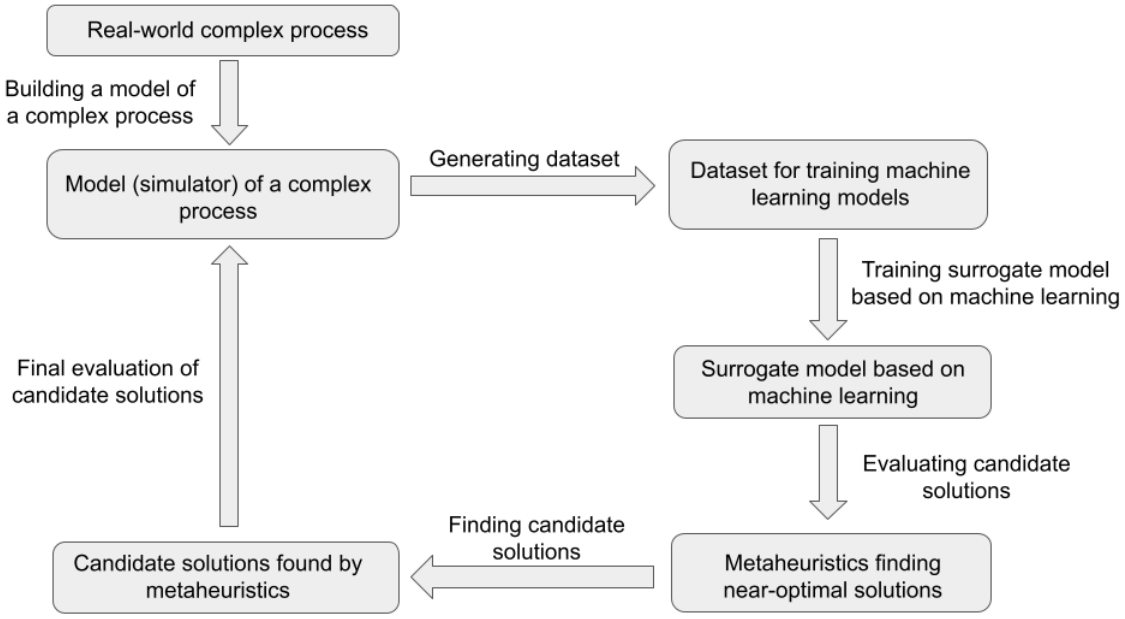


Figure 7.1: Visualization of the methodology for optimizing complex processes using metaheuristics that was developed and validated within the research presented in this thesis.

Chapter 8

Experiments

This chapter presents the most important (milestone) experiments carried out during the work on this thesis - their design, results and conclusions. It is worth emphasizing that it is less than 50% of all experiments performed during the research, as this chapter does not include many preliminary experiments aimed at validating some new ideas or determining reasonable settings of hyperparameters. It is also important to note that all these experiments were designed and coordinated by the author of this thesis, who also analyzed the results, but in some cases the implementation and actual running of the experiments were done by other collaborators (usually members of the TensorCell group [325]), who were also coauthors of the research articles summarizing some experiments.

Section 8.1 contains the definitions and descriptions of the two main problems studied in this thesis: the Traffic Signal Setting problem, and the problem of optimizing radiotherapy for cancer treatment. Section 8.2 presents the the most important aspects of designs of experiments. However, for particular experiments, more detailed description of their design is included in corresponding subsections of the Section 8.3, along with the description of experiments, their results and conclusions. Section 8.4 presents a brief discussion about computational complexity of the introduced methodology. Finally, Section 8.6 presents the author's conclusions after the completion of all experiments.

8.1 Optimization problems

The introduced methodology was experimentally tested on 2 optimization problems, related to 2 complex processes:

1. Traffic Signal Setting problem (defined in Section 8.1.1).

2. The problem of optimizing radiotherapy for cancer treatment (defined in Section 8.1.2).

Based on their definitions presented in Sections 8.1.1 and 8.1.2, both problems can be considered combinatorial optimization problems.

8.1.1 Traffic Signal Setting problem

Definition 8.1 *The **Traffic Signal Setting problem** (or **TSS problem**, in short) is defined as follows:*

- *Given is a directed graph of a road network with traffic signals located at some vertices.*
- *Traffic signals are objects with attributes: duration of the red signal phase (TR), duration of the green signal phase (TG), offset (TO) - values of these attributes are from finite sets and may be modified.*
- ***Traffic Signal Setting (TSS)** - set of values (TG, TR, TO) for all signals in the road network.*
- *Given is a virtual traffic model: cars with initial speeds, positions in some vertices of the road network graph, static routes, and rules of drive on edges.*
- *Given is an objective function OF which calculates the quality of a traffic signal setting.*
- ***Goal**: Find a traffic signal setting for which the value of OF is optimal.*

The formal Definition 8.1 comes from [129] and was proposed by the author of the thesis, but for some particular traffic models the problem was studied earlier and in some cases was proven to be NP-hard [54].

In the context of the notation introduced in Chapter 7, the directed graph of the road network with traffic signals at some vertices, their representation with attributes as well as the representation of cars with speeds, positions, static routes and rules of drive, are parts of the model of a complex system. The initial positions and speeds of cars correspond to the model's state at time 0. The objective function OF can correspond to traffic characteristics like:

- the total time of travel,

- the average delay of all cars (where the delay of a car corresponds to the ratio between the maximum possible speed and the average speed of the considered car),
- the total time of waiting at red signals,
- the average speed,
- the total time of driving at low speed,
- the total throughput (the number of cars that complete driving within a given time).

All those traffic characteristics can be computed for all or only some vehicles in the given region and within a certain time period. Sometimes there can be many types of vehicles and the vehicles can be prioritized, so that their speeds, travel times, delays, or other attributes can contribute to the aggregated value (corresponding to the OF value) with a given weight.

In this context, TSS represents the parameters of the selected OF.

8.1.2 The problem of optimizing radiotherapy for cancer treatment

Definition 8.2 ***The problem of optimizing radiotherapy for cancer treatment*** *is defined as follows:*

- *Given is a virtual model of cancer growth (e.g., EMT6/Ro model [7]).*
- *Radiotherapy protocol is a sequence of pairs (dose of radiotherapy, time of dose administration). Both the doses and the times of their administration can take a finite number of different values.*
- *Given is an objective function OF which calculates the quality of a radiotherapy protocol (the quality can be considered as, e.g., the number of cancer cells after applying the radiotherapy or the likelihood that cancer will be completely cured).*
- ***Goal**: Find a radiotherapy protocol for which the value of OF is optimal.*

This definition has not been introduced earlier in scientific articles, but it is a formalization of the problem that was solved in, e.g., [7] and [264].

8.2 Design of experiments

This Section contains the design of the most important (milestone) experiments carried out during the work on this thesis that are included further in Section 8.3. Section 8.2.1 contains the design of experiments related to the Traffic Signal Setting problems, while Section 8.2.2 contains the design of experiments related to optimizing radiotherapy for cancer treatment.

Since the models of both studied processes (road traffic in cities controlled by traffic signals and cancer growth under radiotherapy treatment) were based on cellular automata (and some cellular automata are Turing-complete [282, 65]), the author of this thesis came up with an idea to investigate how hard it is to approximate the outcomes of cellular automata. However, for simplicity of presentation, the design of these experiments is presented in Section 8.3.3, together with their results and conclusions.

8.2.1 Traffic Signal Setting problem

In the case of the TSS problem, the following experiments are considered in this dissertation:

1. Experiments with genetic algorithms and TSF (Section 8.3.1.1).
2. Experiments with genetic algorithms and a mesoscopic model (Section 8.3.1.2).
3. Training feed-forward fully connected neural networks as surrogate models for TSF (Section 8.3.1.3).
4. Investigating different neural network models and strategies of their training (Section 8.3.1.4).
5. Applying feed-forward fully connected neural network models to optimize traffic signal settings (Section 8.3.1.5).
6. Investigating performance of neural networks and gradient boosting models as surrogate models (Section 8.3.1.6).
7. Traffic signal settings optimization using gradient descent (Section 8.3.1.7).
8. Testing various metaheuristics and surrogate models (Section 8.3.1.8).
9. Experiments with graph neural networks as surrogate models (Section 8.3.1.9).

10. Final experiments with various metaheuristics and surrogate models (Section 8.3.1.10).

The order in which the experiments are presented corresponds to the order in which they were performed (some experiments, however, were performed more or less simultaneously), so the conclusions of some earlier experiments tended to influence the design of subsequent ones.

All the experiments were carried out on a realistic road network of Warsaw taken from the OpenStreetMap service [259]. The main traffic model was the microscopic model described in Section 5.3, implemented in the TSF software presented in Section 5.5. In the experiment described in Section 8.3.1.2, the mesoscopic model presented in Section 5.4 was applied too.

TSF was partially calibrated using real-world data for Warsaw in order to simulate conditions of real traffic in Warsaw during a weekday morning rush hour. In all experiments, the same traffic conditions were assumed (i.e., the number of cars, times of their dispatch, and their routes were constant), and the values of some crucial simulation parameters are summarized in Table 8.1.

Table 8.1: Simulation parameters of TSF's microscopic model (cf. Section 5.3 used in experiments.

Name of the parameter	**Description**	**Value**
Step	Simulated time of a single simulation step	1 second
NrOfCars	Initial number of cars	30000
TimeGap	Time after which new cars start driving	1 second
NewCars	Number of cars that start driving after each TimeGap seconds	20
Steps	Time of a single simulation	600 steps
Acceleration	Maximal increase of cars' speed in a single step	$10\frac{km}{h}$
Crossroad penalty	Parameter responsible for reducing speed before crossing intersection	0.25
Turning penalty	Parameter responsible for reducing speed when turning	0.5

These parameters were set as constants in all experiments so they were non-modifiable control parameters. The only modifiable control parameters corresponded to the traffic signal settings.

As defined in Section 8.1.1, each traffic signal setting is characterized by 3 modifiable attributes: duration of the red signal phase (TR), duration of the green signal phase (TG), and the offset (TO). Offsets are defined as times (in seconds) from a reference point in time (selected as the beginning of the simulation) to the first switch from the red signal state to the green signal state.

In all experiments with metaheuristics, it was assumed that the values TR, TG, and TO are set at the beginning of the simulation (evaluation of the model) and they cannot be changed during the simulation. Therefore, they can be considered values of control parameters that are passed as input to the objective function to calculate the output. Also, in all experiments with metaheuristics, it was assumed that there are no other control parameters, so the values of all other possible settings, including the traffic model, initial locations of vehicles and their rules of drive, the road network structure, etc, were not modifiable.

In most experiments, the durations of the red and green signal phases were also assumed to be constant, set to 62 seconds and 58 seconds, respectively, so were not modifiable. In the first experiment described in Section 8.3.1.1 both values were set to 60 seconds, but in later experiments, these values were adjusted to ensure traffic safety, and the so-called "all-red clearance interval" lasting 2 seconds was introduced after each phase switch from the green signal state to the red signal state before releasing conflicting traffic (this is also visualized in Figure 8.2).

In the case of constant values of duration of green and red signal phases, the offsets were the only modifiable parameters and it was assumed that they can take only integer values from the set $\{0, 1, \ldots, TG + TR - 1\} = \{0, 1, \ldots, 119\}$, which means that for each traffic signal, there were 120 possible values of the offset.

Intersections may have many entries with traffic signals (usually 4), but all these signals must be synchronized in order to prevent car accidents, so it was assumed that the values of the settings of all traffic signals at the intersection should be determined by the values of any single traffic signal at this intersection. In particular, it was assumed that if the state of a single traffic signal changes from red to green, the state of the traffic signal at the opposite entry to the same intersection should also change to green, while the signals at the other (perpendicular) entries should turn to red or be already red to ensure safety. In the case of red signals lasting 62 seconds and green lasting 58 seconds, it was assumed that for all switches between the red signal state to the green signal state, there should be a 2-second long overlap of red signals at all entries.

To give an example, Figure 8.1 visualizes (in TSF's GUI) phases and

entries at the intersection of Banacha and Grójecka streets in Warsaw, while Figure 8.2 presents offsets as well as red and green signal phases for that example intersection.

The assumed representations mean that it is sufficient to specify the settings of a single traffic signal at a given intersection and the settings of other signals at that intersection should be determined. Therefore, since there are only 3 modifiable control parameters (TR, TG, and TO) for each intersection, or 1 (TO) in the cases when TR and TG are set as constant values, all the controllable parameters can be represented as $3 \times N$(or N)-dimensional vectors with integer values of seconds (where N is the number of intersections). These vectors were inputs to the models evaluating traffic conditions in all experiments.

The outputs corresponded to the following traffic characteristics:

- $Time0$: The total time of waiting at red signals;
- $Time20$: The total time of driving slowly (below 20 km/h).

Of course, there can be many other traffic characteristics indicating traffic quality and performance of traffic signal control, like delays, average speed/time of travel, throughput, etc. However, the author observed that these 2 metrics (especially $Time0$ which was the primary metric and was used in all experiments) are the traffic characteristics on which traffic signal control has a big impact, compared to other characteristics, for which other factors, like the interaction between vehicles, may play a significant role too.

In some experiments, these outputs corresponded to aggregated values for all vehicles (on the whole road network of Warsaw), in others, only for vehicles in a selected region. In some experiments, these outputs were calculated using the original, stochastic microscopic model in TSF (described in Section 5.3), but in some other experiments, the stochastic component, inherited from the Nagel-Schreckenberg (NaSch) model [246], was removed to simplify computations. The reason is that the NaSch model was primarily designed to model freeway traffic, without intersections or traffic signals, and its stochastic component is crucial to model the spontaneous formation of traffic jams. However, in urban areas, with dense road networks with multiple intersections and traffic signals, as well as frequent lane changing, traffic jams are usually caused by other factors. Of course, the random behavior of drivers may lead to accidents and other atypical situations, but they are not considered in the NaSch model and in the TSF's microscopic model anyway. Therefore, removing the stochastic component should not have a significant impact on the considered metrics (especially on $Time0$ which was a crucial

metric and was used in all experiments), while it could significantly accelerate computations, so this setting was applied in some experiments.

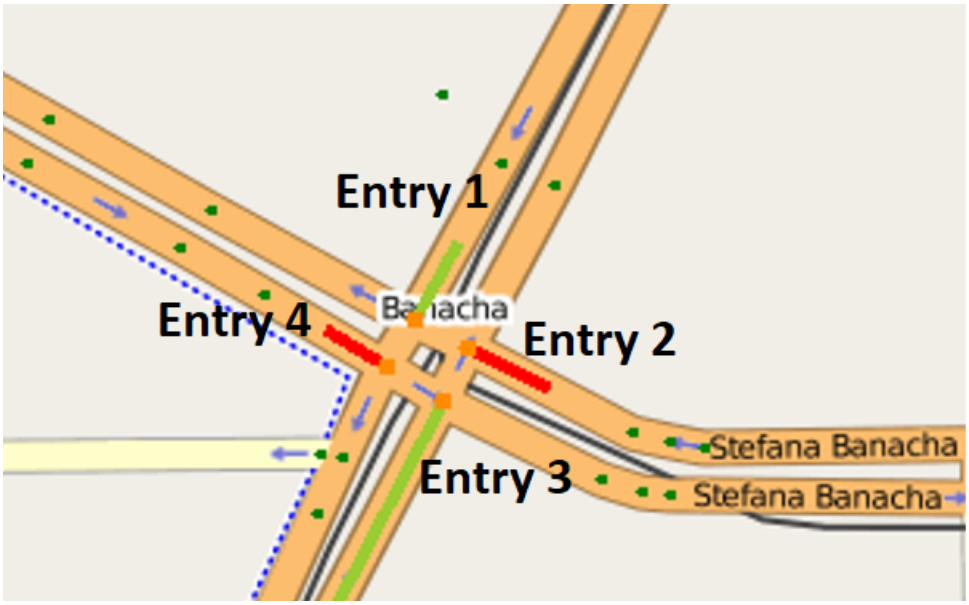


Figure 8.1: Visualization in TSF's GUI of phases and entries at the intersection of Banacha and Grójecka streets in Warsaw.

The real road network of Warsaw contains around 800 intersections with traffic signals [366], but at the time of preparing the simulation tool, only some of them had locations available in the OpenStreetMap service. Some locations of signals were also manually added using the TSF's GUI based on the knowledge of the author of this thesis. In total, only 292 *signal groups* were considered. Signal groups are clusters of traffic signals that are synchronized in such a way that assigning settings (offsets and durations of phases) to a single traffic signal from that group determines the settings for all other signals in that group in order to ensure safety. Usually, signal groups are just intersections, but sometimes intersections might be quite complex, e.g., there can be several intersections located close to each other, so it is reasonable to ensure their synchronization.

Even though the number of intersections with traffic signals was smaller than in the case of a real road network in Warsaw, it was sufficient for research purposes. In some cases, it was assumed that most of the intersections had constant settings, while only some of them were modifiable. For these purposes, 3 scenarios were prepared corresponding to regions of 3 districts in Warsaw: Centrum, Stara Ochota, and Mokotów (therefore, these regions are called in the thesis "Centrum", "Ochota", and "Mokotów", respectively).

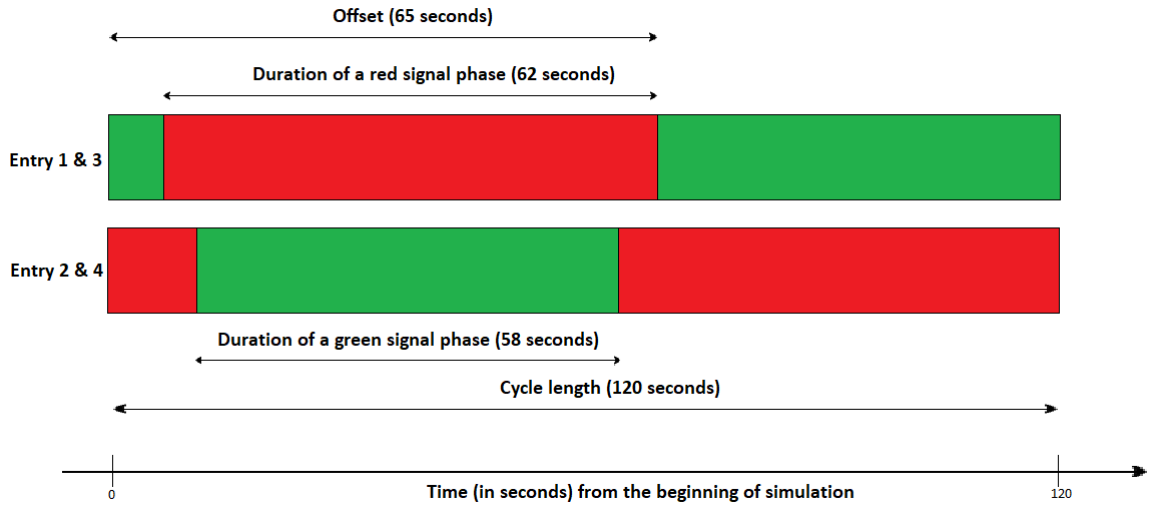


Figure 8.2: Scheme presenting offsets, red and green signal phases for the example intersection in Figure 8.1.

These scenarios contained 11, 21, and 42 signal groups with controllable traffic signal settings. For some experiments, a subset of signal groups from the "Ochota" region, with 15 signal groups, was also considered.

8.2.2 The problem of optimizing radiotherapy for cancer treatment

In the case of the problem of optimizing radiotherapy for cancer treatment, the following experiments are considered in this thesis:

1. Experiments with a simulator of cancer growth and genetic algorithms. (Section 8.3.2.1)

2. Experiments with surrogate models and metaheuristics. (Section 8.3.2.2)

The concept of experimenting on this particular use case arose from the author's consideration that the general framework developed and tested on the use case of traffic signal setting optimization could potentially be applied to use cases related to complex processes in various other domains. Since the Traffic Simulation Framework that was used in traffic signal control experiments is based on cellular automata which are generally considered universal

tools for modeling complex processes, it was natural to look for other complex processes for which a model based on a cellular automaton exists, while the evaluation of the model is time-consuming. During this exploration, the thesis author learned about research carried out by M. Piotrowska and S. Angus [7] in which a model of EMT6/Ro cell line (derived from the Experimental Mammary Tumor-6 cell line that was isolated from the breast of a mouse with a mammary tumor) based on a cellular automaton was applied in combination with genetic algorithms to find good radiotherapy protocols for the given 10 test tumors. The model was an extension of the MCS asynchronous stochastic cellular automata model of EMT6/Ro dynamics that was earlier introduced in [273] by adding a calibrated multi-fraction irradiation module. Since the model is stochastic, the authors of [7] decided to evaluate all considered protocols across a multi-tumor case library consisting of ten 10-day-old tumors developed in-silico as described in [8]. The radiotherapy protocols were also represented as pairs of doses and times of their application. The outcome of the simulation for each initial tumor was the state of each cancer at the end of the 10-day growth period. It was also assumed that radiotherapy can be applied only during the first 5 days of treatment to consider the long-term impact of the treatment. The optimized value corresponding to the objective function was the average number of cancer cells after the 10-day growth period. The optimization was carried out using genetic algorithms, but due to significant time complexity, it was possible to run a relatively small number of experiments and evaluate a relatively small number of candidate treatment protocols. However, the results indicated that it is possible to find reasonably good protocols and there might be potential for further improvements. Hence, it turned out that this could be a good use case to evaluate the developed methodology.

The initial model based on cellular automata was implemented in MATLAB and it turned out the implementation can be highly optimized thanks to using C++ and GPUs. The optimization was performed by a team of students supervised by the author of this thesis [15] and resulted in a c. 717-fold acceleration of candidate protocol evaluation times. It gave the opportunity to run many more experiments, evaluate the quality of optimization algorithms much faster, test new algorithms, and find better radiotherapy protocols, as a consequence.

8.3 Experiments and their results

This section contains the descriptions, designs, results and conclusions from all most important experiments related to the Traffic Signal Setting prob-

lem (Section 8.3.1) and the problem of optimizing radiotherapy for cancer treatment (Section 8.3.2).

8.3.1 Traffic Signal Setting problem

8.3.1.1 Experiments with genetic algorithms and TSF

The initial experiments were carried out using genetic algorithms and the TSF software (presented in Section 5.5) as a tool for evaluating the quality of traffic signal settings using a microscopic model. They were also described in [120] and [121].

These experiments were conducted before the main methodology based on the use of surrogate models (presented in Chapter 7) was invented to see if the use of metaheuristics, such as genetic algorithms, could be a good approach, and to assess whether it was worth further study.

8.3.1.1.1 Setup of experiments

The goal of the experiments was to find good settings for traffic signals on a road network in Warsaw with 292 intersections. All experiments were conducted with a microscopic model implemented in TSF (cf. Section 5.3), using its stochastic version, and simulation parameters listed in Table 8.1. Due to the fact that the results of a single evaluation of a traffic signal setting were non-deterministic, each signal setting was evaluated 5 times and the results (values of the fitness function) were averaged.

The only setting that distinguished simulations was the traffic signal setting. A green phase duration and a red phase duration of a single signalization were constant and set to 60 seconds, so the total cycle duration was set to $60 + 60 = 120$ seconds. The only parameter that distinguished traffic signal settings was the offset. The allowed values of offsets were from the set $O = \{0, 1, \ldots, 119\}$, so the feasible solutions were encoded as points from the set O^{292} (as mentioned in Section 8.2.1, it is sufficient to encode a setting of only a single traffic signal from the given signal group and the settings of the other signals at that intersection are determined in order to ensure traffic safety). These 292−dimensional vectors were encoded as chromosomes for genetic algorithms, where each position in a vector (gene) corresponded to the offset of a single signal group.

The first batch of experiments was run with the following settings:

1. Fitness functions:

- $Time20$: The total time when cars have a speed lower than 20 km/h computed using a stochastic version of the microscopic model in TSF (cf. Section 5.3). Each signal setting was evaluated 5 times and the results were averaged.
- $Time0$: The total time when cars have a speed of 0 km/h computed using a stochastic version of the microscopic model in TSF (cf. Section 5.3). Each signal setting was evaluated 5 times and the results were averaged.

2. Size of the initial population: 100 (randomly generated) chromosomes.
3. Selection operator: Square-root selection - taking $\sqrt{N}$ best chromosomes from the population of N chromosomes.
4. Crossover operator: Uniform crossover - all selected $\sqrt{N}$ chromosomes cross with each other and form new N chromosomes.
5. Mutation operator: Uniform mutation - for each gene (offset), with a given probability ($p = 0.05$) a randomly selected (from a uniform distribution) offset value is assigned.

8.3.1.1.2 Results of experiments

Table 8.2 presents the results of these experiments: values of fitness functions for the best individuals in consecutive iterations for 2 considered metrics.

Table 8.2: The best values of fitness functions in consecutive iterations

Step of the evolution	*Time0*	*Time20*
1	5937535	8381108
2	5897720	8350837
3	5884938	8328534
4	5855378	8292620
5	5840454	8292925
6	5811827	8242777
7	5794758	8228153
8	5780861	8234131
9	5752666	8228039

Due to high computational complexity, it was possible to conduct only 9 steps of evolution (for both fitness functions independently). Still, the results indicated that the progress in values of the fitness functions may appear

in consecutive iterations. In the case of the fitness function *Time0*, the improvement was 3.11%, and in the case of the fitness function *Time20*, the improvement was 1.82%.

8.3.1.1.3 Conclusions from experiments and follow-up experiments

One of the conclusions from these experiments was that it can be possible to find better traffic signal settings using genetic algorithms. However, another conclusion was that the method may be too slow to optimize signal settings in a reasonable time. The method can already help find relatively good default signal settings for typical, repeatable traffic conditions when it is not necessary to operate in real-time, but even for the purpose of running the experiments necessary for this thesis, high computational complexity was problematic and it was difficult to run multiple experiments testing many different metaheuristics in many different settings. The analysis of performance led to the conclusion that the most expensive component of computations is related to running traffic simulations in a microscopic model. For example, evaluating a single traffic signal setting for the whole road network of Warsaw took about 2 minutes on standard machines in a computer laboratory available to the author of this thesis at that time. In the summarized experiments, a genetic algorithm had to evaluate 900 settings for each fitness function. Evaluations of settings in a single iteration of the genetic algorithm can be potentially parallelized but it requires greater computational power and might be expensive and technically difficult to scale. Later, such approaches were tested too and it was possible to parallelize the process of running multiple simulations within the same iteration at the same time within a single iteration of the genetic algorithm using tools such as Apache Spark [299] and a multi-cloud infrastructure governed by the MELODIC platform [158]. However, due to the construction of the genetic algorithm, parallelization is limited because usually evaluation of chromosomes from the next iteration can be started only after the evaluation of chromosomes from the previous iteration is completed. Hence, even in the case of the complete population-level parallelization, the running time is proportional to the *number of iterations* $\times$ *time of running a single evaluation*. In addition, in the case of other metaheuristics, it might be necessary to run even more iterations than in the case of genetic algorithms.

Therefore, it was concluded that it is reasonable to investigate how evaluating a single signal setting can be accelerated. One of the options could be accelerating the simulation of the microscopic model. After optimizing the implementation of the model, it was possible to speed up the evaluation of a single traffic signal setting a few times. In 2022, it took about 30 seconds on

a machine with Processor Intel(R) Core(TM) i7-4702MQ CPU @ 2.20GHz, 32GB RAM (which was already faster than the machines used for experiments several years earlier), but it was still quite slow taking into account that at least several iterations of the genetic algorithms are necessary.

Since the model is based on cellular automata, the next idea was that it could be accelerated using GPUs or multithreading. However, it turned out that due to the complexity of the model and the interactions between cars, this is technically complicated and may not bring the expected advantage. The implemented microscopic model is still relatively simple, but it may turn out that for other complex processes, the underlying simulation models are much more difficult to parallelize. Since the goal of this research was to develop a universal methodology that could be applied to other complex processes, it was concluded that it is better to think about some other approaches.

Further research on traffic modeling led to the conclusion that it might be reasonable to implement another traffic model that approximates the original microscopic model but could be evaluated faster. The considered microscopic model is already relatively simple, so it was not expected that any other reasonable microscopic agent-based model, that computes the positions and speeds of each vehicle at each step could be evaluated significantly faster. On the other hand, macroscopic models (described in Section 4.2.1.1.2) are good at capturing relationships between aggregated traffic characteristics such as flow, density, and speed, but it is more difficult for them to capture the effects of traffic signals. Also, the existing macroscopic models usually concern traffic on straight road segments, but developing good macroscopic models for urban areas and realistic road networks with many intersections and interactions between vehicles is more challenging. On the other hand, implementing an intermediate, mesoscopic model (cf. Section 4.2.1.1.2) seemed to be a reasonable approach, so this was the next step in the considered experiments.

Another conclusion was that it might be reasonable to reduce the size of the space of possible traffic signal settings. This could be done in 2 ways:

1. By reducing the number of possible values of signal offsets;

2. By considering smaller regions, with fewer intersections.

In general, for N intersections with traffic signals and V possible settings of a single signal, the size of the space of possible settings is V^N. Exponential functions grow faster than polynomial functions, so reducing the number of considered intersections could reduce the size of the space more than the analogous reduction of the number of intersections. Intuitively, this gives more benefits, but it may be difficult to build a good solution for a larger

area from a good solution for a smaller area while reducing the number of possible offsets may still allow finding good solutions, so it is good to consider both options.

The number of possible settings for a single intersection can be reduced, e.g., by assuming higher granularity: instead of considering values of offsets from the set $\{0, 1, 2, \ldots, 119\}$, one can consider only values from the set $\{0, 5, 10, 15, \ldots, 110, 115\}$. This can help, but it is not a scalable solution: setting the granularity too low can lead to overlooking promising regions of the explored space.

In the case of considering smaller regions, it might be necessary to merge the solutions for smaller regions into solutions for larger regions. However, in the case of developing a method that could do this efficiently, this approach may be scalable and transferable to other larger regions. Also, in traffic engineering practice, it is typical to consider traffic signal control in smaller regions (e.g., major corridors) rather than in the entire metropolitan area [134].

In further experiments, values of offsets from smaller sets were considered (e.g., $\{0, 5, 10, 15, \ldots, 110, 115\}$, $\{0, 10, \ldots, 110\}$), but this change did not result in significantly better performance of genetic algorithms (in terms of finding better solutions in the same time), so in the next experiments, the focus was rather on considering smaller regions, "Centrum", "Ochota", and "Mokotów", with 11, 21, and 42 intersections, respectively. However, the approaches based on reducing granularity should be also investigated further in combination with other techniques, as presented in Section 9.1.

8.3.1.2 Experiments with genetic algorithms and a mesoscopic model

As justified in Section 8.3.1.1.3, since evaluating traffic signal settings is time-consuming, it was natural to build some other models that could be evaluated faster. In these experiments, genetic algorithms were combined with a mesoscopic model introduced in Section 5.4.

Before the main experiments with genetic algorithms, some initial experiments were run to compare the microscopic model described in Section 5.3 and the mesoscopic model. The time required to evaluate a single signal setting on the same machine was about 30 seconds for a microscopic model and about 1.5 seconds for a mesoscopic model. Taking into account that using the stochastic variant of the microscopic model may require running multiple simulations and averaging results, the time reduction was already significant. However, due to the simplicity of the mesoscopic model, the differences in the results of evaluations ($Time0$) were too big to consider the mesoscopic

model a sufficiently good surrogate model for the microscopic model. The relative difference was usually in the range $10 - 30\%$, and depended on the signal settings and the considered region for computing $Time0$. As a consequence, the settings that are considered as good according to the microscopic model do not have to be good according to the mesoscopic model and vice versa. However, building a more accurate mesoscopic surrogate model may also lead to a longer time for evaluating signal settings.

One of the conclusions was that it might be reasonable to look for another approach to building surrogate models of microscopic simulation models. This is motivated not only by the significant difference between the results of simulations using microscopic and mesoscopic models, but also by the fact that building a mesoscopic model for complex processes is not always an easy task.

On the other hand, both models are only approximations of the real-world process and even though the microscopic model seems to be more detailed, it does not mean that its results are always closer to real-world traffic outcomes. Therefore, it was reasonable to analyze how the simple mesoscopic model performs in combination with genetic algorithms.

These experiments and their results were also described in [131].

8.3.1.2.1 Setup of experiments

These experiments were carried out on 2 regions:

- The whole road network of Warsaw (region A);
- "Ochota" region in Warsaw (region B).

In the case of the "Ochota" region, a slightly smaller region than the default one (a subset) was selected - it had 15 signal groups. The whole road network of Warsaw was also adjusted somewhat - 2 signal groups (representing 2 close intersections) were merged into a single group to ensure additional synchronization of signals ensuring traffic safety, so there were 291 signal groups considered. Modification of traffic signal settings (TSS) in one region and computation of the fitness function for another region was also considered (in the case of modifying signals in region B, the remaining signals were set to constant, default values).

Similarly as in experiments described in Section 8.3.1.1, only signal offsets were modifiable. However, this time 2 possible differences between consecutive values of offsets were considered, 1 and 10, giving 2 sets of possible offsets: $\{0, 1, 2, \ldots, 119\}$ (set S_1) and $\{0, 10, 20, \ldots, 110\}$ (set S_2).

The durations of green signal phases and red signal phases were set to 58 and 62 seconds, respectively (following the scheme presented in Figure 8.2).

This time, the considered model evaluating traffic signal settings was the simple mesoscopic model introduced in Section 5.4. The main fitness function considered in experiments was $Time0$ - the total time of waiting at red signals during 10 minutes of simulated traffic in the considered region. $Time20$ metric that was used in the previous experiments (Section 8.3.1.1) is not calculable in the mesoscopic model, so the focus was on $Time0$.

Most of the simulation settings were the same as in the experiment described in Section 8.3.1.1. However, besides the default initial number of vehicles (30000), in some experiments, 3 other values were considered: 10000, 50000, 70000. Also, the number of vehicles starting to drive in each step was set to 10. For the same number of vehicles, the routes were always the same.

In terms of genetic algorithms, similar settings as in Section 8.3.1.1 were used. As a selection operator, the square-root operator was chosen. The reason for that was to get rid of poor traffic signal settings as soon as possible and enhance signal settings that are better and may lead to better solutions. As a crossover operator, the uniform crossover was chosen: $\sqrt{N}$ chromosomes were crossed with each other to obtain a new set of N chromosomes, and for each pair of crossing chromosomes and each gene, the offset value was selected from the corresponding gene in the randomly chosen chromosome. As a mutation operator, the uniform mutation was chosen: for each gene, with a given probability the value was set to the value selected randomly (with a uniform distribution) from the set of possible values. This time, 3 values of probability were considered: $\frac{1}{100}$, $\frac{1}{20}$, and $\frac{1}{5}$. Two sizes of the initial population were considered: 100 and 400 individuals. Each run of the genetic algorithm had 50 iterations.

In total, 4 batches of experiments were considered:

1. TSS modified in region A, $Time0$ computed using the mesoscopic model in region A.
2. TSS modified in region A, $Time0$ computed using the mesoscopic model in region B.
3. TSS modified in region B, $Time0$ computed using the mesoscopic model in region A.
4. TSS modified in region B, $Time0$ computed using the mesoscopic model in region B.

In total, there were 192 runs of the genetic algorithm, 48 for each batch. Table 8.3 summarizes considered values of the most important parameters.

Table 8.3: Simulation parameters and their values used in experiments

Name of the parameter	**Possible values**
Duration of traffic simulation	600 seconds
Initial number of cars in the simulation	10000, 30000, 50000, 70000
Number of cars that start driving at each step	10
Interval between possible values of offsets	1 (Set S_1), 10 (Set S_{10})
Optimized value	$Time0$
Number of iterations	50
Population size	100, 400
The region in which traffic signal settings are modified	A (the whole road network), B (a smaller region, "Ochota")
The region in which fitness function is computed	A (the whole road network), B (a smaller region, "Ochota")
Selection operator	Square-root selection
Crossover operator	Uniform crossover
Mutation operator	Uniform mutation
Probability of mutating a single gene	0.01, 0.05, 0.2

This time, the experiments were run on a high-performance computing cluster at the University of Rzeszów[1].

8.3.1.2.2 Results of experiments

When signals are modified in a smaller region (B) and the fitness function is calculated over the entire road network (region A), the improvement is small, in the range $0.4657\% - 0.7182\%$. One of the possible reasons is that changing traffic signal offsets in small regions have usually relatively little effect on global traffic characteristics.

When signals are modified in a smaller region (B) and the fitness function is computed in a smaller region (B), the improvement is significant, in the range $6.8567\% - 15.5888\%$, depending on the values of the other parameters.

[1]The cluster consisted of 40 computational nodes with 2 processors (INTEL Xeon E5-2620, 6 cores, 2:0 GHz, 15MB cache) per node. The computational power is 7.5 TeraFLOPS, 1032GB RAM (258x 4GB DDR3 1333MHz)

When signals are modified over the whole road network (region A) and the fitness function is computed over the whole road network (A), the improvement is also significant, in the range $5.7101\% - 18.1204\%$, depending on the values of parameters.

When signals are modified over the whole road network (A) and the fitness function is computed in region B, the improvement is in the range of $26.6716\% - 51.4472\%$ (much better than in the case of reconfiguring traffic signals only in region B). However, it may cause larger delays on the rest of the road network, thus this method alone cannot be applied but should be further investigated.

8.3.1.2.3 Conclusions from experiments

The conducted experiments proved that genetic algorithms can be successful in finding better traffic signal settings also in the case of the mesoscopic model. The exact improvement may depend on many factors, including the region of modifying signal settings and the region of computing the fitness function, but the behavior is as expected.

The mesoscopic model can evaluate the traffic signal settings much faster than the microscopic model, but for some complex processes building good mesoscopic models can be difficult - it may depend on the specific complex process, so this approach is not universal. However, building microscopic models can also be challenging, and even though the microscopic models are more detailed and seemingly should produce results closer to the real world, investigating the mesoscopic models can be also considered an interesting research area to improve the optimization of complex processes.

In the case of the road traffic in cities, the error of approximating the microscopic model using a simple mesoscopic model was too big compared to the improvements achieved using genetic algorithms, so in the next experiments, the focus was on building surrogate models using machine learning techniques, that seem to be more universal and gave very good results in initial experiments.

8.3.1.3 Training feed-forward fully connected neural networks as surrogate models

As mentioned in Section 8.3.1.2, applying mesoscopic (or macroscopic) models may lead to a much faster evaluation (or approximation) of the objective function but this method has at least 2 weaknesses:

- For some complex systems it might be difficult to develop mesoscopic or macroscopic models that could be evaluated sufficiently fast.

- Mesoscopic or macroscopic models do not have to give sufficient accuracy in approximating the values of the objective function.

Therefore, it was reasonable to think about an alternative approach that could be universal and applicable to a broad range of complex processes. As discussed in Chapter 7, a good approach could be based on neural networks that are known to be good universal approximations of continuous functions on compact subsets of $\mathbb{R}^n$ (and even though the space of traffic signal settings is discrete, it can be extended to a continuous function on a compact subset of $\mathbb{R}^n$, so the universal approximation theorem holds).

However, as discussed in Chapter 7, this theoretical property does not imply that using neural networks to approximate outcomes of simulating traffic using a microscopic model is efficient enough, as the required size of the training set or time of training might be huge. Hence, the goal of the next experiment was to validate how efficient in approximating outcomes of traffic simulations neural networks can be.

8.3.1.3.1 Setup of experiments

The experiments were performed on a dataset generated by simulating 600 seconds of realistic traffic using a deterministic variant of TSF's microscopic model in a setting with 42000 vehicles (30000 cars starting at the beginning and 20 new cars starting every second). In each simulation, the same traffic scenario was simulated, with routes generated based on the same origin-destination matrix. The only setting that was different in each simulation was a traffic signal setting in the "Ochota" region (a part of the Stara Ochota district in Warsaw, covering 15 signal groups with traffic signals). In the experiment, the only modifiable component of the signal setting was its offset - the durations of the red and green signal phases were set as constants: 62 and 58 seconds, respectively. For all other intersections, the values of the phase durations were the same, while the offsets were also set as constants (10 seconds for the representative signal at each intersection - the offsets for other signals at the same intersection had to be determined to ensure traffic safety, as discussed in Section 8.2.1).

Therefore, each traffic signal setting in the studied region can be represented as a vector of 15 offsets - integers between 0 and 119, which were later the features of trained models.

In order to generate training and test sets, a large number of traffic simulations were run using TSF's microscopic model described in Section 5.3. In each simulation, the value of the objective function was calculated - it was the total time of waiting at red signals (time spent with a speed of 0 km/h)

in the considered region during the whole simulation, summed up over all vehicles.

First, 117033 traffic signal settings were randomly selected (for each of the 15 offsets, the value was selected from the set $\{0, 1, \ldots, 119\}$ with a uniform distribution). The values of the objective function were computed for all of them by using TSF's microscopic simulation model.

In this batch of experiments, 12 configurations of feed-forward fully connected neural networks were considered, with 3 different values of the learning rate (0.001, 0.01, 0.1) and 4 different configurations of neurons: [100, 100] [100, 100, 100], [200, 200], [200], where the configuration $[x_1, x_2, \ldots, x_n]$ means that the neural network has n hidden layers, x_1 units in the first layer, x_2 inputs in the second layer, and so on. In each case, the input layer was composed of 15 units (corresponding to traffic signal settings) and the output layer consisted of 1 unit. For all neurons, the activation function was ReLU [110]. It is worth noting that before determining these 4 configurations, many preliminary experiments were conducted to determine lower and upper bounds on the values of the number of layers and the number of neurons that were worthy of further study.

For each configuration, 5-fold cross-validation was applied. Each time, the generated dataset was randomly divided into a training set (93626 settings, i.e., about 80% of all settings) and a test set (23406 settings, i.e., about 20% of all settings). The neural networks were trained using Adam optimizer [183] with a batch size of 100. The loss function was the mean squared error (MSE) [26], while the metrics used for evaluation on the test set were MAPE (mean absolute percentage error) and MAXAPE (maximum absolute percentage error) defined as:

$$MAPE = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{A_i - F_i}{A_i} \right| \times 100\% \tag{8.1}$$

$$MAXAPE = \max_{i \in \{1,\ldots,n\}} \left| \frac{A_i - F_i}{A_i} \right| \times 100\% \tag{8.2}$$

Here, n is the number of data points, A_i is the actual value at the i-th data point, and F_i is the forecasted value at the i-th data point.

These experiments were performed using the Torch library [328].

8.3.1.3.2 Results of experiments

In the best scenario, for the learning rate equal to 0.01, neurons = [100, 100, 100], MAPE= 1.56%, and MAXAPE= 9.18% were achieved. A similar result was

achieved for the learning rate equal to 0.1 and neurons = $[100, 100, 100]$: MAPE= 1.74% and MAXAPE=8.47%.

The complete results are presented in Tables 8.4, 8.5 and 8.6.

Table 8.4: Results of experiments for the learning rate 0.001.

Metrics	[100, 100, 100]	[100, 100]	[200, 200]	[200]
MAPE	2.03%	2.39%	1.75%	4.44%
MAXAPE	10.08%	13.61%	8.89%	22.78%

Table 8.5: Results of experiments for the learning rate 0.01.

Metrics	[100, 100, 100]	[100, 100]	[200, 200]	[200]
MAPE	1.56%	2.28%	2.11%	3.98%
MAXAPE	9.18%	11.72%	10.8%	20.06%

Table 8.6: Results of experiments for the learning rate 0.1.

Metrics	[100, 100, 100]	[100, 100]	[200, 200]	[200]
MAPE	1.74%	2.62%	2.01%	3.76%
MAXAPE	8.47%	13.31%	11.31%	18.64%

The time required for training using standard GPUs was relatively short (from about 15 minutes to a few hours, depending on the experiment and settings) and after training, the time required for the inference to get the value (approximated simulation outcome) for a given traffic signal setting was below 1 millisecond, which is a few orders of magnitude faster than in the case of running traffic simulations.

8.3.1.3.3 Conclusions from experiments

The first experiments with feed-forward fully connected neural networks were preliminary but confirmed that relatively small neural networks are sufficient to approximate some outcomes of microscopic traffic simulations (the total times spent while waiting at red signals with speed $0\frac{km}{h}$ in a given area) with relatively good accuracy. Also, the time required for the inference to get the value (approximated simulation outcome) for a given traffic signal setting was below 1 millisecond, a few orders of magnitude faster than in the case of running traffic simulations. Of course, the time required for training the neural network as well as the time necessary to generate the training set were significantly longer, so their reduction was also one of the goals of

the next experiments. However, these processes have to be performed only once and after that, the trained neural network can be used multiple times in many experiments related to traffic optimization using metaheuristics which significantly accelerates experiments.

More information about these experiments can be found in [129].

8.3.1.4 Investigating different neural network models and strategies of their training

In the next batch of experiments, the goal was to investigate different neural network models and strategies of their training and to answer the following research questions:

- What is the impact of the size of the training set?
- What is the impact of the number of layers?
- What is the impact of the number of neurons?
- What is the impact of the learning rate?
- What is the impact of the dropout technique?

In order to answer these questions and analyze the impact on the accuracy (MAPE and MAXAPE) of the trained models and the training time, different sizes of training sets, different numbers of neurons and layers, and different values of dropout and learning rates were taken into account.

8.3.1.4.1 Setup of experiments

The experiments were conducted on the same dataset as experiments described in Section 8.3.1.3. However, this time, the dataset was organized differently. First, it was randomly divided into a training set (intended to be used to train different models of neural networks), consisting of 81920 cases, and a validation set (intended to help in selecting the best model of neural network from a set of investigated models), consisting of 20479 cases. The remaining 14634 cases were intended to be a test set for later evaluation. This time, one of the investigated factors was how the size of the training set influences the process of training neural networks (e.g., final accuracy, speed of learning), so in some cases, 2 smaller training sets were used too: these were subsets of the whole training set, with 10240 and 30720 elements, respectively.

The following hyperparameters were considered:

1. Size of the training set - 3 values were considered: 81920, 30720, and 10240.

2. Configurations of neurons in hidden layers - 6 settings were investigated: [100, 100, 100], [100, 200, 100], [200, 300, 200], [300, 400, 300], [100, 150, 200, 150, 100], [50, 100, 200, 300, 200, 100, 50]. Each neural network architecture was a feed-forward fully connected neural network. All units had a ReLU activation function. In each case, the input layer was composed of 15 units (corresponding to traffic signal settings) and the output layer consisted of 1 unit.

3. Learning rate in the training process - 4 values were investigated: 0.1, 0.01, 0.001, 0.0001.

4. Dropout technique (randomly removing some units to prevent overfitting [316]) - 4 values of the dropout probability were investigated: 0.05, 0.1, 0.15, 0.2.

In total, there were $3 \times 6 \times 4 \times 4 = 288$ models of training neural networks. To train all these models, a standard grid search method considering all possible combinations of settings was applied. As in the previous experiments (Section 8.3.1.3, [129]), some preliminary experiments were run aiming to determine which parameters and values should be chosen for further investigation.

In order to run the experiments, a Python tool was prepared based on the TensorFlow library [326] (thus, the tool was called: *TensorTraffic*).

All experiments were run on an NVIDIA server with 4× P100 PCIe, some on NC6 machines in a Microsoft Azure cloud with a GPU K80 [228]. The loss function was again the mean squared error (MSE) [26]. Training of neural networks was performed on GPU by DNNRegressor evaluator from TensorFlow library, using Adam optimizer [183] with a batch size 10240.

To analyze and compare the trained neural networks, MAPE and MAXAPE metrics (defined in Section 8.3.1.3) were calculated on the validation set or the test set, depending on the objective.

Neural networks were trained for 10^6 steps (a single step was a training performed on a single batch set, consisting of 10240 elements from the training set), but to finish training when MAPE on the validation set was not improving, an early stopping mechanism was introduced. To check the progress of the training and to stop the training if the MAPE did not decrease, the neural networks were evaluated after every 10000 steps on a validation set consisting of 20479 cases.

After the training was finished, the trained neural network was evaluated on the test set and MAPE and MAXAPE were computed.

8.3.1.4.2 Results of experiments

Tables 8.7 - 8.10 present dependencies of results on values of parameters used in the training process: the size of the training set (Table 8.7), neural network architecture (Table 8.8), learning rate (Table 8.9) and dropout (Table 8.10). Each table has 4 columns: the first column contains considered values of a given parameter, while the second, third, and fourth columns contain information about the average and maximum errors obtained on the test set, and about the training time (number of steps used for training - the maximum possible number of steps or the number of steps to trigger the early stopping mechanism), respectively. The errors considered here are MAPE (for average errors) and MAXAPE (for maximum errors), as defined in Section 8.3.1.3.

Each row contains information about the average, median, best, and worst values, obtained for all possible models of training with a corresponding value of a given parameter. For instance, the second column of the second row of Table 8.7 contains the average MAPE, the median MAPE, the best MAPE, and the worst MAPE, among all errors obtained on the test set for all neural network models (96 models, in total) trained using the training set with 81920 elements. Table 8.11 presents the top 5 results according to MAPE obtained on the test set, while Table 8.12 presents the top 5 results according to MAXAPE on the test set.

Table 8.7: Dependence of the results on the size of the training set.

Size of the training set	Avg MAPE Med MAPE Best MAPE Worst MAPE	Avg MAXAPE Med MAXAPE Best MAXAPE Worst MAXAPE	Avg nr of steps Med nr of step Best nr of steps Worst nr of steps
81920	3.13% 1.91% **1.18%** 15.51%	14.49% 10.85% **7.09%** 31.48%	66907 40000 20000 250000
30720	**3.1%** **1.76%** 1.26% 15.59%	**14.11%** 10.29% 7.55% **31.11%**	**65258** 40000 20000 **220000**
10240	3.26% 1.81% 1.24% **13.7%**	14.94% **6.8%** 10.5% 39.24%	73936 40000 20000 270000

Table 8.8: Dependence of the results on the neural network structure.

Neural network structure	Avg MAPE Med MAPE Best MAPE Worst MAPE	Avg MAXAPE Med MAXAPE Best MAXAPE Worst MAXAPE	Avg nr of steps Med nr of step Best nr of steps Worst nr of steps
[100, 100, 100]	2.55% 1.8% 1.32% 7.64%	13.1% 9.88% **6.8%** 31.09%	87200 60000 20000 270000
[100, 200, 100]	2.48% 1.65% **1.18%** 7.64%	12.86% 9.27% 7.09% **13.14%**	86417 60000 20000 260000
[200, 300, 200]	2.62% **1.51%** 1.3% 7.64%	13.86% 9.2% 7.84% 37.95%	77872 40000 20000 240000
[300, 400, 300]	**2.21%** 1.58% 1.3% **5.53%**	**12.65%** 9.29% 7.33% 30.97%	75000 40000 20000 210000
[100, 150, 200, 150, 100]	3.76% 2.03% 1.42% 8.48%	15.73% **7.53%** 11.26% 39.24%	47292 **30000** 20000 180000
[50, 100, 200, 300, 200, 100, 50]	5.37% 2.96% 1.97% 15.59%	18.86% 15.83% 9.07% 31.13%	**37872** **30000** 20000 **90000**

8.3.1.4.3 Conclusions from experiments

According to Table 8.7, the size of the training set does not influence final results significantly. An important conclusion is that a smaller training set, consisting of 10240 elements, can be sufficient to achieve acceptable accuracy of approximation in this case. Since generating a training set is the most costly and time-consuming part of the whole approach (the training set is generated using microscopic simulations), it gives the opportunity to significantly reduce the time and cost of that process. This may help in generating training sets in real time and help in many engineering applications. One of the drawbacks is that the average training time is slightly longer in the case of a set consisting of 10240 elements, but the difference is not significant (the

Table 8.9: Dependence of the results on the learning rate value.

Learning rate	Avg MAPE Med MAPE Best MAPE Worst MAPE	Avg MAXAPE Med MAXAPE Best MAXAPE Worst MAXAPE	Avg nr of steps Med nr of step Best nr of steps Worst nr of steps
0.1	5.8% 5.69% 2.63% 7.64%	26.61% 29.42% 12.79% 39.24%	**25493** **20000** 20000 **60000**
0.01	**1.84%** 1.71% 1.24% **3.42%**	10.61% 10.17% 7.09% **18.73%**	36806 30000 20000 80000
0.001	2.43% 1.62% **1.18%** 12.11%	**10.15%** 9.4% **6.8%** 19.94%	59577 60000 20000 180000
0.0001	2.58% **1.61%** 1.31% 15.6%	10.68% **9.27%** 7.27% 25.69%	149730 180000 20000 270000

median and the minimum number of steps are identical as in the case of the training set with 81920 elements).

According to Table 8.8, adding new layers and neurons does not improve the approximation accuracy significantly. Moreover, in the case of 5 or 7 layers, the average and maximum errors may not be acceptable for practical applications. Architectures with 3 layers usually perform better, which can be also seen in Tables 8.11 and 8.12. In the case of all networks with 3 layers, accuracy (MAPE and MAXAPE) and required training times do not differ significantly. The architecture $[100, 200, 100]$ gave the best MAPE and quite good average MAPE as well as significantly better performance for the worst cases (Worst MAXAPE) than other models, while the architecture $[300, 400, 300]$ gave also very good MAPE and the best average MAXAPE, so these 2 architectures seem to be best.

According to Table 8.9, in terms of accuracy, the best values of the learning rate parameter are 0.01 and 0.001. This is also supported by results in Tables 8.11 and 8.12. In addition, the required training time significantly increases with decreasing learning rate, which implies that the value 0.01 might be the best choice.

According to Table 8.10, the best values of the dropout parameter are

Table 8.10: Dependence of the results on the dropout probability.

Dropout probability	Avg MAPE Med MAPE Best MAPE Worst MAPE	Avg MAXAPE Med MAXAPE Best MAXAPE Worst MAXAPE	Avg nr of steps Med nr of step Best nr of steps Worst nr of steps
0.05	4.06% 2.42% **1.18%** 15.6%	14.72% 11.61% **6.8%** **31.13%**	**64167** **40000** 20000 270000
0.1	2.81% **1.61%** 1.3% 11.53%	14.06% **9.63%** 7.53% 31.48%	77568 60000 20000 260000
0.15	**2.73%** 1.74% 1.34% **7.63%**	**13.89%** 10.29% 7.33% 33.94%	65278 **40000** 20000 **220000**
0.2	3.06% 1.89% 1.39% 7.64%	15.37% 10.6% 7.85% 39.24%	67286 **40000** 20000 230000

0.1 and 0.15, on average. However, Tables 8.11 and 8.12 show that the best results were achieved for the value of 0.05. The reason is that for the value of 0.05 smaller architectures perform very well, but in general, the value of 0.05 performs worse and usually is not the best choice. However, for practical applications, the best network is needed, so it is reasonable to apply the value of 0.05 with some small architectures which usually give good results. Also, the value 0.05 seems to give the best number of training steps, but this time the difference is not significant compared to 0.15 or 0.2.

Tables 8.11 and 8.12 are important, because for practical applications only the best parameters and networks, giving the best accuracy and training time, are interesting. However, dropout introduces some non-determinism to the training process, so the best results may be slightly different for different runs of training even with the same parameters (e.g., parameters which are the best according to Tables 8.11 and 8.12, turned out not to be the best in some other runs). Also, Tables 8.11 and 8.12 present the best-performing models on the (randomly generated) validation set, so it is possible that the order of the models might be different for another set (e.g., composed of points that are not randomly selected, but are close to local optima found using optimization algorithms). Thus, the average results presented in Tables

Table 8.11: Top 5 results according to MAPE.

Training set size	NN structure	Learning rate	Dropout	MAPE
81920	[100, 200, 100]	0.001	0.05	1.18%
10240	[100, 200, 100]	0.001	0.05	1.24%
81920	[100, 200, 100]	0.01	0.05	1.24%
30720	[100, 200, 100]	0.001	0.05	1.26%
81920	[300, 400, 300]	0.001	0.1	1.3%

Table 8.12: Top 5 results according to MAXAPE.

Training set size	NN structure	Learning rate	Dropout	MAXAPE
10240	[100, 100, 100]	0.001	0.05	6.8%
81920	[100, 200, 100]	0.01	0.05	7.09%
10240	[100, 200, 100]	0.0001	0.05	7.27%
81920	[300, 400, 300]	0.001	0.15	7.33%
10240	[100, 100, 100]	0.01	0.05	7.37%

8.7-8.10 are also important because they show general trends in a larger set of models, and consequently they should also be considered when determining the best hyperparameter settings to use in the training process.

It might seem that the models with the architecture [100, 200, 100], trained on a set of 10240 elements, with a dropout of 0.05 and a learning rate of 0.01, composed of hyperparameter values that seem to be one of the best after averaging over multiple trained models, should give relatively good accuracy, time of training, and time of generating the training set. However, in the conducted experiments, the model with these values of hyperparameters (10240, [100, 200, 100], 0.01, 0.05) gave MAPE of 1.34% and MAXAPE of 10.52%, which did not place it in top 5 in terms of the average and maximum error on the test set. However, dropout introduces some noise and non-determinism, so the models that are best after just a single training do not have to best as good in the next training with exactly the same parameters. Since training a single model takes a significant amount of time on the used machine with a GPU (at least 10 minutes), it is not practical to train each model many times and observe their average performance. However, some promising models were trained a few times and then evaluated on the test set. The configuration (81920, [100, 200, 100], 0.001, 0.05), which gave the best average accuracy in the previous experiment, turned out not to be as good this time, most likely due to non-determinism introduced by dropout (in one case, MAPE increased to 1.7% and MAXAPE to 9.3%). On the other hand, the configuration (10240, [100, 200, 100], 0.01, 0.05) reached

MAPE 1.2% and MAXAPE 6.8% (also, this model has quite a good training time and time required to generate the training set). It does not mean that this model would be best after performing multiple training procedures and averaging the results. Nevertheless, this model was chosen as the surrogate model for the next experiments with genetic algorithms aimed at solving the Traffic Signal Setting problem (Section 8.3.1.5).

8.3.1.5 Applying feed-forward fully connected neural network models to optimize traffic signal settings

In this experiment, one of the best-performing feed-forward fully connected neural networks in experiments described in Section 8.3.1.4 was selected as a surrogate model to evaluate traffic signal settings explored by a genetic algorithm aiming to find good signal settings and solve the Traffic Signal Setting problem.

8.3.1.5.1 Setup of experiments

First, a neural network with the architecture [100, 200, 100] was trained on a training set of size 10240 (a subset of the dataset used in the experiments described in Section 8.3.1.4), with a learning rate 0.01 and dropout 0.05, which, according to Section 8.3.1.4, turned out to be one of the best models for practical applications. It gave MAPE of 1.2% and MAXAPE of 6.8% on the test set (described in Section 8.3.1.4). The training process took approximately 700 seconds on the NVIDIA server with the P100 GPU [254].

Then, a single run of a genetic algorithm was performed with the trained neural network applied to evaluate the quality of traffic signal settings. The representation of offsets of traffic signal settings as chromosomes was the same as in the experiment described in Section 8.3.1.1. The crossover operator was also the same - a uniform crossover. However, the selection and mutation operators of the genetic algorithm were different. As a selection operator, the tournament selection described in Section 6.2.2 was applied. As a mutation operator, the swap mutation described in Section 6.2.2 was applied. The number of iterations and the number of chromosomes in the initial population were also different, i.e., there were 100 iterations with 30 chromosomes (signal settings) in each iteration. Also, this time, only 1 fitness function was investigated - the total time of waiting at red signals with speed $0\frac{km}{h}$. All the other settings had default values based on the implementation of an external Python's library that was used for experiments - *Pyevolve* [278].

In the end, the result of the genetic algorithm run was compared with the best setting from a set of 10^7 randomly generated signal settings evaluated

using the same neural network.

8.3.1.5.2 Results of experiments

The genetic algorithm found a traffic signal setting with a total waiting time of 34926 seconds (it was the setting [118, 118, 50, 39, 88, 25, 115, 109, 118, 118, 50, 117, 0, 103, 25]). It took about 800 seconds because it was done using an external Python library, *Pyevolve* [278], that did not work with the GPU available on the virtual machine at the time of running the experiments, so did not take advantage of it (using GPU, the whole process can be potentially accelerated even more than 1000 times, depending on the number of chromosomes in a single population).

Then, 10^7 settings were randomly selected from the set of 120^{15} possible settings and evaluated using the same neural network in 650 seconds on the same machine, without external libraries, but using P100 GPU (thus, it took advantage of the accelerated inference provided by GPU). The best setting ([115, 116, 26, 43, 92, 40, 115, 75, 103, 112, 51, 112, 14, 58, 8]) gave the total time of waiting (total time with speed $0\frac{km}{h}$) 35718, which is more than 2% worse than for the best setting found after only 3000 evaluations of the fitness function using a genetic algorithm.

8.3.1.5.3 Conclusions from the experiment

This experiment showed that integrating the trained feed-forward fully connected neural networks as surrogate models that evolve the quality of traffic signal settings can significantly accelerate the execution of genetic algorithm experiments and the search for near-optimal traffic signal settings.

It was also shown that the genetic algorithm integrated with a surrogate model can have a better performance than the random search algorithm in terms of the quality of the traffic signal setting it can return and the time required to do so. However, the evaluations were performed using the surrogate model, so it is not certain that the traffic signal settings that are good according to the trained model are also good according to the traffic simulation. It may turn out that the approximation error close to local minima found using a genetic algorithm increases. The next batch of experiments was designed to investigate it.

8.3.1.6 Investigating performance of neural networks and gradient boosting models as surrogate models

The goal of this batch of experiments was to investigate the accuracy of the trained surrogate models applied to traffic optimization as fitness functions

of genetic algorithms, especially in points close to local optima.

This time, in addition to feed-forward fully connected neural networks, gradient boosting models (LightGBM [178]) were also investigated, as some preliminary experiments indicated that such models can also give good results in approximating the results of microscopic traffic simulations.

Gradient boosting models belong to the family of machine learning algorithms. The main idea assumes that weak predictive models become better with subsequent iterations, through learning from mistakes made by previous models. In these experiments, weak learners were represented by decision tree regressors. The most widely used implementations of this algorithm are XGBoost [57] and LightGBM [178]. The LightGBM algorithm is a gradient-boosting framework that uses tree-based learning algorithms. Unlike other tree-based algorithms that use level-wise tree growth, LightGBM grows trees vertically. LightGBM has been shown to outperform XGBoost in terms of memory consumption and computational speed [178], which agrees with the outcomes of experiments. These factors are vitally important in the conducted research, as the execution time of the surrogate model should be as short as possible. On top of that, preliminary experiments indicated slightly better accuracy of LightGBM compared to XGBoost, hence the choice was to base further experiments upon this gradient boosting algorithm.

8.3.1.6.1 Setup of experiments

For this experiment, a new dataset was generated using TSF, this time for a slightly larger region - the whole "Ochota" region with 21 signal groups. The dataset was generated for 105336 traffic signal settings. Each signal setting was represented as a vector of 21 elements corresponding to traffic signal offsets (which are integers from the set $\{0, 1, ..., 119\}$) for one of 21 selected signal groups. For each setting, TSF computed the total time of waiting at red signals in the selected region using a deterministic variant of its microscopic model. For the purpose of training surrogate models, this dataset was divided into a training set (consisting of the first 85336 elements) and a test set (the remaining 20000 elements). This dataset consisting of the evaluated signal settings was made available publicly to enable further research on this topic (cf. Appendix A.2).

LightGBM has many parameters, a core set of hyperparameters was initially selected and their optimal values (presented in Table 8.13) were found through training and validating the model on the training set, with 80%/20% split into training/validation sets. In order to find good values of hyperparameters, the Tree-structured Parzen Estimator (TPE) was used, which is an efficient and robust hyperparameter optimization technique [24]. Compared

to other popular methods like grid search and random search, TPE strives to improve the overall result with every iteration by taking distributions of prior results as a baseline. This fluid approach produces similar results to Random Search, but in a much faster manner.

Table 8.13: Optimal values of core parameters of the LightGBM algorithm found using the Tree-structured Parzen Estimator.

Name of a parameter	**Value**
colsample (fraction of columns randomly sampled for each tree)	0.88
learning_rate (determines the impact of each tree)	0.1
max_depth (maximum depth of a tree)	5
n_estimators (number of decision trees)	2500
subsample (fraction of data randomly sampled for each tree)	0.67
min_data_in_leaf (minimum number of the records in a leaf)	10
lambda_l1 (L1 regularization)	0.1
lambda_l2 (L2 regularization)	0.2

The optimized values of these core hyperparameters were kept and further experiments aimed to optimize some other important parameters: number of leaves, regression technique, and boosting technique. Since LightGBM uses a leaf-wise growth algorithm, the number of leaves (*num_leaves* parameter) controls its complexity. 2 values of this parameter were considered in experiments: 31 and 63. In terms of regression techniques, *Regression_l*1, *Regression_l*2, and *Poisson* evaluated models based on the mean absolute error, root mean squared error, and negative log-likelihood, respectively, as evaluation metrics [207]. In terms of boosting technique, apart from the traditional Gradient Boosting Decision Tree (GBDT) [57] method, Dropouts meet Multiple Additive Regression Trees (DART) [281] method was also included in the experiments. Different values of *num_leaves*, regression, and boosting techniques influence results strongly, so it was decided to test various values of these parameters, train multiple models and select the best 8 models - they are presented in Table 8.14.

In the case of surrogate models based on neural networks, multiple feed-forward fully connected neural network architectures were tested with different layers and numbers of neurons, different activation functions (ReLU [110] and TANH [109]), presence of residual connections (True/False) and values of the weight of the L2 regularization. All models were trained using the RMSProp optimizer [289] for 200 epochs with a batch size of 128 and mean squared error (MSE) as the loss function. In order to normalize the input data and capture its periodic properties, each offset input x was transformed

Table 8.14: 8 best LightGBM models selected for tests.

Boosting	Regression	Num_leaves	MAPE	MAXAPE
GBDT	regression_l2	63	1.78%	10.02%
GBDT	regression_l2	31	1.79%	9.46%
GBDT	poisson	63	1.93%	12.16%
GBDT	poisson	31	1.94%	11.09%
GBDT	regression_l1	63	2.08%	13.27%
GBDT	regression_l1	31	2.09%	12.50%
DART	regression_l2	63	2.28%	12.52%
DART	regression_l2	31	2.29%	12.64%

into a pair $\left(\cos\frac{2\pi x}{120}, \sin\frac{2\pi x}{120}\right)$ which changed the input size from 21 to 42. In total, 216 models were trained with different values of regularization rates, number of layers and neurons, activation functions, etc. Finally, 8 models were selected (from the set of 30 best models with the lowest relative error rates) that had good diversity in terms of hyperparameter values. All models were trained using Keras [59] package with TensorFlow [326] backend. The parameters of the chosen models are presented in Table 8.15.

Table 8.15: Neural network models selected for tests. Architecture $(N,) * L$ refers to a model with L hidden layers with N neurons each. Architecture (N_1, N_2, N_3) refers to a model with 3 hidden layers with N_1, N_2 and N_3 neurons, respectively. *Activation* specifies the activation function in neurons. *L2* specifies the parameter of the L2 regularization. *Residual* indicates if residual connections were applied.

Architecture	Activation	L2	Residual	MAPE	MAXAPE
$(200,) * 3$	TANH	$10e-4$	True	1.77%	8.34%
$(200,) * 3$	TANH	$10e-4$	False	1.78%	9.2%
$(350,) * 2$	TANH	$10e-4$	True	1.8%	10.56%
$(500,) * 5$	ReLU	$10e-3$	True	1.8%	9.5%
$(350, 500, 350)$	ReLU	$10e-4$	False	1.8%	10.17%
$(200, 300, 200)$	ReLU	$10e-3$	False	1.81%	10.95%
$(200, 300, 200)$	TANH	$10e-4$	False	1.81%	9.36%
$(100, 200, 100)$	ReLU	$10e-4$	False	1.84%	10.95%

For conducting experiments using genetic algorithms, a dedicated Python library was developed. Thanks to that, it was possible to test 288 configurations of hyperparameters, all of them are described in Tables 8.16, 8.17, 8.18. In this batch of experiments, the size of the population was constant

- each population contained 120 chromosomes. It was also assumed that for each run there are 100 iterations. 3 selection operators were considered: *rank selection*, *roulette wheel selection* and *tournament selection*, they are described in Section 6.2.2. *Roulette wheel selection* and *rank selection* where combined into a single operator called *combined selection*: there was a parameter *NBest* corresponding to the number of the best chromosomes (ordered according to their fitness function) to be selected deterministically as parents for the next population, and the remaining $(SelectN - NBest)$ chromosomes were selected from the rest of the population with probability proportional to individuals' fitness. *SelectN* was a parameter specifying the number of chromosomes to be selected as parents to generate the next population - it was a parameter of both selection operators: *tournament selection* and *combined selection*.

Tournament selection had also 3 additional parameters that were analyzed in these experiments: *TournamentSize*, *RankingPr* and P_s. *TournamentSize* is the size of a single tournament. *RankingPr* determines the method for the calculation of selection probability in the case of tournaments. If the value is *True*, the selection probability depends on the individual's place in the ranking and is calculated as follows: $(1 - P_s)^{(rank-1)} \times P_s$, where *rank* is an individual's place in the ranking and P_s determines the decrease rate of selection probability calculated proportionally to the place in the ranking. If the value of *RankingPr* is *False*, the selection probability is proportional to the individual's fitness, which corresponds to the *roulette wheel selection*.

Two crossover operators were considered: *single point crossover* and *uniform crossover*, both operators are described in Section 6.2.2. In the case of the *single point crossover* operator, the algorithm randomly selects one point in the parents' chromosomes and swaps the tails. In the case of the *uniform_crossover*, the algorithm randomly selects genes from the corresponding positions in both parent chromosomes.

Two mutation operators were considered: *swap mutation* and *uniform mutation*. The *mut_swap* parameter specified the probability of swapping two adjacent genes in a chromosome (compared to the standard *swap mutation* operator, this time swapping only adjacent genes was considered). The *mut_random* parameter specified the probability of randomly changing the value of the considered gene to another value chosen at random with a uniform distribution from the set of possible values. In total, there were 504 configurations: 432 configurations with the *tournament selection* operator and 72 configurations for the *combined selection* operator.

For each configuration, experiments were run for each of the selected 16 models, 5 times per model (each run started from a random initial population, so different initial settings were tested), giving 2520 runs for each model.

Table 8.16: Hyperparameters of genetic algorithms tested in experiments.

Description of the parameter	**Tested values**
Size of the population	120
Number of iterations	100
SelectN - Number of chromosomes to be selected as parents population	30, 60
Crossover operator	• **Single point crossover** - the algorithm randomly selects one point in the parents' chromosomes and swaps the tails • **Uniform crossover** - the algorithm randomly selects genes from the first or the second parent
mut_swap: Probability of swapping 2 adjacent genes in a chromosome (in swap mutation)	0.01, 0.05, 0.1
mut_random: Probability that the uniform mutation randomly changes the gene's value	0.01, 0.05, 0.1
Determines how the parents' population is chosen	• **Combined selection** - selects $NBest$ best chromosomes and draws $(SelectN - NBest)$ chromosomes with a probability proportional to individuals' fitness • **Tournament selection** - randomly divides the whole population into groups(tournaments) of size $TournamentSize$ and draws one chromosome from each with the probability proportional to individuals' fitness or to the place in the ranking

8.3.1.6.2 Results of experiments

The average value of the fitness function (computed using TSF) in the training set was about 48923, but the genetic algorithms were able to find points with a value at level 32000-33000 according to the surrogate model. However,

Table 8.17: Hyperparameters of genetic algorithms applicable only to the combined selection.

Description of the parameter	Tested values
NBest - Number of the best chromosomes to be selected deterministically	15, 30

Table 8.18: Hyperparameters of genetic algorithms applicable only to the tournament selection.

Description of the parameter	Tested values
TournamentSize - the size of tournaments	2, 3, 4
RankingPr - the method for calculation of selection probability	• **True** - the selection probability depends on the individual's place in the ranking (*rank*) and is calculated as follows: $(1 - P_s)^{(rank-1)} \times P_s$ • **False** - the selection probability is proportional to the individual's fitness (roulette wheel)
Ps - it determines the decrease rate of selection probability calculated proportionally to the place in the ranking (only if *RankingPr* is True)	0.7, 0.85, 1

the goal of this experiment was to investigate the accuracy of approximations using surrogate models near local optima, so for some traffic signal settings produced by a genetic algorithm, the true value of the simulations was calculated using TSF. Each run of the genetic algorithm produced about 12000 settings (120 (number of settings in a population) × 100 (number of iterations)). In some cases, this number was lower because some settings were shared among different populations, but since evaluating signal settings using TSF takes a considerable amount of time, it was important to carefully select which settings to evaluate with TSF.

For each of the 16 surrogate models, 2520 genetic algorithm runs were sorted based on the final best setting (according to the surrogate model) found in a given run. From the 10 best runs and from 10 runs randomly selected from 90 runs on positions 11-100, the best signal settings from each of the 100 iterations were selected for evaluation using TSF. In total, there

were 2000 evaluated settings per model.

Figures 8.3 and 8.4 present distributions of the mean relative error on the test set (composed of randomly selected settings) for surrogate models based on neural networks and LightGBM, respectively. It can be seen that the distributions are similar to the normal distribution and in most cases, the errors are relatively small (below 5%).

On the other hand, Figure 8.5 presents distributions of the mean relative error for settings found using genetic algorithms for all 16 surrogate models. It can be seen that these distributions are not regular and the errors can be high (the errors above 5% are quite common). In most cases, surrogate models underestimate the results of simulations, and usually, the error of approximation is larger than the average error of a given model on a randomly generated test set (which is at the level $1.7\% - 2\%$) and it can reach 12.9%. The same increase can be observed for the maximum error too - it can reach 21.5%. It can be also observed that in the case of neural networks, the activation function may significantly change the distribution of the relative error. For $ReLU$, neural networks almost always underestimate the results of simulations. For $TANH$, the error is more symmetric. This result is also visible in Figure 8.6 which presents relations between approximations and true values returned by TSF for traffic signal settings found using genetic algorithms.

By analyzing errors for traffic signal settings from trajectories of genetic algorithms (Figure 8.7), it can be confirmed that, in general, errors of approximations increase toward underestimating the outcomes of simulations. Furthermore, the errors produced by LightGBM tend to surpass the errors produced by NN, i.e., the simulation results are underestimated more in the case of LightGBM. Another important observation is that despite increasing errors, genetic algorithms are still able to find better settings than in the initial populations.

The principal component analysis (PCA) [271]) was also run on settings from trajectories and it turned out that depending on the surrogate model, genetic algorithms may converge to different regions in the space of traffic signal settings (Figure 8.8).

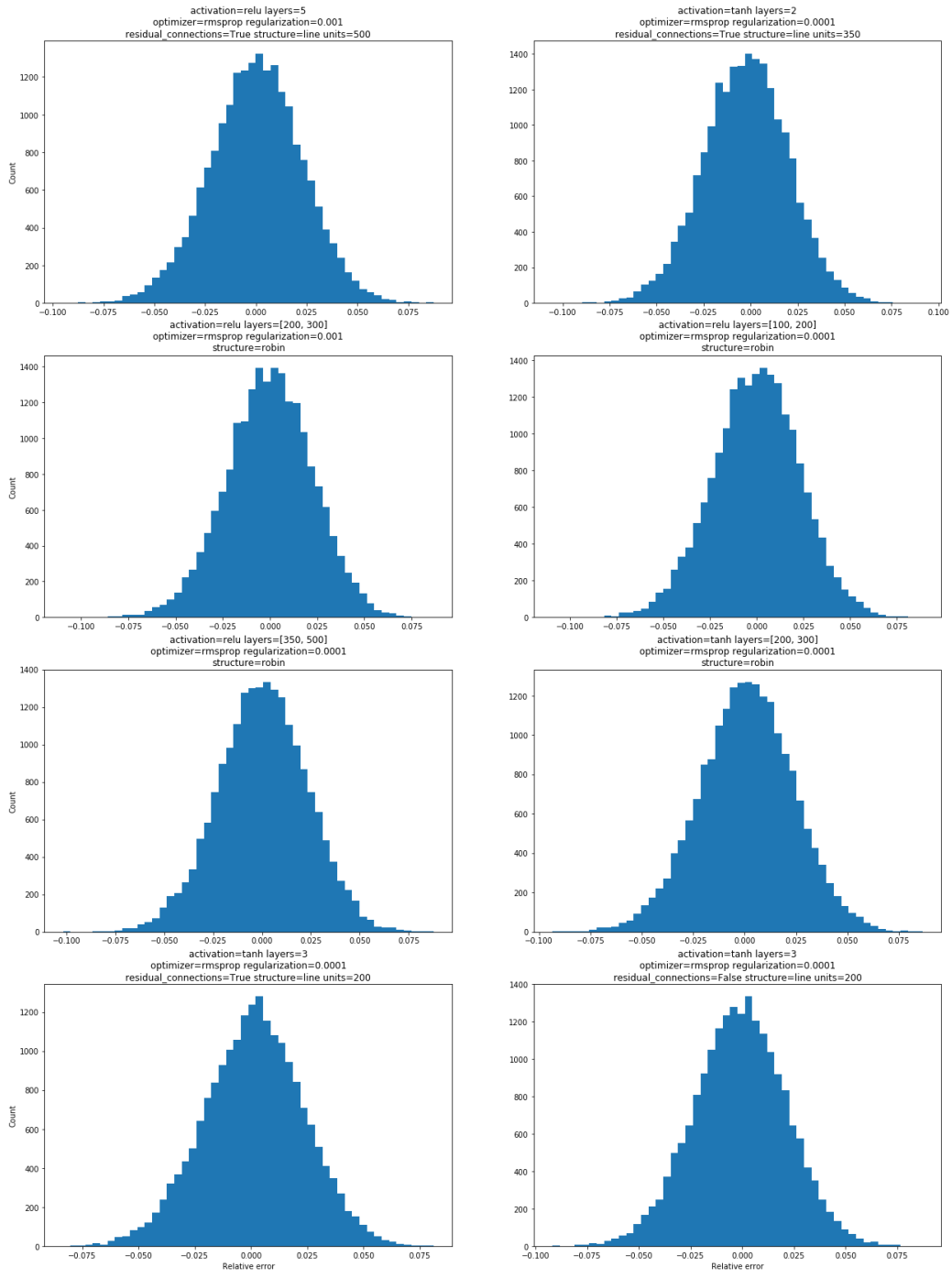


Figure 8.3: Distributions of the mean relative error on the test set for neural networks. The figure is from [130].

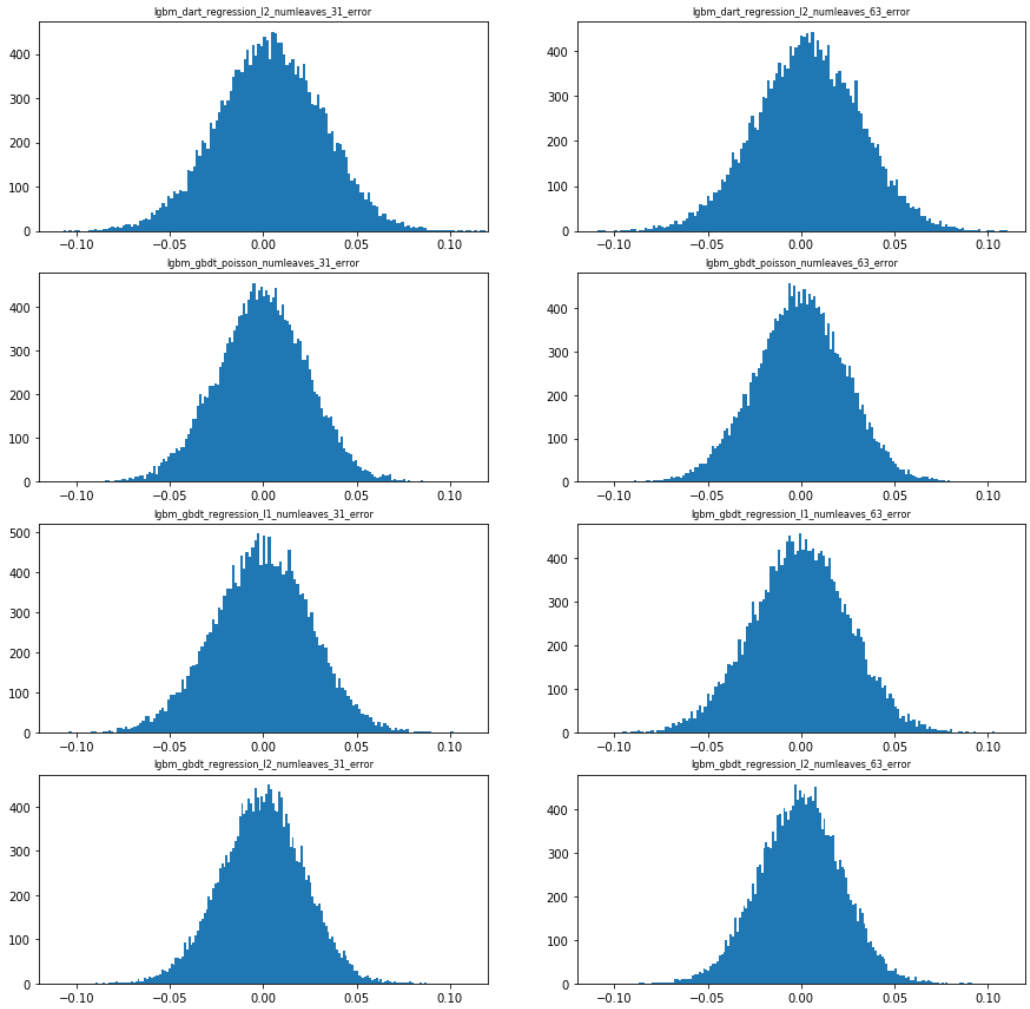


Figure 8.4: Distributions of the mean relative error on the test set for LightGBM models. The figure is from [130].

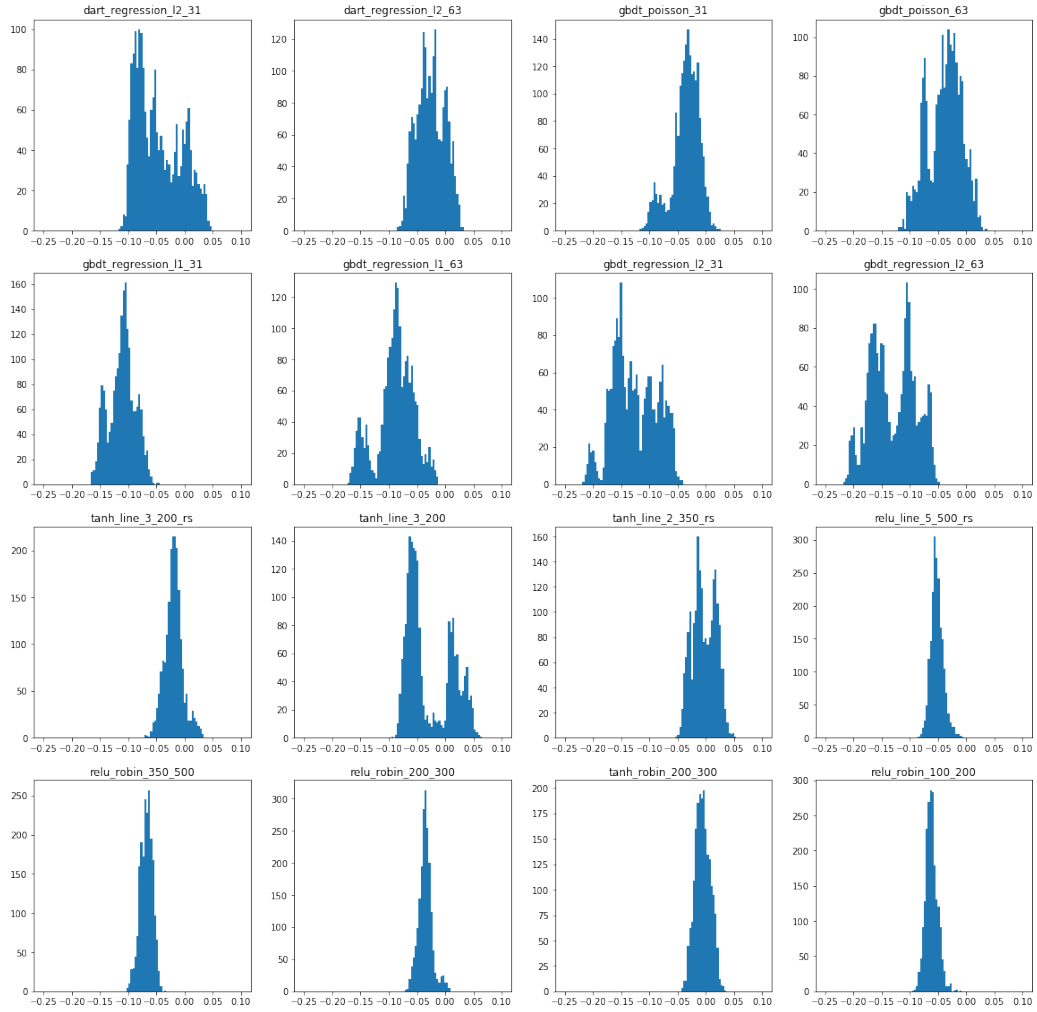


Figure 8.5: Distributions of the mean relative error for settings found using genetic algorithms for all 16 surrogate models. The figure is from [130].

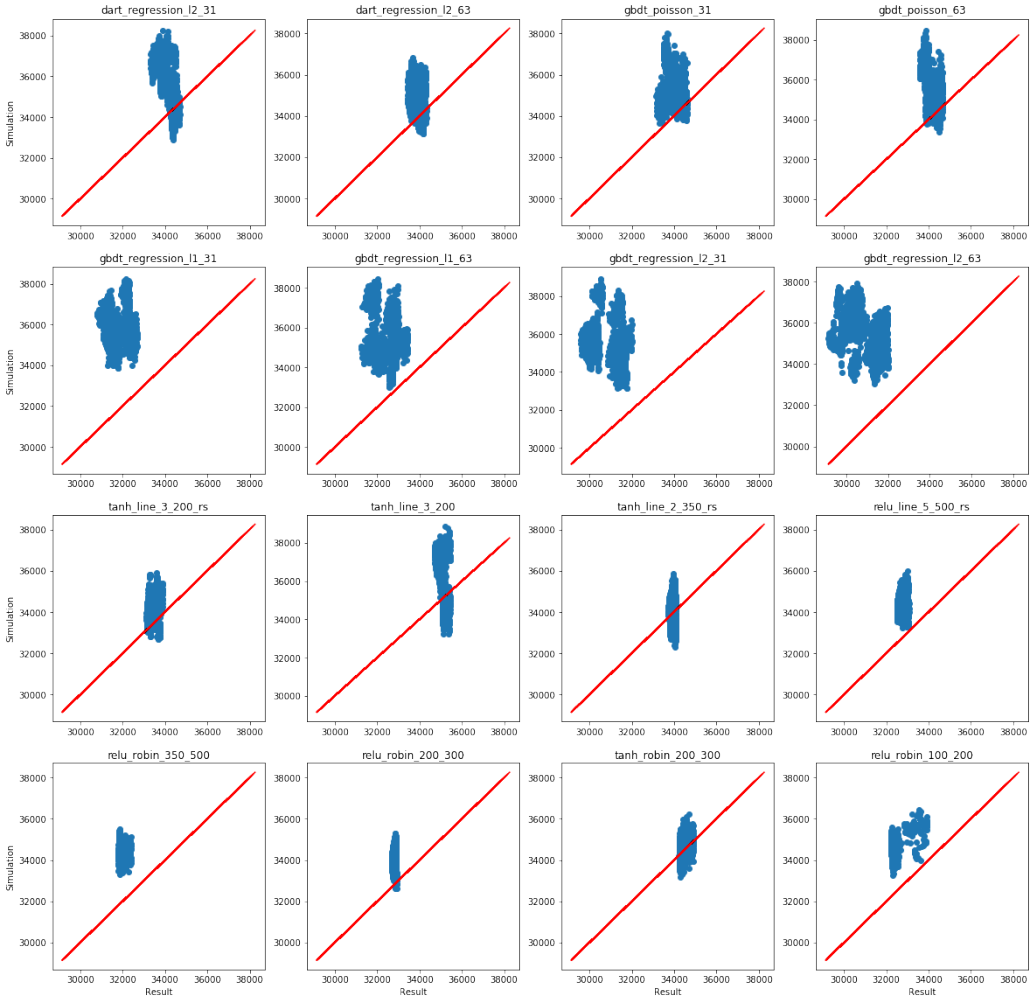


Figure 8.6: Comparison of the results of model predictions and the results of simulations for traffic signal settings with the lowest predictions (100 best settings for 20 genetic algorithm runs) for all 16 surrogate models. The figure is from [130].

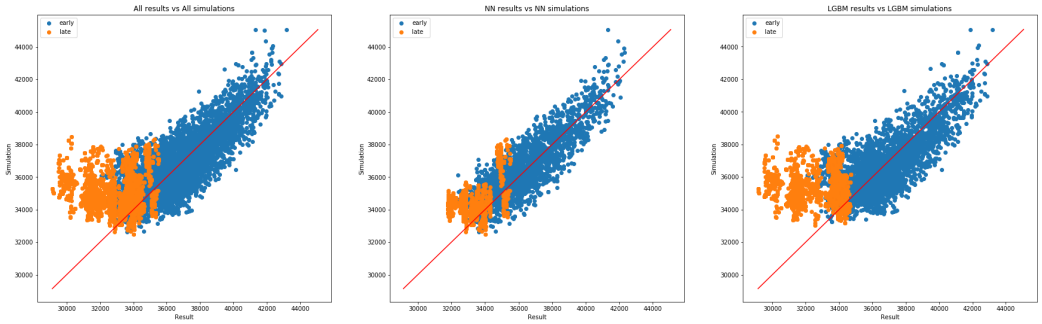


Figure 8.7: Results of approximations (for 3 types of metamodels: LightGBM, neural networks with TANH activation, neural networks with ReLU activation) for best settings from genetic algorithms. The last 20 points from each GA run are marked as orange. The figure is from [130].

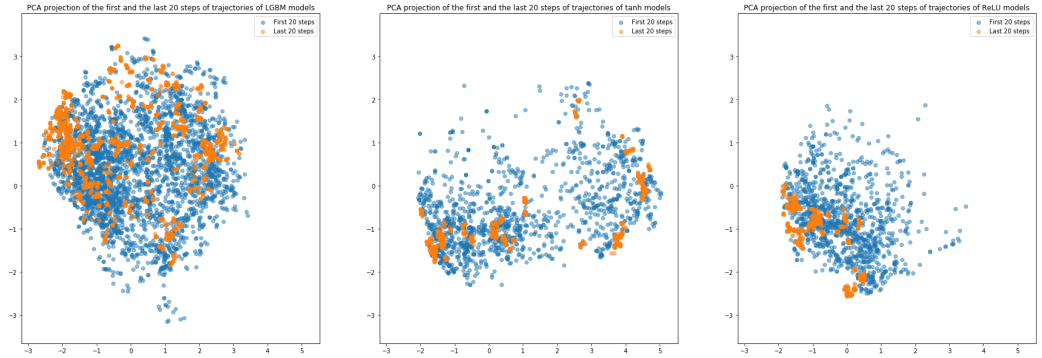


Figure 8.8: Principal component analysis performed on best settings from different populations in GA, for LightGBM models and neural network models with TANH and ReLU activation functions. The figure is from [130].

8.3.1.6.3 Conclusions from experiments

It turned out that MAPE for traffic signal settings close to local optima can be significantly greater than for typical traffic signal settings randomly selected from the large space of possible settings.

For traffic signal settings found by genetic algorithms, errors of approximations increase close to points considered (by a surrogate model) as local minima and the phenomenon is universal - it can be observed in the case of many different surrogate models (neural networks, LightGBM), but its scale

and properties may differ depending on some parameters, e.g., activation function in the case of neural networks.

Two potential reasons for this phenomenon were identified:

- Underrepresentation of points close to local minima (found by genetic algorithms) in the training set. As a consequence, it is difficult for surrogate models to generalize properly.

- Genetic algorithms search for settings with low values according to the surrogate model. Therefore, among the settings where the values according to the surrogate model are low, they find the settings that are farther from the real simulation results (the points that are closer to the real simulation values and, as a consequence, have greater values, have a lower chance of being found, as they might be considered worse by a surrogate model than points for which the surrogate model has a larger error). This could also explain why the surrogate models tend to underestimate the points found by genetic algorithms after many iterations.

Moreover, the principal component analysis showed that genetic algorithms may converge to different areas of the search space, depending on the surrogate model.

One of the interesting research questions at this point was: "Does the phenomenon of increasing error of approximation depend on the optimization algorithm?". The identified potential reasons for this phenomenon suggested that it should not depend on the optimization strategy, but only genetic algorithms had been investigated, so the goal of the next batch of experiments was to check other optimization techniques too, starting from the gradient descent (Section 8.3.1.7).

8.3.1.7 Traffic signal settings optimization using gradient descent

The goal of these experiments was to investigate the performance of gradient descent optimization applied to the Traffic Signal Setting problem with neural networks used as surrogate models evaluating the quality of signal settings. This time the focus was on comparing the performance of the gradient descent method with genetic algorithms in terms of the quality of the found signal settings and the accuracy of approximation at these signal settings.

Gradient descent is an iterative optimization algorithm for finding a local minimum of a differentiable function. The idea is to take repeated steps in the direction opposite to the gradient (or approximate gradient) of the function at the current point, since this is the direction of the steepest descent. In this

case, the differentiable functions to be optimized were the neural networks trained on the dataset introduced in Section 8.3.1.4 (consisting of 85336 traffic signal settings) to be surrogate models approximating microscopic traffic simulations of TSF. They took a traffic signal setting as input and returned the (approximated) total waiting time at red signals in a given region ($Time0$ metric defined in Section 8.2.1). Neural networks are typically trained using the backpropagation algorithm, which calculates the gradient descent over the space of weights of connections between neurons to minimize the (differentiable) loss function that compares the neural network output with the ground truth value (the correct values according to the training set entries). In the case of optimization, the gradient descent can be computed over the input space instead. An apparent problem here is that the traffic signal settings used in the experiments (e.g. the settings in the dataset used) are represented as vectors of integer values from a finite set ($\{0, 1, \ldots, 119\}$), and the ground truth values computed using TSF (the total waiting times at red signals) are also integers. However, the domain can be extended to a continuous domain because the neural networks can take arbitrary values (in practice, the values that can be represented in a given computer program) as input. If the activation functions are continuous and differentiable, the function computed by the whole neural network is also continuous and differentiable. Therefore, it is possible to calculate the gradients of the objective function evaluated by a neural network in the input space and use them to optimize that function.

It is also worth noting that a similar approach is used to find adversarial examples. After training the neural network, the weights are kept constant, but the inputs to the network are modified using gradient descent to maximize the loss function [166].

Also, it is good to emphasize that in the case of LightGBM models such gradient descent optimization is not feasible as the optimized function is not differentiable, so in these experiments, only neural networks were tested.

8.3.1.7.1 Setup of experiments

The dataset and 8 neural network models used in these experiments were the same as in experiments described in Section 8.3.1.6. The results of traffic signal settings optimization using genetic algorithms from those experiments were also used this time, they were just compared with the results of optimization using gradient descent. The optimization processes were run using 4 variants of gradient descent (with appropriately tuned values of hyperparameters):

- stochastic gradient descent (SGD) [35] with a learning rate 0.1;

- Nesterov [249] with a learning rate 0.1 and momentum 0.9;
- RMSProp [289] with a learning rate 1.0;
- Adam [183] with a learning rate 0.5.

For each method and each of 8 metamodels, 1000 gradient descent experiments (each with 100 iterations) were run and 20 best runs (with the lowest predicted values according to the surrogate model) were chosen. This resulted in 160 trajectories per optimization method. Although setting a high learning rate for both Adam and RMSProp is believed to harm the training process in the case of neural networks [183], it turned out that in this case, lower values of the learning rate slowed down the training process and made it stuck at local minima, whereas higher values made the optimization process unstable. In the case of RMSProp - the proposed value of the learning rate helped solve these problems. Unfortunately, even careful fine-tuning of optimizer hyperparameters did not work sufficiently well for Adam.

8.3.1.7.2 Results of experiments

In order to compare the results of different optimization methods (4 based on gradient descent and 1 genetic algorithm), values predicted by surrogate models, values returned by traffic simulations for the same traffic signal settings, as well as errors of approximation for each model were analyzed in points found by the optimization algorithms. For this, the best points from the last 20 iterations of optimization algorithm trajectories were selected for evaluation using TSF. Analysis of the values showed that before this phase (i.e., before the 80th iteration) optimization algorithms had already converged, so these points represented areas close to minima to which a given optimization process converged.

The following measure of discrepancy between the results of optimization methods was introduced:

$$\Delta(A \mid B) = \sum_{X \in res(A)} \min_{Y \in res(B)} \delta(X, Y)/|res(A)|, \tag{8.3}$$

where A and B are different optimization methods, $res(M)$ is the set of all 160 trajectories (which are represented as sets of best traffic signal settings obtained in the last 20 steps of the optimization process, for each of the 8 considered models) generated by method M, and δ is a Euclidean distance between 2 trajectories given by:

$$\delta(X, Y) = \min_{x \in X, y \in Y} ||x - y||_2. \tag{8.4}$$

So, the Δ-discrepancy between methods A and B is given by a mean distance of trajectories (δ) obtained by method A from trajectories obtained by method B. This discrepancy measure is not symmetric, so it was computed for all possible 20 tuples of the used methods and the results are presented in Figure 8.9.

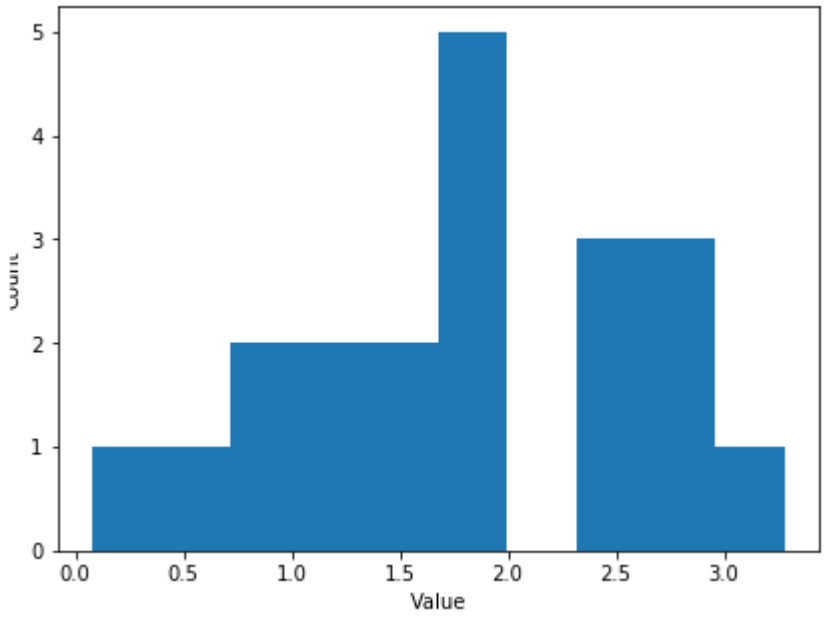


Figure 8.9: Distribution of Δ-discrepancy between all 20 possible tuples of different optimization methods used in experiments (Genetic Algorithm, SGD, Nesterov, RMSProp, and Adam). The figure is from [245].

The values of Δ-discrepancy applied to all pairs of methods varied between 0.08 and 3.28 with a mean value equal to 1.8 and a standard deviation 0.84.

Thus, a hypothesis was formulated that all methods (both gradient-based and genetic algorithms) produce traffic signal settings from roughly the same regions of the input space. In such a case, the values of simulations using TSF, predictions and errors on the traffic signal settings from the generated trajectories should be similar too. The reason is that in the TSF model that was used in experiments, the differences in the total times of waiting should be relatively small for signal settings that are close.

The collected outcomes of simulations evaluating traffic signal settings found by the optimization methods using the deterministic variant of the TSF's microscopic model (Table 8.19) and errors of approximations (Table

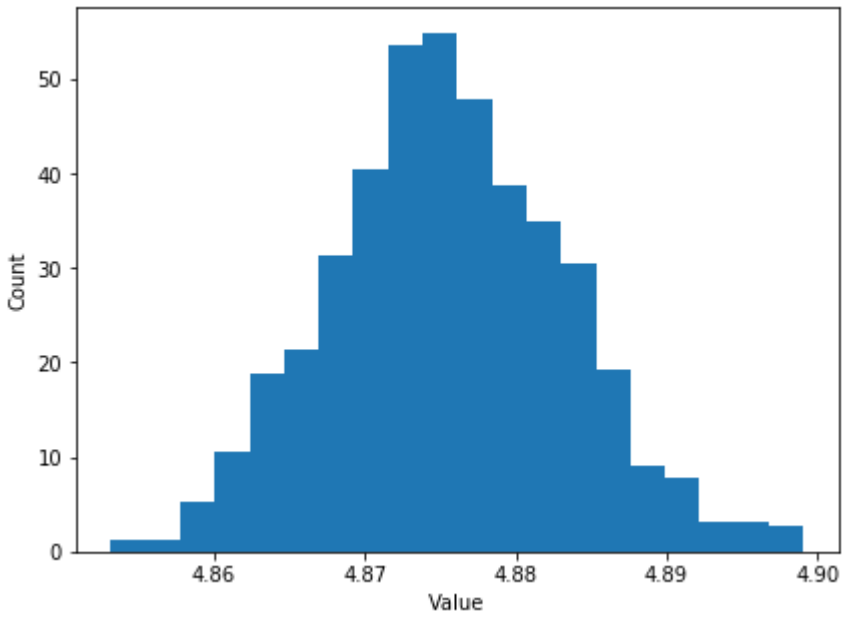


Figure 8.10: Distribution of the δ distance between randomly sampled sets of trajectories ($N = 1000$ pairs were sampled). The figure is from [245].

8.20) confirm that for various optimization methods, the results are similar. Further analysis (presented in [245]) of outcomes for various methods confirmed that the distributions of simulation results and errors of approximations are quite similar.

Table 8.19: Simulation results across different optimization methods.

Metric	**Genetic**	**SGD**	**Adam**	**RMSProp**	**Nesterov**
Min	32620	32565	32406	32383	33026
Max	38320	39143	38062	38015	37602
Mean	34439	34682	34891	34731	34598
Median	34216	34420	34831	34490	34354
Std	1008	1095	817	1060	986

The errors (differences between predictions of models and values from simulations) and absolute errors of approximations of surrogate models for points from trajectories close to local minima are presented in Table 8.20. In terms of the mean and the median absolute error, it seems that the Nesterov method is the most accurate. The second best approach with respect to errors

is the genetic algorithm, which also suffers the least from extreme positive errors. Other gradient methods (RMSProp, Adam, and SGD) have similar mean and standard deviation for both, errors and absolute (ABS) errors.

Table 8.20: Comparison of errors (differences between predictions of models and values from simulations) across different optimization methods. ABS means *absolute error*.

Metric	**Genetic**	**SGD**	**Adam**	**RMSProp**	**Nesterov**
Min	-1573	-1688	-731	-1549	-1826
Max	3326	4041	3657	3834	3814
Mean	1112	1363	1442	1432	905
Median	1206	1388	1435	1600	1010
Std	1033	1066	934	1018	1079
Min ABS	20	1	1	4	1
Max ABS	3326	4041	3657	3834	3814
Mean ABS	1294	1470	1460	1508	1171
Median ABS	1239	1393	1435	1600	1115
Std ABS	792	912	905	899	782

It was also found that for a given surrogate model, gradient descent and genetic algorithms tend to converge to roughly the same regions of input space, which is visualized in Figure 8.11 using PCA dimensionality reduction. One may notice that there are three main regions to which algorithms converge. Two on the left have rather small errors. They also have a small PCA reconstruction error (which refers to the difference between the original data and the approximated data after being transformed and reconstructed using PCA) which suggests that they lie close in the space of signal settings. The region on the right seems to have a much steeper structure of the error values. A greater reconstruction error suggests that this region lies far away from two regions on the center-right and is underrepresented in the trajectory space.

Further analysis showed that the most important factor that made the optimization trajectories different was the activation function of the surrogate model. This is also consistent with the results obtained in previous experiments with genetic algorithms (Section 8.3.1.6). The results of experiments that used models with the same activation (ReLU or TANH) were similar (even if a different optimization method was used) whereas different activations usually resulted in different minima (Figure 8.11). Only models based

on TANH activation explored the less frequent region in the input space. Although the TANH-based models have lower mean and median absolute errors, the possibility of reaching poor local minima is greater which is confirmed by higher maximum value and standard deviation of simulation values (Table 8.21).

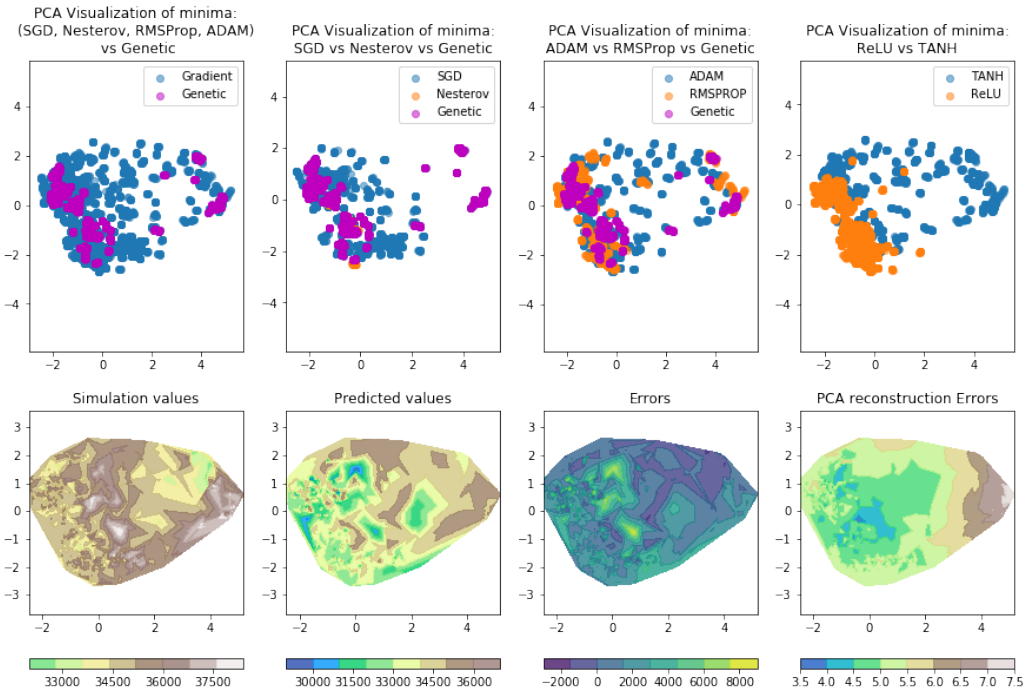


Figure 8.11: PCA visualization of points obtained in the last 20 iterations for different types of optimization algorithms. Most of the points concentrate in roughly the same input region. In regions on the left of the image, the error is smaller, whereas the minima on the right have a much steeper structure of the error values. The PCA reconstruction error suggests that these trajectories might be outliers. The figure is from [245].

8.3.1.7.3 Conclusions from experiments

It was discovered that all considered optimization methods (based on genetic algorithms and gradient descent) produce similar results and shown that the accuracy of neural networks, minima reached and errors, depend on the activation function of the surrogate models based on neural networks. The impact of the optimization technique is not that significant. Gradient

Table 8.21: Simulation results and errors of approximation for ReLU and TANH activation functions.

Metric	ReLU Sim.	TANH Sim.	ReLU Error	TANH Error	ReLU ABS Error	TANH ABS Error
Min Value	32620	32383	-246	-1826	7	1
Max Value	36787	38320	3834	3198	3834	3198
Mean Value	34464	34883	1872	603	1872	866
Median	34416	34526	1887	524	1887	660
Std	546	1237	683	946	682	713

optimization has to check fewer signal settings, so it gives good results a few times faster than GA which makes it a natural candidate for the optimization procedure in future experiments (on the other hand, genetic algorithms have the potential to explore more regions of the traffic signal settings space and can be combined with other surrogate models, e.g., based on LightGBM). However, the errors obtained using different optimization methods and different surrogate models may differ. These results were also a motivation for further experiments in which other optimization algorithms (Section 8.3.1.8) and other surrogate models (Section 8.3.1.9) were applied.

8.3.1.8 Testing various metaheuristics and surrogate models

The goal of these experiments was to test various metaheuristics for solving the Traffic Signal Setting Problem with surrogate models based on neural networks and LightGBM as evaluators of the fitness function.

8.3.1.8.1 Setup of experiments

For this experiment, the same dataset as in experiments described in Section 8.3.1.6 was used and the division into the training and test set was the same. This time, a new neural network model and a new LightGBM model were trained.

The neural network model was a feed-forward fully connected neural network with 4 hidden fully connected layers, with 375, 750, 750, and 375 neurons, respectively. The final output was a linear unit applied to the last of the hidden layers. For all neurons, the activation function was ReLU.

In the training process, the L2 regularization with a weight penalty of 0.0001 was used. Also, dropout [316] with the rate of 0.2 was applied. The

neural network was trained using a loss function which was a linear combination of the mean squared error (with a weight of 0.9) and the mean absolute error (with a weight of 0.1). The network was trained using SGD [35] with Nesterov [249] momentum of 0.8 and a learning rate of 0.2. The reducing learning rate schedule [144] was applied with the learning rate cut by a factor of 0.7 after every 20 epochs without any improvement of training. The training lasted 1000 epochs with a batch size of 128. The input data was scaled by a factor of 120 whereas the output was scaled to have a mean 0 and a unit variance.

After training, this surrogate model had $MAPE = 1.96\%$ and $MAXAPE = 10.95\%$. The average time of a single evaluation was 0.8 ms.

In order to find the best settings of hyperparameters for LightGBM, the Tree-structured Parzen Estimator (TPE) was used [24]. The found values of core hyperparameters are presented in Table 8.22.

Table 8.22: Optimal values of some core parameters of the LightGBM algorithm found using the Tree-structured Parzen Estimator. The meaning of parameters is the same as in Section 8.3.1.6.

Name of a parameter	**Value**
boosting	*GBDT*
regression	*regression L2*
colsample	0.8384
learning_rate	0.0970
max_depth	5
n_estimators	2408
subsample	0.7711
min_data_in_leaf	10
*lambda_l*1	0.96
num_leaves	63
*lambda_l*2	0.2

This LightGBM model with the optimal values of hyperparameters achieved on the test set $MAPE = 1.72\%$ and $MAXAPE = 10.83\%$. The average time of a single evaluation was 0.4 ms.

The trained models were later integrated with several metaheuristics to serve as surrogate models evaluating the quality of traffic signal settings. The following metaheuristics were tested in this batch of experiments:

- genetic algorithms (explained in Section 6.2);
- particle swarm optimization (explained in Section 6.3);

- simulated annealing (explained in Section 6.5);
- tabu search (explained in Section 6.4);
- Bayesian optimization (explained in Section 6.10).

For each metaheuristic, a grid search was performed to find good values for several hyperparameters, exploring all possible combinations of values from the considered sets. For each configuration of hyperparameters, there were 5 runs and the results were averaged to reduce the impact of random initialization and other stochastic effects.

For genetic algorithms, the same hyperparameters and operators were tested as in experiments described in Section 8.3.1.6 (Tables 8.16, 8.17, and 8.18). The only differences were: the number of iterations (300), the sizes of the population (3 values were considered: 40, 120, 160), the values of the $SelectN$ parameter specifying the number of chromosomes to be selected as parents population (6 values were considered: 10, 20, 30, 40, 60, 80), and the values of the $NBest$ parameter specifying the number of the best chromosomes to be selected deterministically (7 values were considered: 0, 5, 10, 15, 20, 30, 40). In total, there were 2538 combinations of hyperparameters for each surrogate model: 648 for the *tournamentselection* and 1980 for the *combinedselection* (note that for the *combinedselection* only feasible combinations of the $NBest$ and $SelectN$ parameters were tested). A grid search technique was run with all combinations of hyperparameter values.

For simulated annealing, the following hyperparameters were tested:

- Initial temperature (IT): Temperature from which the annealing algorithm starts (considered values: 20000, 100000);
- Final temperature (FT): Temperature at which annealing algorithm stops (considered values: 250, 2.5);
- State change (SC): Step of a single (randomly chosen out of 21) signal offset change, by which a given state is decreased/increased after each step of the algorithm (considered values: 10, 5, 1);
- Offset upper limit (UL): Upper bound for each of the 21 signal offsets (considered values: 119, 110);
- Offset lower limit (LL): Lower bound for each of the 21 signal offsets (considered values: 0, 10);
- Underlying model (UM): Model type used to predict the total time of waiting based on 21 signal offsets (considered values: LG for LightGBM and NN for neural nets).

Each run of simulated annealing consisted of 50000 iterations.

For particle swarm optimization, the following hyperparameters were tested:

- n: The number of particles in the swarm, considered values: 5, 10, 20, 30, 40;
- $c1$: Cognitive parameter (following personal best), considered values: 0, 1, 2, 3, 4, 5, 6;
- $c2$: Social parameter (following the swarm's global best position), considered values: 0, 1, 2, 3, 4, 5, 6;
- w: The inertia of the swarm's movement, considered values: 0, 1, 2, 3;
- k: The number of neighbors to be considered (used only for local-best), considered values: 1, 2, 3, 4, 5.

In addition, for the local-best approach, L1 distance (Manhattan distance) was used as the method for computing the distance between particles.

In the case of tabu search, in order to find better results and prevent the algorithm from forming cycles, a Reactive Tabu Search (RTS) [19] was used which is an extended version of tabu search, equipped with a long-term memory structure and an escape procedure. RTS enables finding the points that are often a temporary solution and increase or decrease the short-term memory if specific criteria are satisfied [20]. For RTS, the following hyperparameters were analyzed:

- *Neighborhood structure*: it determines the set of solutions that are considered neighbors of the current solution. Several variants of moves were tested and finally, it was decided to generate the neighborhood of a point x by combining two types of moves:
  - 2-element permutations of TSS elements of x,
  - increasing values of randomly chosen TSS elements of x by 1,

  in such a way that the number of permutations and the number of the increased TSS elements were equal to the size of the neighborhood ($SIZE$). Considered values of $SIZE$ were from the set $\{2, 10, 20\}$;
- *Short-term memory (STM)*: it determines for how many iterations a given element should belong to the Tabu List (is considered as *taboo*). Considered values: $\{1, 5, 10\}$;

- *Aspiration criteria*: it determines the number of times (NR) the element must be added to the Tabu List to apply the escape procedure and consider that element as a candidate solution again. Considered values: $\{5, 20\}$;
- N_{max}: the number of iterations after which the algorithm is stopped. Considered values: 1000 and 10000.

Another metaheuristic that was tested in experiments was Bayesian optimization. It is a well-known technique for the optimization of black-box functions without derivatives [239]. In this case, the function was evaluated using a deterministic variant of TSF's microscopic model (cf. Section 5.3). It took as input traffic settings and computed the total time of waiting at red signals in a given region (the $Time0$ metric).

The optimization strategy in Bayesian optimization is to use some surrogate model which approximates the true function and is cheap to evaluate, then find a minimum of the surrogate model, sample the original function at this point and, based on that, update the surrogate model. As mentioned in Section 6.10.2.1, the usual surrogate used in practice in Bayesian optimization is a Gaussian Process. However, it is slow and requires a lot of memory, while in the considered case, the original function can be easily sampled thousands of times, making this model not useful. Therefore, 3 regressors based on trees were tested instead: Extra Trees, Random Forest, and Gradient Boosted Trees (GBT). They are all described in detail in Section 6.10.2.1.

In addition to the surrogate model, Bayesian optimization also requires an acquisition function to evaluate how likely it is that a true function has a lower value at a given point. It should use the surrogate model and approximate profit from various sample points. In turn, the acquisition function should balance between exploration and exploitation - search both in points with low estimated value and with high variance. As an acquisition function, 3 options were considered (all of them are described in Section 6.10.2.2):

- Probability of Improvement, $PI(x) = Pr(\hat{f}(x) < f^*)$;
- Expected Improvement, $EI(x) = E(max(f^* - \hat{f}(x), 0))$;
- Lower Confidence Bound, $LCB(x) = E\hat{f}(x) - k \cdot Var\hat{f}(x)$;

where $\hat{f}$ is a surrogate model, f^* is a current minimum, k is a parameter that controls the trade-off between exploration and exploitation, Var is the function's variance, while for evaluating probability, a normal distribution was used. The first 10000 points were randomly sampled (using the original

function). Next, for each setting, the most promising point according to the acquisition function was sampled 1000 times. For experiments, the Scikit-Optimize library was used [297]. Since the best result was achieved for Extra Trees and Expected Improvement, 5000 extra points were evaluated with these settings getting the best result.

Although this approach is simple and cheap, it has one disadvantage - even if a good point is found, the probability that the model randomly takes again a point close to it is very small. Consequently, the found minimum does not even have to be a local minimum.

Also, Bayesian optimization does not use surrogate models trained on the datasets generated by traffic simulations, so it is also not a realization of the main methodology presented in this thesis. In addition, it is not generally considered a metaheuristic. However, it was worth studying with other approaches just for comparison.

8.3.1.8.2 Results of experiments

By comparing the results for various settings of genetic algorithms, it was concluded that *Population size*, *Crossover type*, *Swap mutation*, *Uniform mutation*, *Tournament size*, *NBest* and *SelectN*, did not significantly influence the results, but for values of *SelectN* over 20 the results were usually better. Also, it was observed that the *combined selection* operator usually returns better values than *tournament selection* operator.

By comparing the applications of LightGBM and neural networks as surrogate models, the obtained results (the best total times of waiting at red signals in the "Ochota" region according to the surrogate model) were better for LightGBM (according to both, LightGBM and simulations) (cf. Table 8.23) and calculations were two times faster. Also, LightGBM had a relatively small error and a positive correlation with the results of microscopic simulations using TSF when results were higher than around 32000 seconds. On the other hand, the model had also much bigger errors and a negative correlation with simulations when the LightGBM results were lower than 32000 (cf. Table 8.23).

For neural networks, the results according to the surrogate model were much higher than in the case of LightGBM and the errors of approximating the outcomes of microscopic simulations in TSF were lower but were still significant (above 10%). However, the best signal settings were slightly worse according to simulations than in the case of LightGBM.

In addition, after evaluating all the best results of genetic algorithms runs using TSF's microscopic model it turned out that in some cases the found signal settings are really good, and the best results can be found near the

Table 8.23: The best results for genetic algorithms according to the surrogate model. The first 5 rows contain the best results for runs in which LightGBM was used as a fitness function, the next 5 rows contain the best results for runs in which neural networks were used as a fitness function.

Surrogate model (SM) used in GA	**Best result for SM**	**Simulation result for the best result according to SM**
LightGBM	**25318**	37693
LightGBM	25602	36346
LightGBM	25668	36848
LightGBM	25684	36754
LightGBM	25843	34885
Neural networks	**31890**	37179
Neural networks	31899	37448
Neural networks	31906	36953
Neural networks	31907	36461
Neural networks	31908	37351

value of 32000, but such good settings are produced by using LightGBM and not neural networks (cf. Table 8.24). In such cases, there was also a very small error in approximating TSF's simulations using LightGBM (e.g., 31049 by LightGBM and 31735 by simulation). Thus, it was concluded that some local minima of LightGBM can be also quite good settings according to the simulator, so in general, it might be a better surrogate model in this use case.

In the case of simulated annealing, the best 5 results for both surrogate models are presented in Table 8.25. Grid search results show that a small stage change (SC) of 1 unit does not provide good results. Therefore, only $SC \in \{5, 10\}$ appears in the best results. As expected, decreasing the upper bound and increasing the lower bound does not give any advantage to the number of steps considered, since the extreme values for these bounds may be needed to achieve better results. Moreover, both the best predictions and the best simulation results are for traffic settings found by the LightGBM model, which confirms that this surrogate model might be better than neural networks, similar to genetic algorithms.

The results of experiments with tabu search are presented in Table 8.26. In some initial experiments, it was observed that there is no difference between the results obtained for $N_{max} = 1000$ and $N_{max} = 10000$, so in the final experiments $N_{max} = 1000$ was assumed to reduce the time of running computations. Therefore, this parameter is not considered in Table 8.26.

Table 8.24: The best results according to the simulation (using a deterministic variant of the TSF's microscopic model) applied to the best results from all runs of GA. Columns 2 and 3 contain the results of evaluation according to the surrogate models based on LightGBM and a neural network, respectively.

Result for simulation	**Result for LightGBM**	**Result for neural network**
31735	31049	38899
31915	32404	39524
31964	32087	37349
32015	32267	37208
32102	32187	37327

In general, the results of RTS were relatively poor, the algorithm was able to find traffic signal settings better than the average, but they were still worse than the results of other algorithms. At the same time, the error of approximating the outcomes of TSF's microscopic model was still significant.

For particle swarm optimization, the results are presented in Table 8.27. It was observed that LightGBM was generally able to find better settings according to model and simulation, although the best setting according to simulation was found by a neural network model.

In the case of the Bayesian optimization, the results are presented in Table 8.28. For Extra Trees and Random Forest, the best value was achieved for the Expected Improvement (EI) acquisition function (36692 and 37582, respectively). For Gradient Boosting Trees, the best value was for the Lower Confidence Bound (LCB) function (37351).

Table 8.25: 5 best results of experiments with simulated annealing for the 2 considered surrogate models: LightGBM and a neural network. The models were selected based on the best average of 5 evaluations of the best traffic signal settings found using simulated annealing with the given parameters (MIN column). IT is the initial temperature; FT is the final temperature; UL and LL are upper limits and lower limits for the offsets, respectively; SC is the step change; UM is the underlying model (LG = LightGBM, NN = neural network); MIN is the average value of 5 evaluations of the best traffic signal settings found using simulated annealing with the given parameters; SIM is the average value of 5 evaluations using a TSF's microscopic model of best settings found using simulated annealing with the given settings.

IT	**FT**	**UL**	**LL**	**SC**	**UM**	**MIN**	**SIM**
100000	2.5	119	0	5	LG	**32377**	35335
20000	2.5	119	0	5	LG	32671	**34092**
20000	2.5	119	0	10	LG	32990	34864
100000	2.5	119	0	10	LG	33006	35530
100000	250	119	0	10	LG	33136	35694
100000	2.5	119	0	10	NN	**33949**	37362
20000	2.5	119	0	10	NN	34050	37089
100000	250	119	0	5	NN	34076	**35736**
100000	2.5	119	0	5	NN	34143	36950
100000	250	119	0	10	NN	34342	36156

8.3.1.8.3 Conclusions from experiments

Based on these experiments, a comparison of the best results achieved by all studied optimization algorithms was generated and presented in Table 8.29. It contains the best results found by the considered algorithms in all experiments for both studied surrogate models (Neural Network - NN, and LightGBM - LGBM), results of simulations using the microscopic model of TSF for these found settings, as well as the best results according to the simulations after evaluation of all candidate solutions (the best settings) found in all runs of the given algorithm.

It can be concluded that genetic algorithms give the best results in this case: they are able to find the best traffic signal settings within a few seconds. It also turned out that LightGBM produces better traffic signal settings than neural networks and evaluates them about twice as fast. However, one of the drawbacks of using genetic algorithms with a LightGBM evaluation is that the algorithm sometimes ends up at points with very low values returned by LightGBM, while the true values (returned by the simulation) are larger

Table 8.26: Minima found by Reactive Tabu Search depending on hyper-parameters. *SIZE* is the size of the neighborhood; *STM* is the size of the short-term memory; *NR* is the parameter of the aspiration criteria (the number of times (NR) the element must be added to the tabu list to apply the escape procedure and consider that element as a candidate solution again); The next columns contain values found using LightGBM and neural network evaluations as well as corresponding results of evaluations using the deterministic variant of TSF's microscopic model.

SIZE	**STM**	**NR**	**Best LGBM**	**Simulation LGBM**	**Best NN**	**Simulation NN**
2	1	5	42143	48534	43038	45534
2	1	20	42147	49723	43018	48362
2	5	5	41634	49827	42983	54395
2	5	20	41245	53184	42736	49734
2	10	5	41757	48394	42616	48392
2	10	20	41463	49274	42753	48375
10	1	5	40956	48742	41473	47248
10	1	20	41095	48394	41273	48364
10	5	5	**40647**	47384	**40873**	44864
10	5	20	40937	47387	40972	**44747**
10	10	5	42274	50873	41277	49573
10	10	20	42885	50463	41936	48735
20	1	5	41080	50483	41204	46237
20	1	20	41082	49937	41184	46147
20	5	5	41183	45795	40937	48472
20	5	20	40847	**44853**	41038	48264
20	10	5	40834	49374	40937	49264
20	10	20	40842	47382	40956	46247

Table 8.27: The best five settings of the PSO algorithm evaluated using LightGBM model and a neural network model.

Algorithm	PSO variant	n	c1	c2	w	k	Mean score model	Mean score simulation
LGBM	local-best	30	2	1	1	3	**41356**	43989
LGBM	global-best	40	5	6	4	1	41695	41607
LGBM	global-best	30	5	4	5	3	42212	42449
LGBM	global-best	20	5	1	2	1	42826	42512
NN	local-best	30	5	2	1	3	**41938**	44332
NN	global-best	20	5	1	2	1	42317	**41431**
NN	global-best	30	5	6	6	1	42931	43386
NN	global-best	40	5	6	4	1	43222	43833
NN	local-best	30	6	4	1	2	43295	47070

Table 8.28: The best results found using different surrogate models and different acquisition functions in case of Bayesian optimization.

	EI	PoI	LCB
Extra Trees	**36692**	37067	37689
Random Forest	**37583**	37692	37760
Gradient Boosted Trees	38132	37861	**37351**

so that the approximation error close to local optima is too large to make further exploration reasonable. From this perspective, neural networks might be a better option for the next experiments.

Still, one of the problems is the lack of sufficient generalizability of the surrogate models and larger errors near local optima, so the next research works were aimed at solving this problem.

8.3.1.9 Applying graph neural networks as surrogate models

In these experiments, a family of sparsely connected neural networks with connectivity determined by the adjacency matrix of a road network graph was applied to solve the TSS problem. In the rest of the dissertation, they are called *graph neural networks* (or GNN, in short). The presented GNN architectures were initially proposed by one of the members of the TensorCell research group, Łukasz Skowronek, but the entire group's research works and presented experiments were coordinated by the author of this thesis, and their

Table 8.29: Comparison of the best results found by all algorithms.

Algorithm	**Best result [LGBM]**	**Simulation for the best result [LGBM]**	**Best result [NN]**	**Simulation for the best result [NN]**	**Best result according to simulation**
Genetic algorithm	25318	37693	31890	37179	31735
Simulated annealing	31910	33860	32681	35885	33217
Tabu Search	40647	47384	40873	44864	44747
PSO	41356	43989	41938	44332	41431
Bayesian optimization	N/A	N/A	N/A	N/A	36692

summary is also published in [310] and [311].

8.3.1.9.1 Introduction of the graph neural networks

The key idea in defining the sparse graph-based neural network architecture was an intuitively compelling rule that information should propagate locally between the layers of the neural network. The locality in this context means the presence of only those connections between neurons that have a non-zero entry in the adjacency matrix of the corresponding graph (i.e., the signal groups are connected in the road network graph, so that the flow of information in GNN correspond to the flow of vehicles in the road network). In the case of the road network, in order to implement such a rule, the neurons in the successive layers of the neural network should be linked to the neurons corresponding to vertices and/or edges of the underlying road network graph. This leads to the following general ways to build GNN:

1. Neurons in the even-numbered layers, starting from the input layer as layer 0, correspond to graph vertices with traffic signals (in this case - road intersections). Neurons in the odd-numbered layers correspond to graph edges (in this case - road segments, routes between intersections with traffic signals). An exception should be the output layer with just one neuron. Connections from a vertex-localized layer to an edge-localized layer should be present if and only if a given vertex is one of the ends of a given edge in the corresponding road network graph. There are exactly two such connections for every edge neuron. Connections from an edge-localized layer to a vertex-localized layer should

only be present if the edge has the vertex as its end in the corresponding road network graph.

or

2. Neurons in all layers, with the exception of the output layer, correspond to road network graph vertices. Connections from a neuron in one layer to a neuron in the next one should only be present if the corresponding vertices are neighbors in the road network graph. The number of connections for the vertex node is equal to the number of the vertex neighbors. It is also important to note that although the road network is a directed graph, the underlying graph should be its indirect counterpart since both vertices at the end of a road segment can influence each other. However, GNN structures based on a directed graph can also be examined.

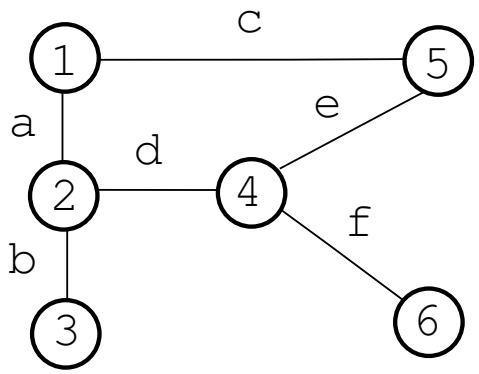
Using formulas, the graph neural network architecture of type 1 uses the following propagation rule:

$$\begin{aligned} L_0 &= In && (8.5)\\ L_{2i+1} &= \phi\left(W_{2i+1}^T L_{2i} + C_{2i+1}\right) && (8.6)\\ L_{2i+2} &= \phi\left(W_{2i+2} L_{2i+1} + B_{2i+2}\right) && (8.7)\\ Out &= V^T L_n + D && (8.8) \end{aligned}$$

where ϕ is an activation function, n is the number of layers, In corresponds to input $N \times 1$ vector, L_{2i} is a vertex-localized $N \times 1$ vector (output from the $2i$-th layer and input to the $2i+1$-th layer) and L_{2i+1} is an edge-localized $M \times 1$ vector (output from the $2i+1$-th layer and input to the $2i+2$-th layer), where N is the number of nodes and M is the number of edges in the considered road network graph. The core elements of the propagation rule are the trainable weight matrices W_j of shape $N \times M$, which fulfill the property:

$$(W_j)_{kl} \neq 0 \Leftrightarrow \text{vertex } k \text{ is one of the two ends of the edge } l. \quad (8.9)$$

Additionally, B_j and C_j are trainable bias vectors, and there is a weight and bias pair (V, D) for the output layer. Figure 8.12 shows an exemplary road network and the corresponding matrix W structure (rows correspond to vertices and columns to edges of the underlying graph).



$$\begin{array}{c} \begin{array}{cccccc} a & b & c & d & e & f \end{array} \\ \begin{pmatrix} w_{1a} & 0 & w_{1c} & 0 & 0 & 0 \\ w_{2a} & w_{2b} & 0 & w_{2d} & 0 & 0 \\ 0 & w_{3b} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & w_{4d} & w_{4e} & w_{4f} \\ 0 & 0 & w_{5c} & 0 & w_{5e} & 0 \\ 0 & 0 & 0 & 0 & 0 & w_{6f} \end{pmatrix} \end{array} \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array}$$

Figure 8.12: An exemplary road network and the corresponding matrix W. Crossroads are denoted by numbers from 1 to 6 and edges by letters from a to f. The figure is from [311].

For the architecture of type 2, the propagation rule is even simpler:

$$\begin{aligned} L_0 &= In && (8.10) \\ L_i &= \phi\left(A_i L_{i-1} + B_i\right) && (8.11) \\ Out &= V^T L_n + D && (8.12) \end{aligned}$$

where all L_i's are now $N \times 1$ vectors and the A_i's are trainable $N \times N$ weight matrices that fulfill the property

$$(A_i)_{kl} \neq 0 \Leftrightarrow \text{vertex } k \text{ is adjacent to vertex } l. \tag{8.13}$$

The meaning of B_i, V and D is the same as in equations 8.5-8.8. Figure 8.13 shows the corresponding matrix A for the graph structure from Figure 8.12.

One can also consider variants in which the neurons corresponding to the same vertices in successive layers are connected. However, later experiments did not show that this approach gives any advantage in terms of accuracy or the time of training.

$$\begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{pmatrix} 0 & a_{12} & 0 & 0 & a_{15} & 0 \\ a_{21} & 0 & a_{23} & a_{24} & 0 & 0 \\ 0 & a_{32} & 0 & 0 & 0 & 0 \\ 0 & a_{42} & 0 & 0 & a_{45} & a_{46} \\ a_{51} & 0 & 0 & a_{54} & 0 & 0 \\ 0 & 0 & 0 & a_{64} & 0 & 0 \end{pmatrix} \end{array} \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array}$$

Figure 8.13: An exemplary road network and the corresponding matrix A. Crossroads are denoted by numbers from 1 to 6.

The presented definition covers only cases with a single feature/channel per edge/vertex. However, GNN can have multiple channels at each edge/vertex neuron and the analogous formulas for an arbitrary number of features can be found in [311].

One may notice a similarity between the GNN architecture of type 2 and the graph neural networks proposed by Thomas Kipf [184]. However, in the case of GNN presented in this thesis, the weights are not shared, as the aim is to focus on local patterns connected to roads/intersections. Theoretically, some weight sharing in the 'edge' layers of GNN of type 1 could be introduced, but initial experiments using this approach led to highly disappointing results.

In typical machine learning literature terminology, the presented GNN should likely be called "neural networks with a fixed sparse connectivity mask". In the case of multi-channel networks, sparsity is applied in the *spacial*, but not in the *channel* dimension.

8.3.1.9.2 Setup of experiments

In the presented experiments, only the architecture of type 1 was used and it was always assumed that the number of channels is constant across the hidden layers of the neural network and is a hyperparameter. However, preliminary experiments with the architecture of type 2 and other settings gave similar results.

The datasets for experiments were generated using the TSF software [125] for 3 regions in Warsaw: "Centrum", "Ochota" and "Mokotów". For "Ochota", the dataset was the same as in the previous experiments (Sections 8.3.1.6 - 8.3.1.8), it had 21 signal groups. For "Centrum" and "Mokotów",

new datasets consisting of about 10^5 traffic signal settings were generated, with 11 and 42 signal groups, respectively. All these datasets are publicly available to enable further research (cf. Appendix A.2).

Each position in the input vector represented the offset of the traffic signal on the corresponding intersection, and for each position in the input vector values from the set $\{0, 1, \ldots, 119\}$ were sampled independently from the uniform distribution. The TSF's output in each case was the total waiting time at red signals, summed for all the cars participating in the simulation in the considered region. Each simulation lasted 10 minutes and consisted of 42000 cars on the whole road network of Warsaw and the evaluation was performed using the deterministic variant of the TSF's microscopic model (described in Section 5.3).

Output values were roughly in the range from 38000 to 60000 for "Ochota", $285000 - 330000$ for "Mokotów", and $67000 - 85000$ for "Centrum". The exact distributions of outputs in the three datasets are shown in Figures 8.14, 8.15, and 8.16. The minimum simulator outputs in the datasets are 37838 for "Ochota", 285483 for "Mokotów", and 66742 for "Centrum". These values are important because the optimization algorithms applied to the TSS problem (such as gradient descent) had the goal to find traffic signal settings (inputs to the simulator) minimizing the TSF's output.

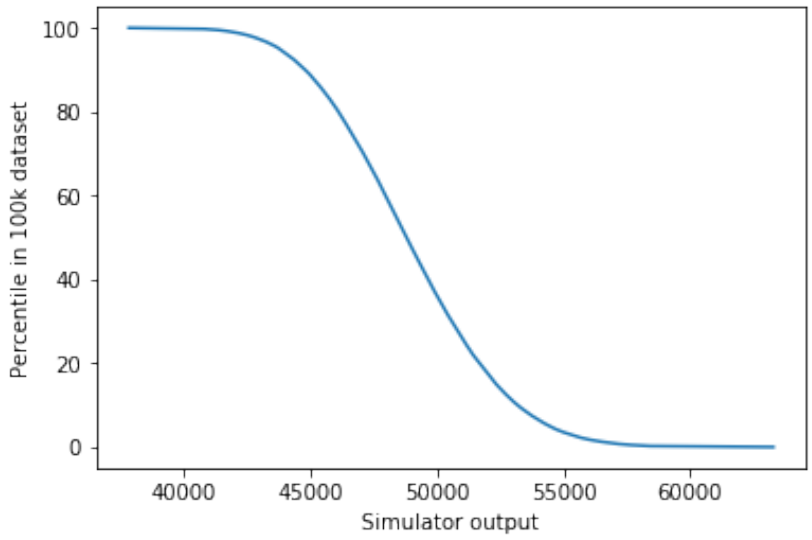


Figure 8.14: The percentage of the TSF's output values greater than the given value for the dataset generated for the "Ochota" region. Volume corresponding to near-optimal parameter settings is extremely low.

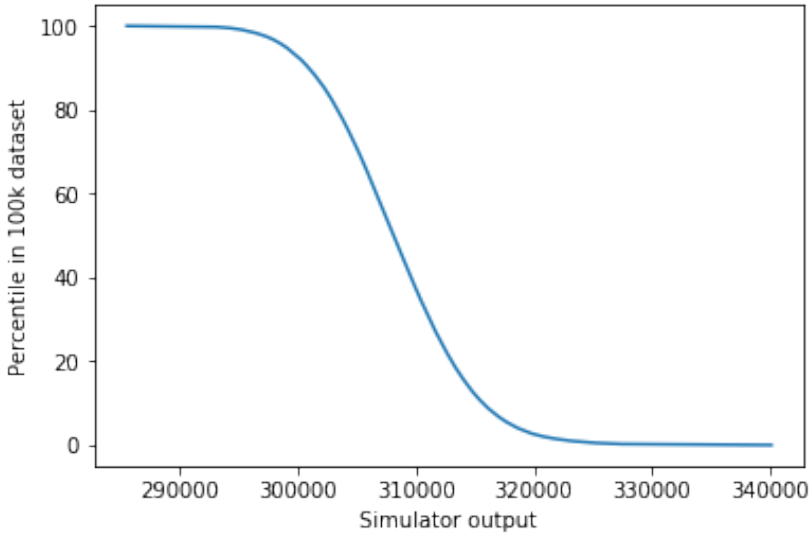


Figure 8.15: The percentage of the TSF's output values greater than the given value for the dataset generated for the "Mokotów" region. Volume corresponding to near-optimal parameter settings is extremely low.

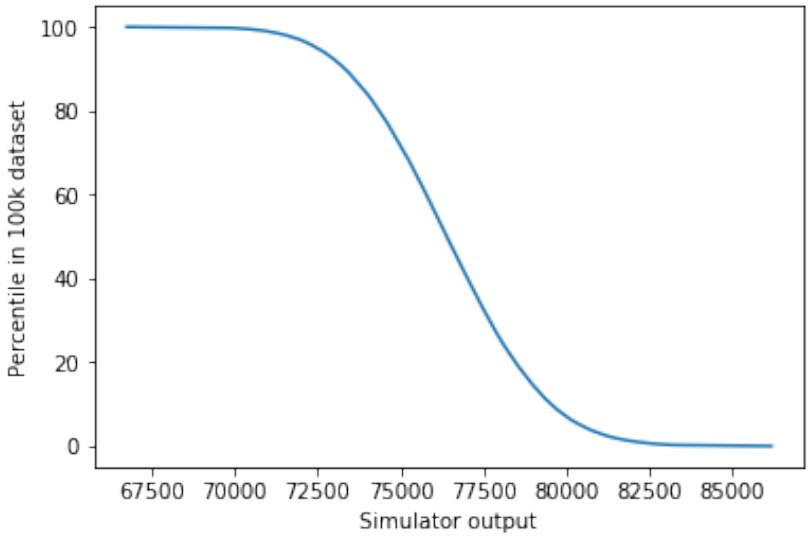


Figure 8.16: The percentage of the TSF's output values greater than the given value for the dataset generated for the "Centrum" region. Volume corresponding to near-optimal parameter settings is extremely low.

Before training, the values at positions in the input vectors were scaled to

$[-1, 1]$. The values in each input vector were transformed by $x \mapsto \sin(2\pi x/120)$ and $x \mapsto \cos(2\pi x/120)$ mapping, thus doubling the input size (actually, increasing its number of channels to 2 in the case of GNN). This is motivated by the periodicity of the signal settings - it turned out that thanks to such a transformation, the neural network may learn that the offsets are periodic and values 0 and 120 correspond to the same setting, while 0 and 119 are neighboring offsets (otherwise, the surrogate model may treat these values as very distant and its performance can be worse). For the output, a standard scaler was used that divides the data by its standard deviation and shifts the mean to zero.

The tensors W_j and A_j (cf. description in Section 8.3.1.9) were initialized using Glorot initialization [109].

For each of the 3 considered road networks, 9 different GNN architectures were tested. In addition, 9 fully-connected neural network (FCNN) architectures were tested for comparison.

The 9 selected GNN architectures corresponded to all the possible combinations of values from the following two sets of hyperparameters:

- number of hidden GNN layers: 2, 3, 4 (not counting input and output layers);
- number of channels per layer: 3, 4, 5.

For comparison, 9 different FCNN architectures were tested, covering all the possible combinations of values from the following two parameter sets:

- number of hidden layers: 2, 3, 4;
- number of neurons per layer: 20, 40, 100.

For both families of neural networks, the activation function was TANH, indicated as superior to ReLU in preliminary experiments, as well as in previous works on this problem [130, 245]. The number of neurons per layer, as well as other hyperparameters (like learning rate or batch size), were chosen based on the results of preliminary experiments.

For each of the 3 datasets, the 90%/10% split into the training set, and the test set was used for each of the considered 18 hyperparameter settings (9 GNN architectures and 9 FCNN architectures). The following procedure was applied for each of the considered architectures:

1. Train a model on the training set for about 1100 epochs using Adam optimizer [183] and a learning rate of 0.0035.

2. Evaluate the trained model on the test set using MAPE with respect to the original outputs (before normalization and scaling) as the core metric.

3. Generate 100 gradient descent trajectories (for 3000 steps) of the trained model output with respect to its inputs in the original input space (before the sin/cos transformation). Gradients were evaluated at inputs rounded in the original parameter space (the applied traffic simulator (TSF) accepts only integer values of offsets). The Nesterov variant of gradient descent [249] with a learning rate of 0.01 and momentum of 0.9 was used. This is similar to the approach used in Section 8.3.1.7 and in [245].

4. Every 30 step, transform the current trajectory point to the original parameter space, round it, and send it to the TSF simulator. Save the inputs and the simulator outputs to a new *simulation test set*.

5. Evaluate the trained model on the *simulation test set* using various metrics (cf. the discussion in Section 8.3.1.9.3).

An important note is that the "test set" did not play the typical role of a test set in machine learning experiments. Instead, it was used only to define an order on the considered GNN and FCNN architectures, for use in tables 8.30-8.34 and plots 8.17-8.19. The actual *testing* was performed using the *simulation test set* (cf. point 5 of the procedure described earlier). These datasets were different for each model, as gradient descent trajectories were also different.

All the experiments were run on virtual machines in the Azure cloud (NC6 with NVIDIA Tesla K80 [227]).

The core dataset has been made publicly available to enable further research in this domain (cf. Appendix A.2).

8.3.1.9.3 Results of experiments

The key results of experiments with GNN are shown in Tables 8.30–8.34 , as well as in Figures 8.17–8.19.

Table 8.30: Core results for the three best GNN and the three best FCNN architectures according to the accuracy (MAPE) on the test set (i.e., gradient descent results did not affect the selection of these models).

Measure	**Model**	**Ochota**	**Mokotów**	**Centrum**
Minimum MAPE on the test set	GNN	1.33%	0.76%	0.80%
	FCNN	1.71%	0.94%	0.87%
Minimum simulation output	GNN	32,205	265,129	63,606
	FCNN	32,587	266,237	63,553
Avg MAPE on the lowest 5% simulation outputs	GNN	1.26%	0.53%	0.76%
	FCNN	5.35%	3.04%	2.49%
Avg MAPE on the lowest 10% simulation outputs	GNN	1.75%	0.84%	1.22%
	FCNN	4.53%	2.74%	2.25%
Avg MAPE on the lowest 15% simulation outputs	GNN	1.51%	1.00%	1.11%
	FCNN	4.65%	2.56%	2.04%

Table 8.30 shows a summary of core performance measures, calculated for three top GNN and three top FCNN, ranked based on the average accuracy (Mean Absolute Percentage Error - MAPE) on the test set. The core presented measures are:

- **Minimum MAPE on the test set**: This number can be obtained before performing gradient descent. The minimum was taken among the three top-ranked GNN or FCNN (according to the row description). Because of the model selection criterion used for Table 8.30, this minimum is global within the respective 9-element model universe (GNN or FCNN).
- **Minimum simulation output** obtained on selected points explored by gradient descent.
- **Average MAPE** at $x\%$ (for $x = 5, 10, 15$) gradient descent trajectory ends, selected according to the corresponding simulator output value (sorted lowest first). To arrive at the values presented in rows 5-10 of Table 8.30, an average was taken over the three models selected, GNN or FCNN, according to the row description.

The results presented in Table 8.30 show a better performance of GNN compared to FCNN, particularly in terms of minimum MAPE on the test set and average MAPE on the lowest points from the gradient descent trajectory according to the simulation. The improvement is visible for all 3 road

networks (from "Ochota", "Mokotów", "Centrum" regions) and all the five core measures (with the exception of the minimum simulation output value obtained for "Centrum", where one FCNN turned out to yield slightly lower result than all the GNN).

To summarize, the core improvements are:

- Much lower error on the test set.
- Lower minimum simulator output value obtained when doing the gradient descent (except for "Centrum", for which the results are similar).
- Much lower approximation error obtained on the trajectory ends corresponding to 5%, 10%, and 15% lowest simulator output values.

For performance measure calculations in Table 8.30, 3 GNN and the 3 FCNN that yielded the best test set performance were selected for each of the three considered road networks ("Centrum", "Ochota", "Mokotów"). It is natural to ask if the conclusions are robust against choosing a different number of the best-performing models. The demonstration of robustness is included in Tables 8.31-8.34. These tables were prepared using one, two, five, and nine (i.e., all) best-performing GNN and FCNN according to the test set performance. The minimum average test set error is not shown, because, by construction, it would be the same in all these tables and in Table 8.30. It is easily seen that the conclusions are upheld.

Table 8.31: Core results for the best GNN and the best FCNN according to test set accuracy.

Measure	**Model**	**Ochota**	**Mokotów**	**Centrum**
Minimal simulation output	GNN	32,299	265,129	63,606
	FCNN	33,098	266,237	63,941
Avg MAPE on the lowest 5% simulation outputs	GNN	0.51%	0.41%	1.69%
	FCNN	1.90%	3.18%	2.07%
Avg MAPE on the lowest 10% simulation outputs	GNN	0.71%	0.66%	1.33%
	FCNN	1.75%	3.20%	1.72%
Avg MAPE on the lowest 15% simulation outputs	GNN	0.87%	0.76%	1.29%
	FCNN	1.78%	2.93%	1.77%

Table 8.32: Core results for two best GNN and two best FCNN according to test set accuracy.

Measure	**Model**	**Ochota**	**Mokotów**	**Centrum**
Minimum simulation output	GNN	32,288	265,129	63,606
	FCNN	32,587	266,237	63,553
Avg MAPE on the lowest 5% simulation outputs	GNN	1.85%	1.10%	0.29%
	FCNN	2.28%	5.05%	1.52%
Avg MAPE on the lowest 10% simulation outputs	GNN	1.42%	1.02%	0.29%
	FCNN	2.15%	4.19%	1.63%
Avg MAPE on the lowest 15% simulation outputs	GNN	1.33%	1.13%	0.36%
	FCNN	2.03%	3.88%	1.53%

Table 8.33: Core results for five best GNN and five best FCNN according to test set accuracy.

Measure	**Model**	**Ochota**	**Mokotów**	**Centrum**
Minimum simulation output	GNN	32,205	265,129	63,606
	FCNN	32,587	266,237	63,553
Avg MAPE on the lowest 5% simulation outputs	GNN	0.99%	0.35%	0.58%
	FCNN	8.59%	1.89%	1.92%
Avg MAPE on the lowest 10% simulation outputs	GNN	0.98%	0.44%	0.88%
	FCNN	8.59%	1.30%	1.92%
Avg MAPE on the lowest 15% simulation outputs	GNN	0.92%	0.37%	1.01%
	FCNN	8.54%	1.01%	1.92%

Table 8.34: Core results for all the considered GNN and FCNN.

Measure	**Model**	**Ochota**	**Mokotów**	**Centrum**
Minimum simulation output	GNN	32,205	265,129	63,606
	FCNN	32,587	266,237	63,553
Avg MAPE on the lowest 5% simulation outputs	GNN	0.87%	1.12%	1.39%
	FCNN	4.63%	2.27%	1.60%
Avg MAPE on the lowest 10% simulation outputs	GNN	0.87%	1.73%	0.88%
	FCNN	4.44%	2.71%	1.79%
Avg MAPE on the lowest 15% simulation outputs	GNN	0.84%	1.87%	0.88%
	FCNN	4.30%	3.03%	1.39%

Figures 8.17–8.19 show the density of gradient descent trajectory points as heatmap plots for 3 best GNN/FCNN models for each district.

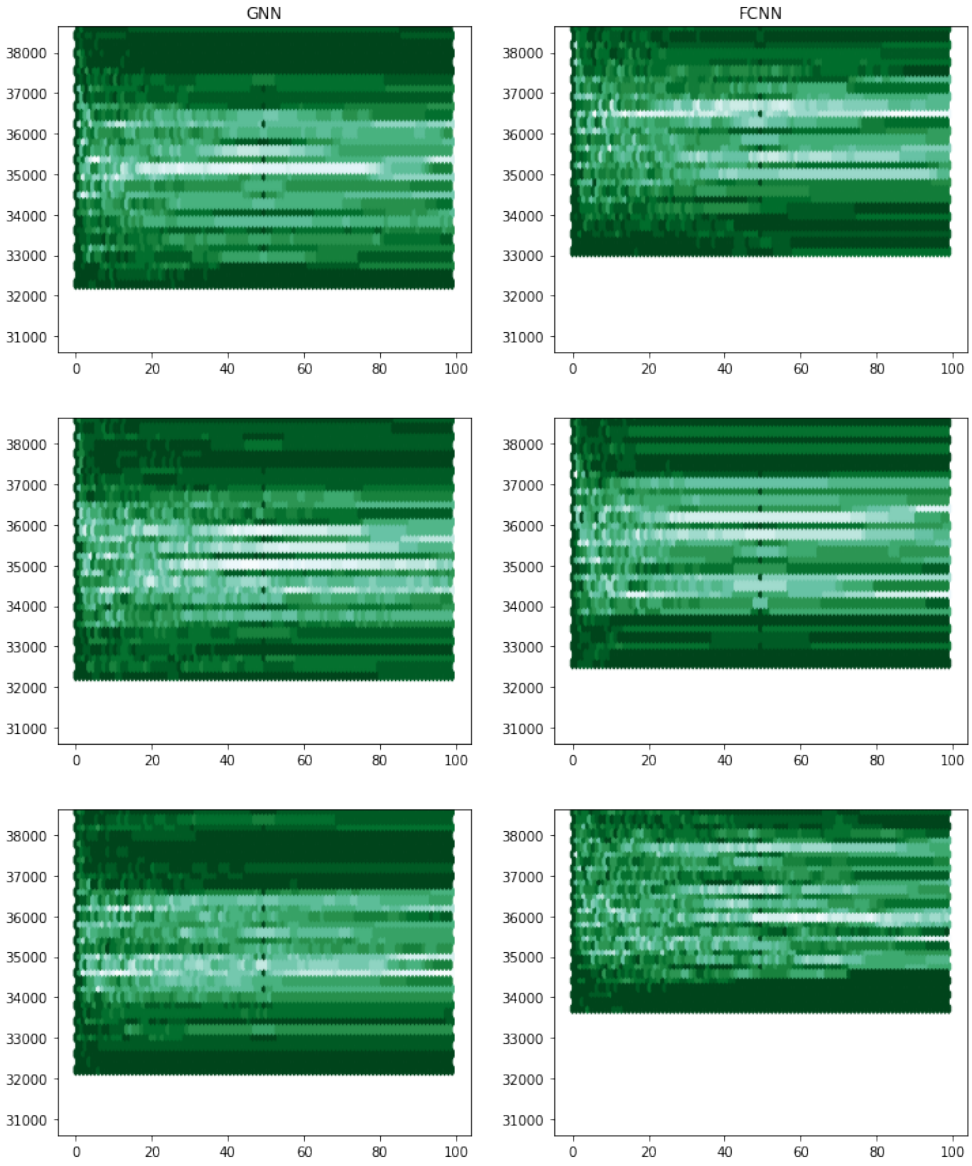


Figure 8.17: Gradient descent trajectory density plots for "Ochota" for the 3 best GNN and FCNN models (charts on the left correspond to GNN, charts on the right to FCNN). The horizontal axis corresponds to the trajectory point number (recorded every 30 steps), vertical axis to the simulator output value. The more points in some regions, the brighter the color. The figure is from [311].

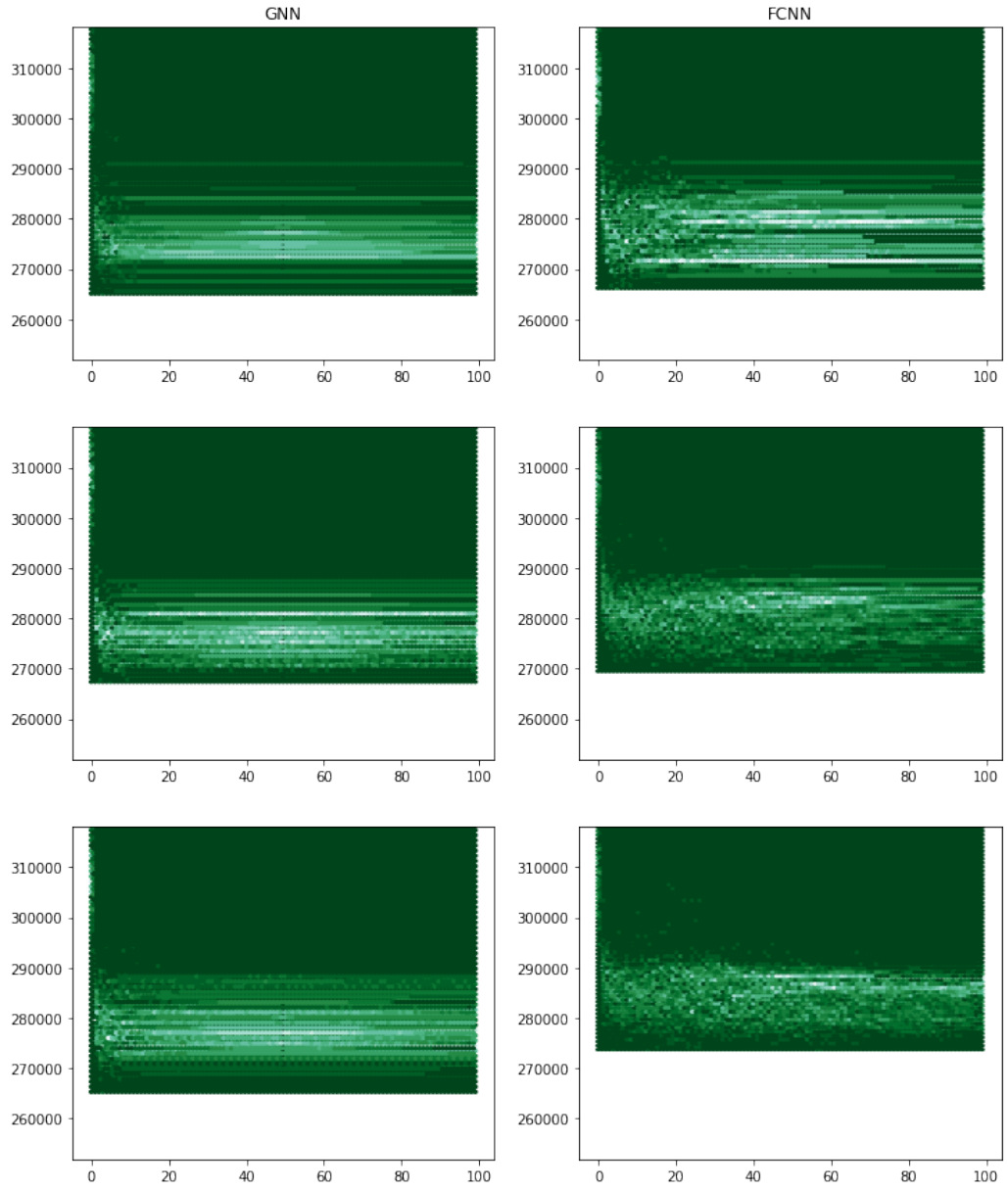


Figure 8.18: Gradient descent trajectory density plots for "Mokotów" for the 3 best GNN and FCNN models (charts on the left correspond to GNN, charts on the right to FCNN). The horizontal axis corresponds to the trajectory point number (recorded every 30 steps), vertical axis to the simulator output value. The more points in some regions, the brighter the color. The figure is from [311].

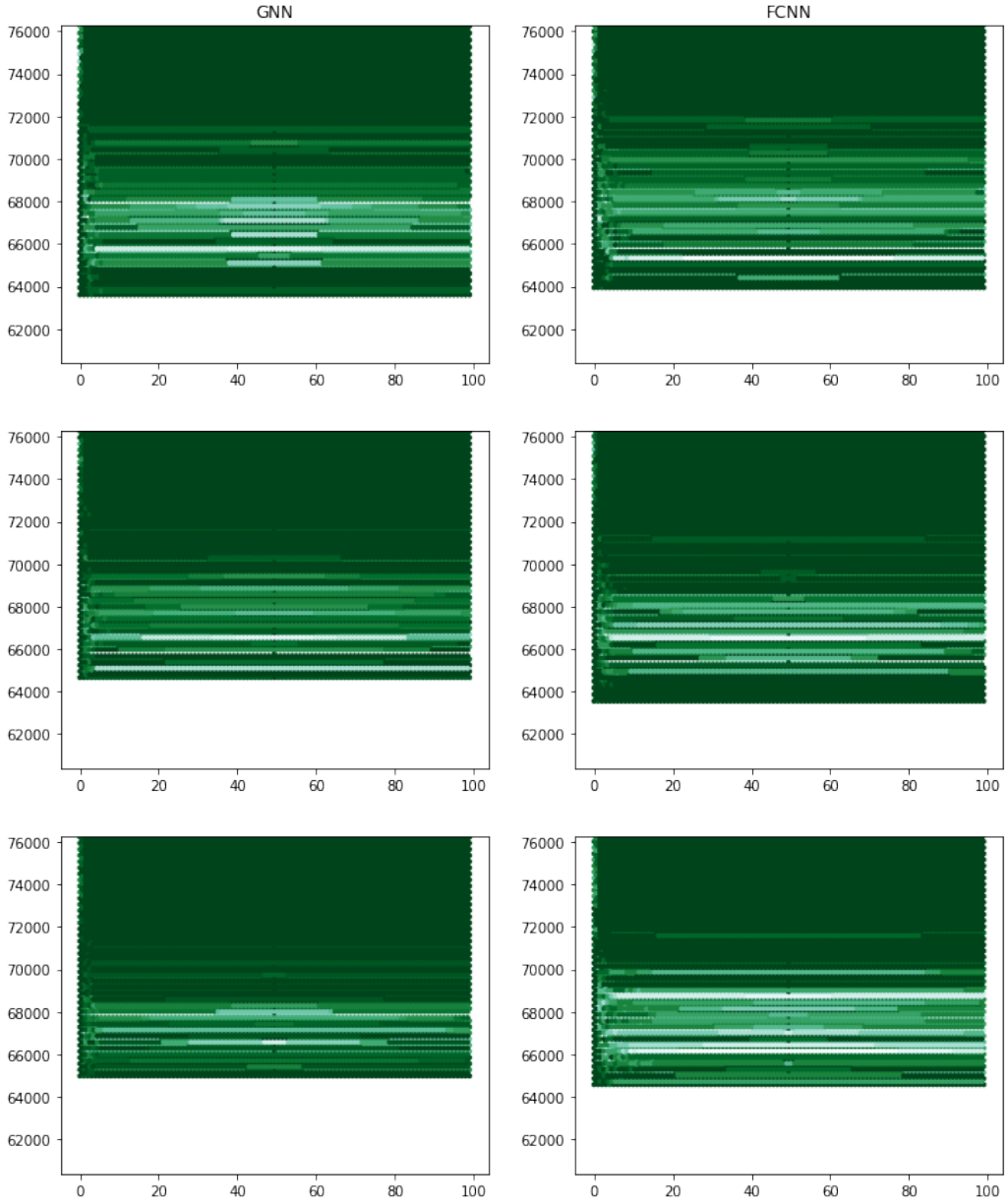


Figure 8.19: Gradient descent trajectory density plots for "Centrum" for the 3 best GNN and FCNN models (charts on the left correspond to GNN, charts on the right to FCNN). The horizontal axis corresponds to the trajectory point number (recorded every 30 steps), vertical axis to the simulator output value. The more points in some regions, the brighter the color. The figure is from [311].

The horizontal axes correspond to gradient descent trajectory point numbers (each trajectory had 3000 steps, but points were recorded every 30 steps), and the vertical axes correspond to respective simulator output values. The plots simply show a heatmap plot of these points on the (point number, simulator output) plane. Thus, the more points in the given region, the brighter the color. Also, if one architecture happened to reach a lower minimum than another, the resulting heatmap is taller.

Besides confirming some of the quantitative conclusions from Tables 8.30-8.34, the heatmap plots also show that in many cases, the gradient descent is less "noisy" for GNN, suggesting a smoother function surface, less prone to overfit noise.

8.3.1.9.4 Consistency checks

The presented findings call for some careful consistency checks before reaching final conclusions. In particular, it is not fully clear that the actual adjacency matrix brings any value. Perhaps, using any similar sparse graph, even not related to the problem at hand, could result in similar performance.

Therefore, it was decided to run some additional experiments where the original graph topology was perturbed (either by turning some edges to non-edges and vice versa, or by permuting graph vertex labels). The other steps of the model fitting procedure were kept exactly the same as before. The achieved results indicate that the mean squared error obtained on the (fixed) test set grows approximately monotonically as a function of the distance (symmetric difference between edge sets) of the graph used (to build the neural network) to the actual instance of the road network graph for which the dataset was generated (cf. Figure 8.20-8.25). This implies that the information about the actual graph topology was important for achieving better accuracy in the main batch of experiments.

In detail, the procedure was performed as follows. First, it was decided to fix the number of layers to 3 and the number of channels to 4 per layer (for GNN of type 1). Then, the neural networks were built using random graphs with various degrees of resemblance to the original problem graph. This process was repeated for all three road networks considered in this batch of experiments. As a measure of graph similarity, the symmetric difference between the sets of graph edges was used. The random graphs were generated in two ways. The first method (later referred to as "Edge/Non-edge switching") used random edge insertions and deletions, keeping the desired value of the symmetric difference fixed. The second method (later called "Vertex label permutation") used random permutations of vertex labels while keeping the connection graph structure exactly the same. Graphs generated by

this method were isomorphic, but not identical to the original.

It is worth noting that although the first method generates truly random graphs similar to the original, the new graph may not represent a plausible road network. The second method, on the other hand, always maintains the same realistic road network graph structure but provides false insights to the training algorithm by swapping intersections.

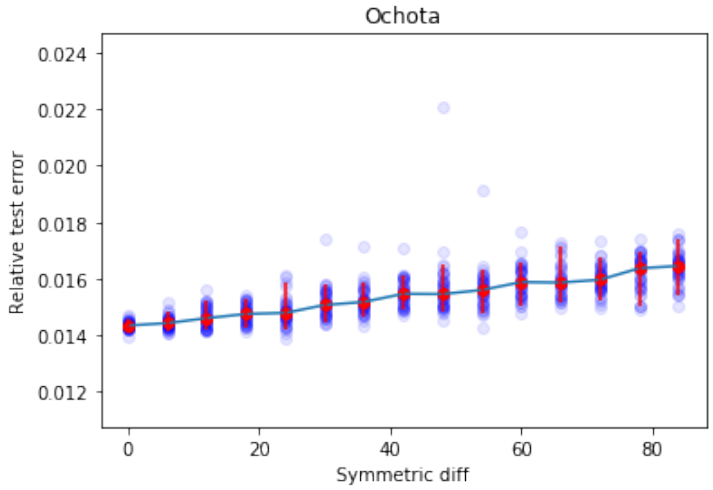


Figure 8.20: Mean relative error achieved by a GNN on the test set for "Ochota" after roughly 330 epochs of training, plotted as a function of the distance of a random graph to the true one. Edge/non-edge switching was used to generate random graphs. Error bars correspond to 5% sample quantiles. Red dots denote the median result. The figure is from [311].

The results obtained by these two methods are shown in Figures 8.20-8.25, corresponding to "Ochota", "Mokotów", and "Centrum" datasets, respectively. The plots show the mean relative error achieved on the test sets by neural nets based on random graphs, after roughly 330 epochs of training (in all the initial experiments the training and test errors after $100 - 200$ epochs had already been stable, so the number of epochs was chosen with some security margin). The values on the horizontal axis correspond to the distance of the graph used for constructing the network to the actual road network graph. The distance was measured using the symmetric difference between the two graphs' edge sets. For each training run, the same train/test split of the respective dataset was used: 90%/10%. However, the weights of neural networks were initialized randomly. It can be seen that the median of the mean relative error, denoted with a red dot, grows almost monotonically as a function of the distance between the graph used to build a GNN and

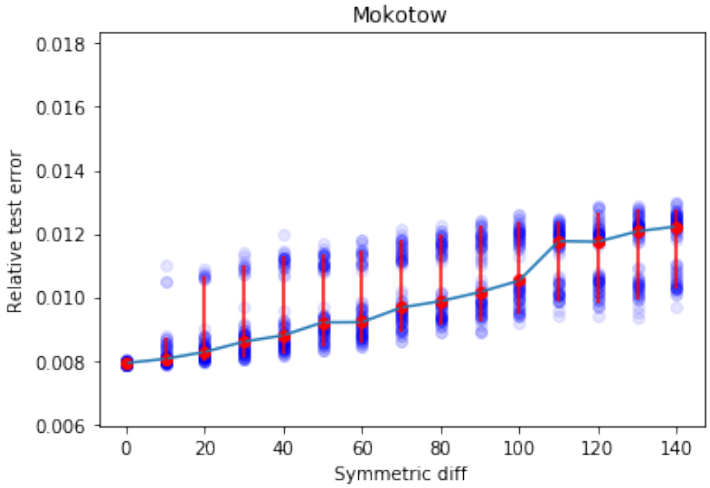


Figure 8.21: Mean relative error achieved by a GNN on the test set for "Mokotów" after roughly 330 epochs of training, plotted as a function of the distance of a random graph to the true one. Edge/non-edge switching was used to generate random graphs. Error bars correspond to 5% sample quantiles. Red dots denote the median result. The figure is from [311].

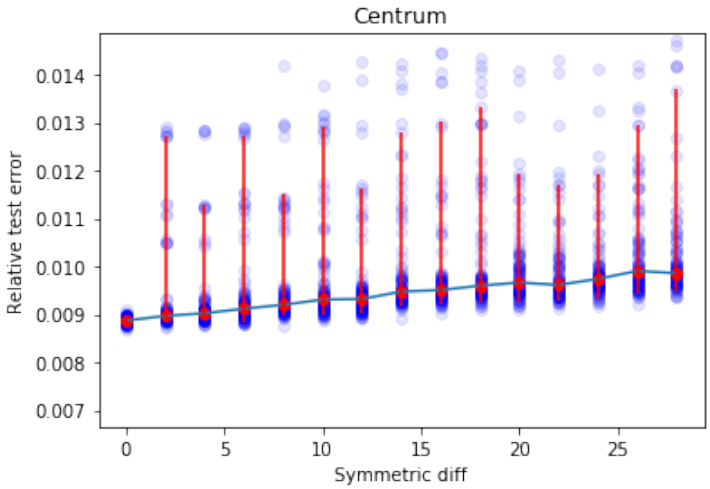


Figure 8.22: Mean relative error achieved by a GNN on the test set for "Centrum" after roughly 330 epochs of training, plotted as a function of the distance of a random graph to the true one. Edge/non-edge switching was used to generate random graphs. Error bars correspond to 5% sample quantiles. Red dots denote the median result. The figure is from [311].

the actual problem graph. This is visible for both graph sampling methods. The minimum average relative error attained for a particular value of the distance also grows, perhaps with slightly more noise.

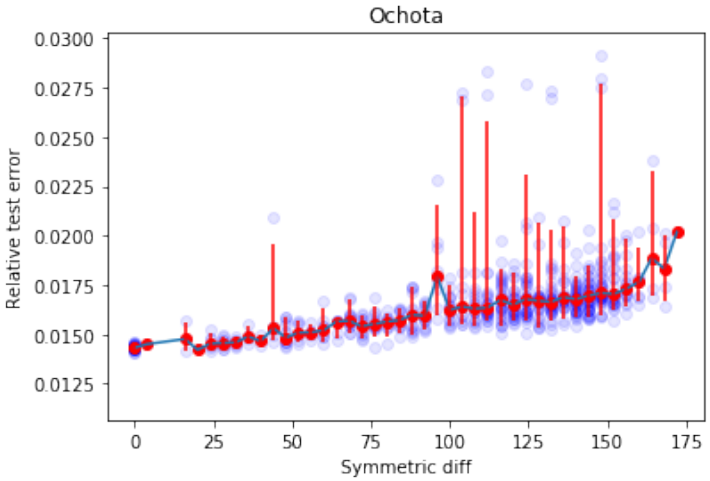


Figure 8.23: Mean relative error achieved by a GNN on the test set for "Ochota" after roughly 330 epochs of training, plotted as a function of the distance of a random graph to the true one. Vertex label permutation was used to generate random graphs. Error bars correspond to 5% sample quantiles. Red dots denote the median result. The figure is from [311].

8.3.1.9.5 Conclusions from experiments

Based on the presented evidence, the following conclusions were drawn regarding the introduced sparse graph-based neural networks:

- They outperform FCNN on the original fixed test sets better for the three considered road networks ("Ochota", "Mokotów", and "Centrum" regions).
- They generalize significantly better than FCNN near unseen optimization target function minima.
- They make it easier to find inputs that correspond to small simulator outputs. Most of the time, the values obtained are also lower than those obtained with the help of FCNN.
- They produce smoother gradient descent trajectories that are less prone to overfitting, for multiple hyperparameter settings. By using randomly

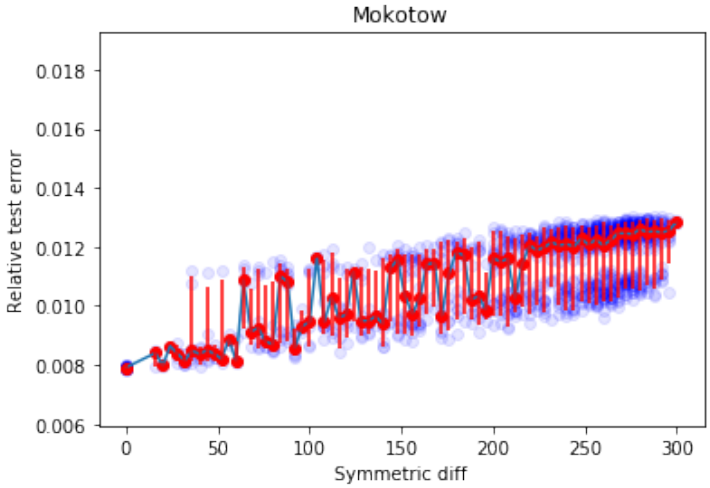


Figure 8.24: Mean relative error achieved by a GNN on the test set for "Mokotów" after roughly 330 epochs of training, plotted as a function of the distance of a random graph to the true one. Vertex label permutation was used to generate random graphs. Error bars correspond to 5% sample quantiles. Red dots denote the median result. The figure is from [311].

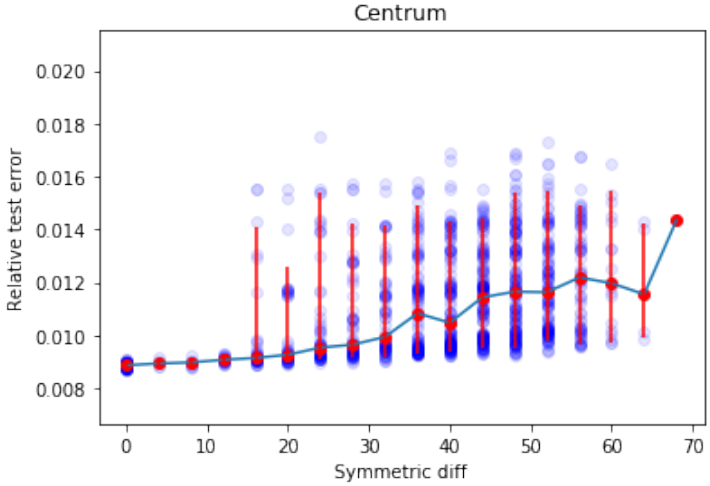


Figure 8.25: Mean relative error achieved by a GNN on the test set for "Centrum" after roughly 330 epochs of training, plotted as a function of the distance of a random graph to the true one. Vertex label permutation was used to generate random graphs. Error bars correspond to 5% sample quantiles. Red dots denote the median result. The figure is from [311].

perturbed graphs, it was also shown that the choice of the proper graph when constructing a GNN is important for achieving good results on a test set.

It is also worth mentioning that the simulator output values obtained when doing the gradient descent runs are significantly lower than ones that could likely be obtained by random search algorithms. In the training datasets (of sizes approximately equal 100k), the minimum simulator output values were around 38000, 285000 and 67000 for "Ochota", "Mokotów", and "Centrum", respectively (with the mean values of approximately 49000, 308000, 76000, cf., Figures 8.14-8.16) whereas the gradient descent method steadily achieved (based on 5%, 10% and 15% percentiles), results below 33000, 270000 and 65000, respectively (with acceptable average relative errors: 1.65%, 0.64% and 1.68%, respectively).

The kind of NN sparsity considered, where only some of the connections are allowed, may be regarded as a kind of regularizer based on the problem graph. It is similar to the L1 regularization as it keeps only some weights non-zero in the trained model. The resulting networks have far fewer parameters than analogous fully connected networks and turn out to generalize significantly better than any architecture considered so far for solving the TSS problem.

Also, sparsity implies that there are fewer parameters to tune and it is easier to train such neural networks than FCNN. In addition, thanks to the fact that the neurons correspond to road network intersections (with traffic signals) or connections between them, it could be possible to investigate the impact of settings at each intersection. Therefore, it can be suspected that analyzing the behavior of these neural networks may also bring a better understanding of urban road traffic and the spatiotemporal impact of traffic conditions on various intersections. In fact, this observation has already triggered further research on the explainability of the introduced GNN, as discussed in Section 9.1.

8.3.1.10 Final experiments with various metaheuristics and surrogate models

In this final batch of experiments related to the Traffic Signal Setting problem and metaheuristics, the goal was to compare the performance of 7 optimization algorithms (CMA-ES, tabu search, genetic algorithms, memetic algorithms, particle swarm optimization, simulated annealing, gradient descent) in combination with surrogate models based on GNN and LightGBM. The experiments were similar to those described in Section 8.3.1.8, but there

were more optimization algorithms tested, the GNN models introduced in Section 8.3.1.9 were used instead of feed-forward fully connected networks, there were road networks from 3 different regions tested, and the experiments were more systematic.

8.3.1.10.1 Setup of experiments

The same datasets as in Section 8.3.1.9 were used (trigonometric transformation on the inputs, as described in Section 8.3.1.9), and the same 90%/10% split on training/test sets was used. Then, several GNN and LightGBM models were trained with various parameters, and 5 best GNN and LightGBM (according to MAPE on the test sets) were selected for further experiments. The selected models are presented in Tables 8.35 and 8.36.

Then, the selected models were applied as surrogate models evaluating the qualities of traffic signal settings in experiments with 7 optimization algorithms (6 metaheuristics and gradient descent):

1. Genetic Algorithms (described in Section 6.2);
2. Particle Swarm Optimization (described in Section 6.3);
3. Tabu Search (described in Section 6.4);
4. Simulated Annealing (described in Section 6.5);
5. CMA-ES (described in Section 6.6);
6. Memetic Algorithms (described in Section 6.7);
7. Gradient Descent (described in Section 6.11).

For each surrogate model, each optimization algorithm was run 100 times starting from different initial solutions (before that, a hyperparameter optimization was performed using Optuna [260] for each optimization algorithm to identify the best configurations of hyperparameters). Hence, for both types of surrogate models (GNN and LightGBM), there were 500 runs for each optimization algorithm, but the exception was gradient descent because it is not possible to run it for LightGBM. The following metrics were collected:

- Minimum, maximum and mean values of the fitness function (the total time of waiting at red signals in the given region) for best settings according to the applied surrogate model among 500 runs;

- Minimum, maximum and mean values of the fitness function for the best settings among 500 runs computed using the deterministic variant of the TSF's microscopic model;
- Minimum, maximum, mean MAPE for best settings among 500 runs;
- Minimum, maximum, mean number of evaluations (calls of the surrogate model) before the stopping criterion is achieved;

The most important metrics seem to be those providing the mean values. Minimum and maximum values are not as relevant because there might be outliers due to the randomness. However, it is also good to know what the best or the worst performance can be.

CMA-ES and gradient descent are suitable for continuous problems but the Traffic Signal Setting problem (as defined in Section 8.1.1) is discrete, so it was necessary to incorporate mapping to the discrete target domain (in this case: the set $\{0, 1, \ldots, 119\}$) before evaluating the candidate solution.

For all algorithms, the key hyperparameters were explored using the Optuna framework [260] independently for each model and each region.

In the case of CMA-ES, there were no observed stable improvements, so the default values for the population size, c_1, and c_μ (cf. Section 6.6) recommended in [141] were applied.

In the case of tabu search, the focus was on Scatter Tabu Search (cf. Section 6.4) in which some best solutions found so far and some best solutions found in the last step were used as the "elite solutions" to generate new candidate solutions. Three hyperparameters were explored: the maximum number of the historical best solutions that can be used as generators, the maximum number of the best solutions from the previous step that can be used as generators, and the size of the tabu list. The best number of the historical "elite solutions" was always in the range $[31, 203]$, the number of the best points from the previous step was in the range $[16, 56]$, the best size of the tabu list was in the range $[24, 162]$.

In the case of genetic algorithms, 5 different sizes of populations were considered (40, 120, 160, 300, 500), 3 numbers of iterations (100, 300, 500), 2 selection methods (tournament selection and roulette wheel selection), 3 crossover methods (one-point crossover, two-point crossover, uniform crossover), and 2 mutation operators (swap mutation and uniform mutation). These hyperparameters were analyzed independently for each model and each region. Their semantics is explained in Section 6.2.

Memetic algorithms 6.7 were combinations of genetic algorithms and hill climbing. The following hyperparameters were tested:

- Size of the population (considered values: $\{20, 40, 120, 200\}$);
- Percentage of the best individuals that took part in the crossover phase (considered values: $\{10\%, 20\%, 40\%, 60\%\}$);
- Percentage of children that take part in the mutation phase (considered values: $\{10\%, 40\%\}$);
- Type of the local search algorithm (considered types: hill climbing, stochastic hill climbing);
- Number of dimensions used for a single operation of the hill climbing algorithm (considered values: $\{1, 3, 6, 10\}$);
- Value of a single step for each dimension in the hill climbing algorithm (considered values: $\{1, 6, 12, 20\}$);
- Number of hill climbing algorithm steps for every individual during one iteration (considered values: $\{1, 5, 10, 60, 120\}$).

For PSO, the following hyperparameters were considered (cf. Section 6.3 for their description):

- Optimizer (considered options: global-best and local-best);
- Number of particles (considered values: $\{10, 11, \ldots, 40\}$);
- c_1 (considered values: range $[1, 10]$ with a step 0.1);
- c_2 (considered values: range $[1, 10]$ with a step 0.1);
- w (considered values: range $[1, 10]$ with a step 0.1);
- Number of neighbors (considered values: $\{1, 2, 3, \ldots, 20\}$);
- Distance metric (considered options: L1, L2).

For simulated annealing, the following hyperparameters were tuned (cf. Section 6.5 for their description):

- Initial temperature (considered values: range $[1000, 100000]$);
- Final temperature used for the stopping criterion (considered values $[1, 250]$);

- Neighbourhood function: contrary to the experiments described in Section 8.3.1.8, this time another approach to generating candidate solutions was tested: Multi-signal change - all traffic signals' settings had a 50% chance to remain the same, 25% chance of an increase by a single step value and 25% chance of decrease by the same single step value. It was assumed that the single step value can be from the set: $\{1, 2, 3, \ldots, 119\}$, and this is a hyperparameter that was tuned.

For the initial and final temperatures, the optimal values were selected by Optuna with a log uniform distribution. In addition, the number of different starting points was 500, while the cooling schedule was exponential cooling:

$$T(step) = T_{max} e^{-ln(T_{max}/T_{min}) \times step/50000} = T_{max} \cdot (\frac{T_{min}}{T_{max}})^{(step/50000)}, \quad (8.14)$$

For gradient descent, 4 algorithms were considered: Momentum [289], Nesterov [249], RMSProp [289], Adam [183]. Their parameters are explained in Section 6.11.

For Momentum, 2 hyperparameters were considered: learning rate (with considered values in the range $[10^{-5}, 0.1]$) and momentum (considered values in the range $[0.01, 1]$). The same hyperparameters and their values were considered for Nesterov. Also, the same values of the learning rate parameter were considered for RMSProp and Adam. In addition, RMSProp and Adam considered different values of the ϵ parameter within the range $[10^{-10}, 10^{-5}]$. RMSProp considered also values of the γ parameter in the range $[0.01, 1]$, while Adam considered values of β_1 and β_2 in the range $[0.7, 1]$.

For all algorithms tested in this batch of experiments, the stopping criterion was initially based on checking convergence. The fitness function value of the best signal setting in a given iteration was compared with the value of the best signal setting some number of iterations behind. If the improvement of the fitness function value was below a certain threshold and the offsets for each signal group were changed by less than 2 seconds, it was determined that there is a convergence. The threshold and the number of iterations to look behind were also the hyperparameters. The best values of the number of iterations to look behind were between 10 and 200, while for the threshold, it was between 0.1 and 30. The exact best value depended on the optimization algorithm, surrogate model, the considered region, and the values of other hyperparameters.

The only exception was simulated annealing, in which the convergence strongly depends on the cooling schedule, as well as maximum and minimum temperatures. These hyperparameters had to be set at the beginning of each algorithm's run, so it was decided (based on some initial experiments) to set the number of iterations to 50000.

8.3.1.10.2 Results of experiments

Tables 8.35 and 8.36 present the selected graph neural networks and LightGBM models that were used in experiments. They display the key model parameters as well as maximum and mean MAPE on the test set. Furthermore, Tables 8.37-8.43 present the results of experiments with optimization algorithms. Each table contains the results for the respective type of model (graph neural networks - GNN, and LightGBM - LGBM) and all 3 regions. The calculated metrics are described in Section 8.3.1.10.1. One important note is that in the case of mean values, the numbers were rounded to integers for simplicity.

Table 8.35: Graph neural network models used in experiments and their MAPE on the test set.

Region	Nr of layers	Nr of channels	Max MAPE	Mean MAPE
Ochota	4	5	1.96%	1.33%
Ochota	2	5	1.91%	1.52%
Ochota	4	4	2.27%	1.39%
Ochota	3	4	2.26%	1.49%
Ochota	3	3	2.07%	1.60%
Centrum	4	5	0.94%	0.79%
Centrum	3	5	0.82%	0.81%
Centrum	3	4	0.86%	0.79%
Centrum	4	4	1.3%	0.84%
Centrum	2	3	0.95%	0.77%
Mokotów	3	4	1.74%	0.77%
Mokotów	2	5	1.36%	0.76%
Mokotów	3	3	2.81%	0.8%
Mokotów	2	3	1.04%	0.77%
Mokotów	2	4	1.31%	0.77%

Table 8.44 contains the best results for each type of surrogate model and for each region. It includes the lowest values of corresponding metrics for all optimization algorithms as well as the name of the algorithm that gives the lowest values. Also, in order to compare the performance of GNN-based and

Table 8.36: LightGBM models used in experiments and their MAPE on the test set.

Region	Boosting	No of trees	Regular.	Max MAPE	Mean MAPE
Ochota	GBDT	5000	L1	4.40%	0.92%
Ochota	GBDT	2500	L2	3.91%	0.87%
Ochota	DART	5000	L2	4.08%	0.82%
Ochota	GBDT	1000	L2	4.39%	0.91%
Ochota	GBDT	5000	L2	3.97%	0.86%
Centrum	GBDT	2500	L2	6.07%	0.82%
Centrum	GBDT	1000	L2	6.04%	0.84%
Centrum	DART	5000	L2	6.12%	0.82%
Centrum	GBDT	5000	L1	6.98%	0.87%
Centrum	GBDT	5000	L2	6.83%	0.82%
Mokotów	DART	2500	L2	8.21%	1.78%
Mokotów	GBDT	1000	L2	9.69%	1.89%
Mokotów	GBDT	5000	L2	9.40%	1.77%
Mokotów	DART	5000	L2	8.30%	1.58%
Mokotów	GBDT	5000	L1	10.18%	1.94%

LGBM-based models, the better (lower) values are bolded for each of the 3 considered regions ("Centrum", "Ochota", and "Mokotów").

It can be seen that in terms of the results evaluated by the surrogate model (that was used to evaluate candidate solutions by the optimization algorithm), the best performance is achieved by population-based metaheuristics, mostly Memetic Algorithms, and CMA-ES. In some cases, Genetic Algorithms were also able to produce the best solution.

In the case of the error of approximation (MAPE) on the set of best signal settings produced by metaheuristics, it was always possible to achieve almost ideal accuracy by at least one algorithm. The maximum and mean errors can be higher, but always at least one algorithm was able to achieve very good accuracy on the whole dataset.

Importantly, CMA-ES and Memetic Algorithms were able to find not only very good settings according to the surrogate models but also according to the microscopic simulation model of TSF. In their case, the error of approximation was usually quite low, even in the worst cases. Genetic algorithms usually also achieved small errors of approximation but sometimes the maximum errors were quite large, which means that the signal settings found using genetic algorithms do not have to be good according to the simulator, even when the graph neural network models are used. Another observation was that the highest MAPE was usually for the "Ochota" region, for which

Table 8.37: Results of experiments for Genetic Algorithms.

	Min/max/mean result for the best settings among 500 runs (model)	Min/max/mean result for the best settings among 500 runs (simulator)	Min/max/mean MAPE for 500 best settings	Min/max/mean number of evaluations (model calls)
GNN Centrum	65142	65484	0%	10500
	66801	67088	12.65%	30500
	65712	65980	1.78%	18050
LGBM Centrum	64074	65268	0%	10500
	66930	67432	10.63%	35500
	65239	65996	1.86%	22550
GNN Ochota	32232	33382	0.01%	10500
	36135	44624	26.96%	45500
	33677	36244	4.92%	27470
LGBM Ochota	30325	32839	0.02%	5500
	38530	34495	28.35%	55500
	33218	33825	6.99%	33520
GNN Mokotów	262717	265674	0.01%	25500
	270940	269618	5.54%	85500
	266515	267530	1.38%	49420
LGBM Mokotów	264002	264262	0.01%	30500
	280502	276658	5.14%	130500
	271879	269825	1.24%	59680

the values of the fitness function were the lowest, while the smallest errors were for "Mokotów", for which the values of the fitness function were the highest. This phenomenon can be observed for both types of surrogate models, so the hypothesis is that this might be caused by the way the input and output data in the training set are transformed into inputs to the trained models. The selected loss function (MSE) may also have an impact. Finally, it is possible that this phenomenon is also related to the structure of the road network and the simulated traffic. This hypothesis can be investigated in future experiments.

In terms of the number of evaluations (calls of the surrogate model), the convergence was fastest in the case of the gradient descent algorithm. However, contrary to metaheuristics, this algorithm cannot be used with the LightGBM models, so for those surrogate models CMA-ES had to use the least number of evaluations.

The quality of traffic signal settings found using gradient descent was usually slightly worse than in the case of Memetic Algorithms or CMA-ES, but the differences were not significant. In the case of Memetic Algorithms, the

Table 8.38: Results of experiments for PSO.

	Min/max/mean result for the best settings among 500 runs (model)	Min/max/mean result for the best settings among 500 runs (simulator)	Min/max/mean MAPE for 500 best settings	Min/max/mean number of evaluations (model calls)
GNN Centrum	66171	64999	0%	33066
	69230	70963	4.07%	90222
	68079	68029	0.85%	49915
LGBM Centrum	66307	65682	0.01%	27081
	70517	71662	4.33%	78016
	68391	68239	0.95%	42360
GNN Ochota	36679	35558	0%	31062
	39638	41094	6.52%	131960
	38322	38248	1.69%	59131
LGBM Ochota	37000	35903	0%	27054
	40694	41953	7.17%	83880
	39260	38856	2.16%	46683
GNN Mokotów	284411	278147	0%	30060
	294871	305936	4.79%	89799
	289886	289948	0.73%	44198
LGBM Mokotów	286976	280120	0.01%	19038
	294212	308343	6.04%	117440
	291456	289730	1.05%	47752

number of evaluations required for convergence was a few orders of magnitude larger, but even before the convergence this algorithm was able to find relatively good candidate solutions quite fast and it was similar for other metaheuristics that needed many evaluations to converge.

Tabu search and Particle Swarm Optimization were also able to find relatively good settings in terms of the results of the surrogate models and the microscopic model of TSF. However, they also required relatively large number of evaluations to reach convergence. The worst results were achieved for simulated annealing. The results of evaluations using surrogate models were relatively good, but the evaluations using the TSF's microscopic model and MAPE showed that the found settings are not acceptable. Perhaps some other values of hyperparameters should be considered in the next experiments (e.g., one of the reasons for poor performance could also be the application of the multi-signal change technique for generating candidate solutions that was described in Section 8.3.1.10.1).

Another interesting observation is that it cannot be concluded which of the considered 2 types of surrogate models (graph neural networks and Light-

Table 8.39: Results of experiments for Tabu Search.

	Min/max/mean result for the best settings among 500 runs (model)	Min/max/mean result for the best settings among 500 runs (simulator)	Min/max/mean MAPE for 500 best settings	Min/max/mean number of evaluations (model calls)
GNN Centrum	65532	63946	0%	74278
	68329	69329	3.48%	297858
	66500	66459	0.85%	127397
LGBM Centrum	64943	64068	0%	157880
	68144	68932	3.56%	237521
	66082	66281	0.98%	174825
GNN Ochota	33174	32484	0%	98862
	38272	38602	6.03%	575749
	35696	35746	0.84%	223655
LGBM Ochota	33263	33234	0%	98641
	38386	40385	9.46%	316194
	36517	36682	2.5%	181638
GNN Mokotów	273370	272357	0%	321807
	285363	298953	7.89%	712818
	279248	280517	1.75%	415614
LGBM Mokotów	276653	270375	0.01%	425310
	289365	292172	3.41%	753915
	284115	283170	0.9%	480095

GBM) are better. In some cases (e.g., "Ochota") the models based on LightGBM performed better, in other cases (e.g., "Mokotów") the models based on graph neural networks were able to produce better results according to the TSF's microscopic simulation. The evaluation times of both models were also similar. One potential advantage of graph neural networks is that they can be used in combination with gradient descent algorithms which are able to find relatively good solutions with very few evaluations using a surrogate model.

8.3.1.10.3 Conclusions from experiments

After running a significant number of experiments for different regions, with different surrogate models, different optimization algorithms, and different values of hyperparameters, it cannot be unequivocally concluded which configurations are the best for the Traffic Signal Setting problem.

Graph neural networks and LightGBM models seem to perform similarly in terms of quality as surrogate models. One of the advantages of graph neural

Table 8.40: Results of experiments for Simulated Annealing.

	Min/max/mean result for the best settings among 500 runs (model)	Min/max/mean result for the best settings among 500 runs (simulator)	Min/max/mean MAPE for 500 best settings	Min/max/mean number of evaluations (model calls)
GNN Centrum	66076	69320	1.80%	50000
	69961	83956	18.33%	50000
	68047	76307	10.74%	50000
LGBM Centrum	66218	68660	1.08%	50000
	68878	87235	22.79%	50000
	67596	76307	11.31%	50000
GNN Ochota	36301	42048	8.99%	50000
	39698	58928	34.94%	50000
	38341	49440	22.16%	50000
LGBM Ochota	36151	41124	5.36%	50000
	40135	59379	36.42%	50000
	38637	49379	21.39%	50000
GNN Mokotów	286517	292637	1.41%	50000
	292580	327125	11.63%	50000
	289780	308657	6.08%	50000
LGBM Mokotów	287547	293963	0.34%	50000
	295117	326077	10.63%	50000
	292666	309035	5.26%	50000

networks is that they can be used in combination with the gradient descent method which is relatively fast and requires the smallest number of calls to a surrogate model to converge. However, gradient descent usually converges to a local optimum, so the solutions found using this method are worse than the solutions found using population-based metaheuristics like CMA-ES or Memetic Algorithms. These algorithms need more evaluations to converge but eventually are able to find slightly better (according to the TSF's microscopic model) traffic signal settings. On the other hand, the differences in qualities of the solutions found by the best algorithms are not significant, but it might be also caused by the properties of the optimized fitness functions and their minima in the considered space of possible solutions.

In these experiments, the worst performance was observed in the case of PSO, but one of the reasons may be a different approach to generating candidate solutions (compared to the experiments presented in Section 8.3.1.8).

All in all, most of the tested optimization algorithms achieved quite good results, and found traffic signal settings much better than the best settings in the training datasets (cf. Figures 8.14 - 8.15) in a reasonable time, so it

Table 8.41: Results of experiments for CMA-ES.

	Min/max/mean result for the best settings among 500 runs (model)	Min/max/mean result for the best settings among 500 runs (simulator)	Min/max/mean MAPE for 500 best settings	Min/max/mean number of evaluations (model calls)
GNN Centrum	64447 74577 65564	66107 72746 66263	0.02% 3.1% 0.89%	1067 11000 1886
LGBM Centrum	65112 74327 66701	66395 72112 66533	0.08% 3.34% 1.05%	1100 11000 1746
GNN Ochota	30618 45619 32537	33496 36047 33556	0.02% 4.9% 1.62%	2262 7111 3960
LGBM Ochota	32339 35094 33423	33104 36362 33268	0.04% 8.07% 3.12%	2119 7982 2975
GNN Mokotów	265895 273529 270789	272712 279814 273196	0.02% 3.1% 0.89%	4515 11910 6546
LGBM Mokotów	263429 300598 268209	271254 276542 271489	0.01% 2.74% 1.05%	4605 9045 6226

seems that the general methodology is valid. The recommendations regarding the usage of specific surrogate models and metaheuristics may depend on the specific use case and priorities because usually, it will be a trade-off between the quality of the desired solutions and the time needed to find them.

Table 8.42: Results of experiments for Memetic Algorithms.

	Min/max/mean result for the best settings among 500 runs (model)	Min/max/mean result for the best settings among 500 runs (simulator)	Min/max/mean MAPE for 500 best settings	Min/max/mean number of evaluations (model calls)
GNN Centrum	65120 65878 65514	63811 67970 65596	0% 3.34% 0.77%	85745 267344 137914
LGBM Centrum	63857 65536 64620	63741 67095 65264	0.04% 3.14% 1.21%	81532 254859 128366
GNN Ochota	32126 34018 32617	32340 35348 33440	0.02% 6.9% 2.52%	191751 683676 284723
LGBM Ochota	30000 34530 32457	31928 36572 34064	0% 14.3% 5.4%	190344 797580 263963
GNN Mokotów	262639 268458 265228	261960 278921 268152	0% 4.34% 1.21%	363888 2090718 780221
LGBM Mokotów	263528 276823 269663	263388 282267 270862	0% 4.27% 1.22%	474138 1843716 725135

Table 8.43: Results of experiments for Gradient Descent.

	Min/max/mean result for the best settings among 500 runs (model)	Min/max/mean result for the best settings among 500 runs (simulator)	Min/max/mean MAPE for 500 best settings	Min/max/mean number of evaluations (model calls)
GNN Centrum	65118 67626 65875	64394 68744 66068	0.01% 3.48% 0.71%	69 409 159
GNN Ochota	32112 35180 33011	32389 36456 33832	0.01% 11.45% 2.51%	82 316 143
GNN Mokotów	265606 273310 269620	264892 281048 272990	0.01% 3.5% 1.28%	95 1040 219

Table 8.44: Best (lowest) results for all optimization algorithms and the type of the algorithm by which they were achieved. GNN and LGBM indicate all GNN and LightGBM models, respectively, for the corresponding region ("Centrum", "Ochota", "Mokotów"). CMA means CMA-ES, MA - memetic algorithms, SA - Simulated Annealing, GA - Genetic Algorithms, TS - Tabu Search, PSO - Particle Swarm Optimization, GD - Gradient Descent. The best results for a given region are bolded.

	Min/max/mean result for the best settings among 500 runs (model)	Min/max/mean result for the best settings among 500 runs (simulator)	Min/max/mean MAPE for 500 best settings	Min/max/mean number of evaluations (model calls)
GNN Centrum	64447 (CMA) 65878 (MA) 65514 (MA)	63811 (MA) **66263 (CMA)** 65596 (MA)	**0% (GA,MA,TS,PSO)** **3.1% (CMA)** **0.71% (GD)**	**69 (GD)** **409 (GD)** **159 (GD)**
LGBM Centrum	**63857 (MA)** **65536 (MA)** **64620 (MA)**	**63741 (MA)** 66533 (CMA) **65264 (MA)**	**0% (GA, TS)** 3.14% (MA) 0.95% (PSO)	1100 (CMA) 11000 (CMA) 1746 (CMA)
GNN Ochota	30618 (CMA) **34018 (MA)** 32537 (CMA)	32340 (MA) 35348 (MA) 33440 (MA)	**0% (PSO, TS)** **4.9% (CMA)** **0.84% (TS)**	**95 (GD)** **1040 (GD)** **219 (GD)**
LGBM Ochota	**30000 (MA)** 34530 (MA) **32457 (MA)**	**31928 (MA)** **34495 (GA)** **33268 (CMA)**	**0% (TS, MA, PSO)** 7.17% (PSO) 2.16% (PSO)	2119 (CMA) 7982 (CMA) 2975 (CMA)
GNN Mokotów	**262639 (MA)** **268458 (MA)** **265228 (MA)**	**261960 (MA)** **269618 (GA)** **267530 (GA)**	**0% (MA, TS, PSO)** 3.1% (CMA) **0.73% (PSO)**	**95 (GD)** **1040 (GD)** **219 (GD)**
LGBM Mokotów	263429 (CMA) 276823 (MA) 268209 (CMA)	263388 (MA) 276542 (CMA) 269825 (GA)	**0% (MA)** **2.74% (CMA)** 0.9% (TS)	4605 (CMA) 9045 (CMA) 6226 (CMA)

8.3.2 The problem of optimizing radiotherapy for cancer treatment

8.3.2.1 Experiments with a simulator of cancer growth and genetic algorithms

The goal of this experiment was to discover good radiotherapy protocols (doses and times of applying them) using genetic algorithms for given constraints, as well as to explore whether genetic algorithms can find better protocols thanks to using the accelerated implementation of the cancer growth simulation model described briefly in Sections 4.3 and 8.2.2, and with more details in [7] and [15].

8.3.2.1.1 Setup of experiments

The experiment assumed delivering up to 10 Gy total irradiation in multi-fractions, over a 5-day period. This limit was set to restrict the total amount of radiation exposure to a reasonable level while considering the potential negative impact on health [288]. Each radiotherapy protocol was evaluated by simulating the cancer growth using the applied model for 10 days (5 days of treatment and 5 additional days) to consider the long-term impact of the treatment. Each evaluation was performed using the EMT6/Ro model described in Sections 4.3 and 8.2.2, and in [7, 264, 15].

The fitness function used for the evaluation of protocols was defined as:

$$f(p) = 1500 - n, \tag{8.15}$$

where p denotes the considered protocol, 1500 is the maximal number of sites possibly occupied by cancer, and n denotes the number of sites occupied by cancer at day 10 of treatment using the given protocol. In order to obtain the value of f, the treatment protocol was evaluated on 10 different tumors (but developed from the same initial seed population of 200 cancer cells) from the available library [8]. This evaluation was performed using the simulation tool (a C++ implementation of the model described in Section 8.2.2 and in [264, 15]) that was run on 2 GPUs. For each tumor, the protocol was evaluated 4 times per GPU. This enabled the calculation of 80 samples for every treatment protocol. The fitness for the evaluated protocol was averaged over the fitness of all 80 evaluations. Such a fitness function was used as an objective function for genetic algorithms.

As defined in Section 8.1.2, the radiotherapy protocols are sequences of pairs (dose value, time of dose administration), so the goal of optimization for genetic algorithms can be defined as:

$$\max_{p \in \mathfrak{P}} f(p), \quad \text{such that} \quad \sum_{i=1}^{k} d_i \leqslant 10.$$

where f is the fitness function (as defined in formula 8.15), p represents a protocol, i.e., a sequence $(d_1, t_1), (d_2, t_2), \ldots, (d_k, t_k)$, where k is the number of pairs, while d_i and t_i are dose values and times (in hours from a reference time) of their administration, respectively. If the total irradiation sum is less than 10 Gy, it is always possible to increase the irradiation and increase or at the very least maintain the current level of quality (e.g., value of a fitness function) of the given protocol. However, when two protocols possess similar qualities radiologists may prefer the protocol with a lower sum of dose values.

Therefore, the search space should also include protocols with a sum of doses that are lower than 10 Gy.

Following the work presented in [7], this experiment focused on 2 clinically relevant benchmark protocols, both delivering 10 Gy total irradiation:

- Benchmark I (BMI): two doses of 1.25 Gy each, given daily at 9 am and 3 pm, with a time interval of 6 hours during the day and 18 hours overnight. This benchmark protocol can be also represented as a sequence $((1.25, 9), (1.25, 15), (1.25, 33), (1.25, 39), (1.25, 57), (1.25, 63), (1.25, 81), (1.25, 87))$.

- Benchmark II (BMII): one dose of 2 Gy per day, doses are given every 24 h e.g. at 9 am. This benchmark protocol can be also represented as a sequence $((2.0, 0), (2.0, 24), (2.0, 48), (2.0, 72), (2.0, 96))$.

For these experiments, it was assumed that the potential irradiation doses can be multiples of 0.25 Gy, starting from 0.25 Gy up to a given maximal dose, d_{max}, which was a parameter and its value (from the range $1.25Gy - 4.5Gy$) depended on the goals of the given experiment.

In the first batch of experiments, for BMI, d_{max} was set to 1.25 Gy, while for BMII, d_{max} was set to 2 Gy because such maximum doses were also used in the previous experiments described in [7], so the goal was to determine the impact of accelerating simulations, running more experiments, and investigating more protocols.

In the next batch of experiments, the maximum dose was incrementally increased beyond 2.0 Gy, in 0.5 Gy steps towards 4.5 Gy.

Different time delays between consecutive doses were explored as well with 30-minute gradations. There were 2 batches of experiments carried out to compare the results with both baselines separately. For BMI, a time-delay from 4h up to 26h was allowed: $\{4, 4.5, \ldots, 25.5, 26\}$. For BMII, the allowed time delays were from the set $\{10, 10.5, \ldots, 31.5, 32\}$.

It was assumed that doses should be administered in multiples of 30 minutes (starting from the beginning of the simulation), as smaller intervals between fractions may prove difficult to administer in a clinical setting. The total treatment time was set to 120 hours, which implies that each protocol should have applied irradiation treatment over 240 time steps. To simplify computations, each protocol was represented as a sparse vector of 240 doses. Each position in the vector corresponded to a specific point in time, with a zero value indicating the absence of dose at that time and a non-zero value indicating the fractional dose to be delivered. This representation ensured that each protocol was of equal length.

In each genetic algorithm run, each population consisted of 40 protocols. In the selection operator, 10% (i.e., 4) of the best protocols in a given population were copied to the next generation without modifications and retained in the genetic pool. Next, 40% of the best protocols (i.e., 16) were selected to become *parents*. Then, in the crossover step, 2 from the selected protocols were randomly sampled for reproduction and for generating 2 new protocols to which mutation operators were later applied. This procedure was repeated 18 times, so in total, 36 child protocols were generated and added to the initially copied 4 protocols to create the new generation.

The selection was performed using one of the following operators: *roulette wheel selection*, *tournament selection*, and *N-best selection*. The crossover used one of the following operators: *single-point crossover*, *two-point crossover*, or *uniform crossover*. All these operators are standard selection/crossover operators frequently used in genetic algorithms and are explained in Section 6.2.2. However, due to the restrictions on the irradiation level (the sum of the doses administered during 5 days should not exceed 10 Gy), it was necessary to check if the new protocols were feasible. If not, the crossover was repeated.

The mutation used one of the following operators: *swap mutation*, *split mutation*, *dose time mutation*, or *dose value mutation*. *Swap mutation* swaps the positions of two randomly selected elements in a vector representing the protocol. *Split mutation* splits a large dose into 2 smaller doses. *Dose time mutation* changes the time of administering the given dose to another, randomly selected time. *Dose value mutation* changes the dose to another allowed value. Application of several mutation operators in the same experiment was allowed, but as in the case of the crossover operators, it was important to ensure that mutation does not lead to unfeasible protocols.

All of these operators are also explained in detail in [264]. The particular type of *selection*, *crossover* and *mutation* operators used were set in the configuration file of a given experiment.

The experiments were run using 4 NVIDIA V100 GPUs [255] and with the C++ implementation. An average speed of 80000 simulations per hour was achieved, which is c. 717 faster than the previous implementation in Matlab run on 13 10-core CPUs. The details of the implementation, its calibration, and fidelity verification are described in [264].

8.3.2.1.2 Results of experiments

For BMI, the average value of the fitness function was 1118.95 with a standard deviation of 40.45. In the previous experiments described in [7], assuming that the maximum dose is 1.25 Gy, the best protocol found using genetic algorithms had an average value of 1139.26 with a standard deviation of

33.28. In these experiments, the best protocol had an average value of 1153.62 and a standard deviation of 34.56.

For BMII, the average value of the fitness function was 1144.83 with a standard deviation of 37.69. In the previous experiments described in [7], assuming that the maximum dose is 2 Gy, the best protocol found using genetic algorithms had an average value of 1158.54 with a standard deviation of 43.68. In these experiments, the best protocol had an average value of 1173.96 and a standard deviation of 37.80.

In the next batch of experiments, when the maximum dose was incrementally increased to 4.5 Gy, for the best protocol found, $((2.5, 19), (3.75, 24.5), (3.75, 75.5))$, the average fitness value was 1273.7 with a standard deviation 64.75.

The details can be found in [264].

8.3.2.1.3 Conclusions from experiments

The experiments showed that thanks to accelerating simulations, running more experiments with the genetic algorithm, with more settings and operators, it was possible to find better protocols for radiotherapy. Therefore, it was natural to check if applying the introduced methodology and training surrogate models based on machine learning can help in finding even better protocols (or protocols of similar quality).

8.3.2.2 Experiments with surrogate models and metaheuristics

The goal of these experiments was to test whether the methodology introduced in this thesis can give an advantage in finding good radiotherapy protocols compared to the standard use of genetic algorithms combined with computer simulations. For this purpose, the simulator of cancer growth under radiotherapy treatment that was used in experiments described in Section 8.3.2.1, was used this time to generate a dataset on which several machine learning models were trained to approximate the outcomes of the simulator. Then, experiments with genetic algorithms were run with the same settings and combinations of hyperparameters as in the previous batch (cf. Section 8.3.2.1) with surrogate models used instead of the simulator to evaluate the quality of radiotherapy protocols.

8.3.2.2.1 Setup of experiments

The first step was to generate a dataset using the simulator of the EMT6/Ro model that was also used in experiments described in Section 8.3.2.1. The dataset consisted of 200000 radiotherapy protocols. For each of them, the

simulator was used 100 times using GPUs, and the results of the fitness function (defined according to formula 8.15 as the number of sites not occupied by cancer cells after testing at day 10 of treatment using the given protocol) were averaged. The protocols assumed that radiotherapy doses can be administered only during the 5 days of treatment, each dose can be a multiple of 0.125 Gy between 0.5 Gy and 4.5 Gy, and the sum of all doses can be up to 10 Gy. It was also assumed that doses should be administered in multiples of 10 minutes (starting from the beginning of the simulation).

This dataset was generated by students from the University of Warsaw during their work on a student project which was proposed and supervised by the author of this thesis [15, 117]. The dataset has been made publicly available to facilitate further research on this topic (cf. Appendix A.4).

Later, several machine learning models were trained on this dataset to approximate the outcomes of simulations. The training was performed by a member of the TensorCell team, Anna Warno, and the results were later summarized in her M.Sc. thesis [348], for which the idea was proposed by the author of this thesis. She split the dataset into training, validation, and test sets with a proportion $80\% : 10\% : 10\%$, and tested the following methods:

- LightGBM (LGBM) [178] (tuned using the Tree Parzen Estimator [24]);
- Fully Connected Neural Networks (FCNN): several variants were tested with $2 - 3$ hidden layers and $16 - 512$ neurons per layer, with Leaky ReLU activation function [181], trained for 150 epochs using the Adam optimizer ([183]) with the MAE loss function;
- Convolutional Neural Networks (CNN) [199]: several variants were tested, with different sizes of kernels, strides, and dilatation. In some variants, CNNs were combined with the attention mechanism [337];
- Temporal Fusion Transformer (TFT) [209]: several variants were tested with different values of parameters.

In some experiments, ensemble methods were used, in which several surrogate models were combined to generate a better surrogate model. Finally, the ensemble model was combined with the Margin Ranking Loss (MRL) [247] which is a loss function suitable for regression tasks in which it is required to preserve the order of elements. As explained in Chapter 7, this is exactly the case of training surrogate models for the purpose of evaluating candidate solutions in optimization tasks. Formally, MRL can be defined as follows:

Definition 8.3 *Let x_1 and x_2 be two points from the input space (in this case, radiotherapy protocols) for which the ground truth values (in this case: mean outcomes of situations) are y_1, y_2. Let z_1, z_2 be the values predicted by the trained model for x_1 and x_2, respectively. Then, Margin Ranking Loss (MRL) is defined as follows:*

$$MRL(y_1, y_2, z_1, z_2) = max(0, -a \cdot (z_1 - z_2) + margin) \tag{8.16}$$

where

$$a = \begin{cases} 1 & \text{if } y_1 > y_2 \\ -1 & \text{otherwise} \end{cases} \tag{8.17}$$

and $margin$ is a fixed margin.

Later, the best-performing surrogate model was integrated with genetic algorithms to verify if using such a surrogate model gives an advantage over using the simulator. For these experiments, the same settings of genetic algorithms were used as in experiments described in Section 8.3.2.1.

8.3.2.2.2 Results of experiments

In the experiments conducted by A. Warno [348], the best performance was initially achieved by CNN with attention mechanism: MAPE on the set was at the level 1.26%. Later, the best architecture was trained in an ensemble learning setup in which 3 networks were trained with 3 different loss functions (mean absolute error, mean squared error, and Huber loss [164]), and their average was trained with MAE loss. This approach reduced MAPE to 1.21%. Adding MRL to the ensemble model reduced MAPE further to 1.15%. In addition, evaluating a single radiotherapy protocol using this surrogate model is about 4000 faster than evaluation using the simulator. Also, in the case of using GPUs and evaluating 256 protocols at once, the speedup is about 150000-fold. The details can be found in [348].

Later, the author of this thesis integrated the best surrogate model with genetic algorithms used in experiments described in Section 8.3.2.1. After running experiments with the same settings, it turned out that the genetic algorithm was able to find an even better radiotherapy protocol in the best run: $((1.0, 70.5), (4.5, 93), (4.5, 118.5))$. For that protocol, the evaluation using the trained surrogate model returned 1279, while the evaluation using the simulator gave 1282. In general, the results were similar as in the case of genetic algorithms with evaluating protocols using the simulator, but the

time required to carry out the computations was a few orders of magnitude lower.

8.3.2.2.3 Conclusions from experiments

This batch of experiments showed that the simulator of cancer growth under radiotherapy treatment can be successfully substituted by surrogate models based on machine learning in the task of finding good radiotherapy protocols using genetic algorithms. Not only are the results of the same quality, but the times for running the experiments are several orders of magnitude faster. Of course, a significant amount of time was required earlier to generate the dataset and train the models. However, once the surrogate model is ready, it gives the opportunity to explore an even larger set of candidate protocols. Therefore, the methodology proposed in this thesis seems to be quite useful.

Another interesting conclusion is that ensemble learning techniques as well as combining neural networks with the attention mechanism and considering the Margin Ranking Loss as the loss function can also improve the accuracy of surrogate models. These techniques can potentially lead to designing even better surrogate models in the future, which is also emphasized in Chapter 9.

Since the introduced methodology proved to be useful in finding good control setting not only for road traffic in cities but also for another complex process, it was natural to investigate whether the same methodology can be used in the case of other complex processes and whether there are any limitations of this methodology. This was one of the goals of the next experiments, described in Section 8.3.3, and a comprehensive discussion of identified limitations and ways to overcome them is also included in Section 9.1.

8.3.3 Approximating cellular automata

In both use cases studied in this thesis, the approximated models of complex processes were based on cellular automata and the trained surrogate models based on machine learning were able to approximate the model outcomes with very good accuracy and with relatively small datasets and training times. Achieving good approximation accuracy is guaranteed by the Universal Approximation Theorem [70], but whether and under what circumstances it can be done efficiently is an open question. Therefore, it was natural to consider in which cases cellular automata can be easily approximated using machine learning models. The topic is especially interesting when one considers the fact that some cellular automata are Turing-complete which

means that they have the ability to perform any computation that a Turing machine can perform. One of the most notable examples of Turing-complete cellular automata is the famous "Game of Life" automaton proposed by J. Conway [100, 282]. The simplest cellular automaton that is known to be Turing-complete is "Rule 110" [357, 65].

The author of this thesis decided to investigate how hard it is to train neural networks to predict the states of "Rule 110", "Game of Life", as well as the original Nagel-Schreckenberg model. The experiments were carried out by the students of the University of Warsaw under the supervision of the author of this thesis and their results are summarized in [275].

It is important to note that there are already research papers investigating how neural networks can learn to predict the future states of cellular automata. One of the first works on this topic was presented in [361] where a neural network was used to imitate 1D and 2D binary cellular automata with chaotic or complex dynamics. However, there are not many papers investigating how difficult it is to predict future states of cellular automata, e.g., depending on Turing-completeness, the size of cellular automata, the complexity of the transition rules or other factors. The article that is probably the closest to the study carried out by the author of this thesis and his team is presented in [315], where it was concluded that the size of the neural networks required to learn the input/output function represented by many steps of "Game of Life" is often significantly larger than the minimal network required to implement the function. However, this article was published a few years later than the the experiments presented in this section were carried out.

8.3.3.1 Design of experiments

"Rule 110" was considered for a tape with 10, 32, and 50 cells. "Game of Life" was considered for the board 10×10, 16×16, and 50×50. The Nagel-Schreckenerg (NaSch) model was considered (in its deterministic variant) for the board with 50, 200, and 1000 cells. For each case, the datasets were generated using simulations that were run for 1000 steps for 10^5 different initial configurations, and they were later divided into training and test sets (with 80%/20% split).

Different architectures of neural networks were considered, e.g., feed-forward fully connected networks, convolutional neural networks [199], long short-term memory networks (LSTM) [154], bidirectional recurrent neural networks (BRNN) [296], etc.

8.3.3.2 Results of experiments

It turned out that recurrent architectures achieved the best performance. For "Rule 110", the MAPE of predicting binary states of cells 15 steps in advance was 98.86%, while for 63.87% cases, the entire state of the tape was perfectly predicted (without any mistake). For the "Game of Life", the MAPE of predicting binary states of cells 10 steps in advance was at the level of 89.72%, while for 24.54% cases, the entire state of the board was perfectly predicted. For the NaSch model, the MAPE of predicting binary states of cells 50 steps in advance was at the level of 99.99%, while for 99.86% cases, the entire state of the tape was perfectly predicted. The details can be found in [275].

8.3.3.3 Conclusions from experiments

It seems that for Turing-complete automata ("Rule 110" and "Game of Life") predicting future states is more difficult than for the NaSch model. Also, considering that such cellular automata can be efficiently implemented and accelerated using GPUs, there is not much advantage in terms of the inference time for such a small number of steps, while for predicting the results for more steps in advance, the accuracy is worse. For example, in the task of predicting the states of "Rule 110", 30 steps in advance, the best MAPE was at the level of 74.57%, while only for 0.31% of the cases it was possible to accurately predict the entire state of the tape.

This is in line with some other recent results published in [315], where it was concluded that, in general, it is difficult to predict the future steps of the "Game of Life" using neural networks.

However, it seems that this interesting topic has hardly been studied, and there is potential for further research on approximating the evolution of cellular automata. The author of this thesis suspects that further research on this topic may lead to experimental as well as theoretical (e.g., related to automata theory) results showing the hardness of approximating states of various cellular automata (especially Turing-complete) using machine learning models, which may also have an impact on the potential applicability of the methodology introduced in this thesis. On the other hand, it also seems that for some cellular automata (e.g., the deterministic variant of the Nagel-Schreckenberg model or its extension in the TSF tool), it may be easier to predict some characteristics of the automata's evolution.

8.4 Discussion about the computational efficiency and hardware

The conducted experiments showed that the methodology proposed in this thesis can be very useful in finding good control settings for both investigated complex processes. However, besides the quality of solutions to the optimization problem, it is also important to discuss the computational complexity of the method.

For both considered cases, the most time-demanding part was related to generating the datasets (training, validation, and test sets), as it required running a large number of simulations. Indeed, this process took many hours and required significant computational power. The process of training surrogate models was much faster, as training a single model usually took about $10 - 60$ minutes on a machine with a single GPU. After training, evaluating the models was very fast, so the time required to run optimization algorithms was negligible compared to the previous steps. One can ask if the significant amount of time required to generate the datasets and train surrogate models is worth the benefits.

From the perspective of running systematic experiments for the purpose of this thesis, training surrogate models based on machine learning was indispensable and significantly reduced the time of computations. After generating the datasets, it was possible to test many different metaheuristics with different settings. Carrying out similar experiments with original simulators would require, in total, much more time and computational power. As discussed in Sections 8.3.1.1 and 8.3.2.1, running experiments only with simulators may already lead to finding relatively good solutions. However, as demonstrated in Section 8.3.2.2, combining metaheuristics with surrogate models may help in finding even better solutions. Also, in the case of the Traffic Signal Settings problem, running only a few iterations of genetic algorithms did not give a significant advantage compared to the best solution in a randomly generated population (cf. Section 8.3.1.1). On the other hand, as demonstrated in Section 8.3.1.10, running more evaluations of surrogate models in optimization algorithms may result in finding better traffic signal settings, much better than the best settings in randomly generated training sets (cf. Tables 8.14 - 8.16).

In addition, the process of generating the datasets can be easily parallelized, so that even millions of computer simulations can be run in a reasonable amount of time, given access to sufficient computing power. Also, as demonstrated in Section 8.3.1.6, there is a potential to decrease the size of the training sets up to a certain point and still get a useful accuracy of the

surrogate model. On the other hand, it is much more difficult to parallelize optimization algorithms. For some algorithms, such as gradient descent or hill climbing, candidate solutions must be evaluated sequentially because the next candidate to be evaluated cannot be known until the previous iteration has been completed. For other algorithms, such as genetic algorithms or particle swarm optimization, some degree of parallelization can be implemented, but candidate solutions from successive iterations must also be evaluated sequentially, so the ability to speed up computation is limited.

From the perspective of real-world applications, it is not so certain that generating a large dataset and training surrogate models will bring significant benefits, but as experiments have shown, exploring more candidates can lead to finding better solutions. Considering that finding even slightly better options for traffic signal control can lead to huge savings in the long run, while even a small increase in the chances that patients will be cured of cancer can be priceless, generating a training set to be able to quickly evaluate more reasonable options seems to be a valuable option.

Moreover, having machine learning based surrogate models gives the opportunity to explore transfer learning approaches to adapt the surrogate models and the found solutions to slightly different conditions (e.g., different initial traffic conditions or different initial cancer states). It is also likely that training the surrogate models with inputs that take into account the initial states of the process can lead to even better adaptive properties.

These and other possible extensions of the presented methodology that may lead to applicability in real-world scenarios are discussed in Chapter 9.

Finally, it is important to mention the computational infrastructure used. Despite initial difficulties, the author was able to perform many series of experiments and witnessed how technological advances in regions such as high-performance computing, cloud computing, and machine learning can help overcome computational challenges and make it easy to perform experiments that were very difficult to perform just a few years earlier.

The initial experiments were carried out on the personal computer of the author of this thesis and on several computers in the computer laboratory of the University of Warsaw. Later, the author was awarded several computational grants, e.g., the "AI for Earth" [230], "AWS Cloud Credits for Research" [4], "Google Cloud Research Credits" [119] and computational grants from ICM [161]. Some experiments were also carried out using "PEGAZ" computing cluster at the Interdisciplinary Centre for Computer Modelling at the University of Rzeszów, or using an NVIDIA server.

8.5 Discussion about the role of the author and other contributors

As mentioned in the previous sections, some of the experiments described in this chapter were performed by research collaborators of the author of this thesis, so it is important to clarify the exact role of the author. In all cases described (with a few exceptions where it is explicitly stated in the text), the author proposed the main idea, designed the experiments, and coordinated the entire research, from building the necessary research team to analyzing and disseminating the results. In some cases, the code was implemented and the experiments were conducted by other team members, but in all such cases, the author personally analyzed all results and reviewed most of the code used.

8.6 Final conclusions

The conducted experiments showed that the introduced methodology and the substitution of computationally expensive simulations by surrogate models can be very useful and, after integration with optimization algorithms, can lead not only to a significant acceleration of experiments, testing more algorithms, hyperparameters, and candidate solutions, but also to finding better solutions, as a consequence. The experiments tested many different machine learning models, metaheuristics, and their settings over many series of experiments, where new batches of experiments were designed based on the results of the previous experiments. Importantly, the methodology proved successful for both studied real-world complex processes (vehicular traffic in cities and cancer growth under radiotherapy treatment).

However, it is also important to analyze the limitations of this methodology, as well as approaches to overcome them, and to discuss possible extensions that may lead to applicability in real-world scenarios. These considerations are presented in Chapter 9.

Chapter 9

Possible extensions and real-world applications

9.1 Identified limitations and ways to mitigate them

The introduced method proved to be successful in the optimization of 2 studied complex processes improving the efficiency of computations and giving an opportunity to find better solutions in a reasonable time. However, it is important to discuss its limitations and its potential applicability to the optimization of real-world complex processes.

There are 3 main limitations that have been identified:

1. The mathematical models of complex processes are only approximations of real-world phenomena (such as real traffic in cities), so the solutions found by metaheuristics that are good according to the models (or surrogate models) may not be good in the real world.

2. The solutions are usually found for certain conditions, e.g., assuming constant durations of signal phases, certain road traffic conditions, and known routes, but a natural question is whether the introduced methods can be extended to adapt the found solutions to other conditions, and thereby avoiding expensive computations of the solutions in new settings from scratch.

3. Training machine learning models as surrogates requires a sufficiently large training set and significant computational power.

The first limitation does not result directly from the methodology introduced, but rather from the limitations of the underlying mathematical models

of complex processes [43]. One of the possible mitigations can be to ensure that the mathematical models are as accurate and consistent with real-world data as possible. However, this is very difficult to achieve, and there are also challenges such as access to the results of real-world data (accurate measurements of the real-world complex processes) needed to calibrate and validate the models. The cost of collecting and storing a sufficient amount of data can be huge and may not be affordable. Moreover, even if the models are good for some conditions, they do not have to be accurate enough for other conditions that are not represented in historical data. Therefore, it would be reasonable to accept that in some cases the models are just not accurate enough, so it is necessary to consider using some other techniques, e.g., the classical techniques that are used nowadays for optimizing specific processes.

In some cases, however, it may be worth investing in the collection of large amounts of data to achieve remarkable results. Recent advances in artificial intelligence and data science, and the tremendous success of machine learning models such as computer vision methods, AlphaFold [172] or GPT-4 [258], show that investing in the collection and processing of large amounts of data can pay off and produce amazing results that are difficult to achieve with fewer data. It is possible that with further technological advancement and development of AI, Internet of Things (IoT), cloud computing, graphical processing units (GPU), tensor processing units (TPU), quantum computing, and other domains, as well as with new methods for collecting, storing, and processing huge amounts of data (Big Data), it will be easier, cheaper, and faster to build very accurate mathematical models of complex processes.

One of the possible approaches in this direction could be to train machine learning models to simulate the complex processes directly so that they could be considered direct models rather than surrogate models. This approach may also require the acquisition of large amounts of data from observations of real-world processes, but there are several premises that suggest this approach can be successful. First, generative machine learning models that are able to dynamically generate output after training have been very successful recently. Large language models, such as GPT-4 [258], LLaMa [329], or PALM-2 [62], are just a few notable examples that show that generative models trained on a large set of data can achieve breakthrough performance and are already assisting humans in many tasks, like searching for information on the Internet, creating art, and coding. It is natural to suspect that further progress in this field, accelerated thanks to the already developed foundation models, can bring further advances, and there may appear new foundation models that could be pre-trained on large amounts of data coming from the real-world observations of complex processes and re-trained on data from a new instance of a given model.

Such a transfer learning approach [371] can also help in solving the limitation 2, as existing models (or surrogate models) of a complex process can potentially be trained on new data (but a smaller amount than used for the initial training) to adapt to changing conditions that cannot be fully predicted due to the sensitive dependence on initial conditions and openness of complex systems and the potential impact of external factors. This will not mitigate the third limitation, because the demand for computational power and a large amount of data may increase, but it is possible that a large amount of data and extensive training of machine learning models would be required only once, offline, and then the models could be additionally trained using much smaller datasets related to a given instance of the complex process and using much smaller computational power. Also, based on the history of advances in computing device technology and ongoing research, it is expected that the availability of sufficient computing power will not be an issue, especially considering the current computing capabilities of centers training large machine learning models. It is also expected that the evolution of machine learning will lead to the development of much smaller architectures with similar capabilities to state-of-the-art techniques. In fact, the development of sparse graph neural networks presented in this thesis (Section 8.3.1.9) is also a step in this direction, and in some experiments presented in this thesis (e.g., Section 8.3.1.5), the impact of training machine learning models on smaller datasets was investigated and the results turned out to be promising.

On the other hand, some approaches aim to solve the limitation 2 in the opposite way, building larger machine learning models that consider more settings. In the experiments conducted, a relatively small set of settings was considered - traffic signal offsets (in the case of the Traffic Signal Setting problem) and radiotherapy protocols (in the case of cancer treatment optimization), the other settings were set as constants. However, there are many more settings that could be controlled in order to optimize the considered processes. In the case of traffic signal control, one can consider not only signal offsets but also signal phase durations and traffic conditions, which could be represented by origin-destination matrices. In addition to traffic signal control, some other approaches to traffic optimization can be applied, such as optimizing routes of fleets, which could be especially useful in the world with connected autonomous vehicles that can communicate with each other and with the road infrastructure in order to find the best routes that optimize the entire traffic (and not only routes of individual vehicles). In the case of cancer treatment, radiotherapy can be enhanced using other treatment methods like chemotherapy or hormonal therapy, so the protocols for their application could also be considered. Such extensions may also result in much larger spaces of possible solutions, so from an optimization point of

view, the problems may become even more challenging. However, it can be expected that also in such cases metaheuristics would be able to find reasonably good solutions. In the conducted experiments, metaheuristics already had to explore very large spaces of possible solutions and evaluate only a very tiny fraction of the available options, but many metaheuristics (such as genetic algorithms) managed to return quite good solutions. However, from the perspective of training surrogate models based on machine learning, the difficulty is more severe. Adding more input features may mean that the training set should be larger and the model architectures should have more parameters.

The author of this thesis has already conducted preliminary experiments with this approach and considered the Traffic Signal Setting problem with 2 additional parameters per signal group - they corresponded to durations of green signal phases in 2 (conflicting) directions. It was assumed that these durations can be taken from the set $\{20, 21, \ldots, 80\}$, while the offsets can be taken from the set $0, 1, 2, \ldots, MAX$, where MAX depends on the sum of the durations of the green signal phases in both directions to ensure traffic safety. The new dataset (generated using the deterministic variant of the TSF's microscopic model) consisted of 1470972 different signal settings for 21 signal groups in the "Ochota" region (used in other experiments too) and is publicly available to facilitate further research (cf. Section A.3). 80% of this dataset was used as the training set and the remaining 20% was the test set. In the experiments conducted by the author of this thesis together with the members of the TensorCell team, the standard BERT (Bidirectional Encoder Representations from Transformer) language model (BERT-base-uncased) [76] from the Hugging Face library [356] was used as the starting point for training, and then it was trained on the generated dataset using the Adam optimizer [183], with a batch size of 64, a learning rate of $5 \cdot 10^{-5}$, and a training time of 12 epochs. It turned out that the new model can give MAPE at the level of 1.99% and MAXAPE=17.79%. The MAPE seems to be low enough to consider the trained model as a good surrogate mode. The MAXAPE is quite high, but further analysis revealed that MAXAPE99 (maximum absolute percentage error among the best 99% results - 99-th percentile) was at the level of 6.64%, so the model performs quite well in most of the cases (however, some outliers exist). Moreover, the performance was much better than in the case of surrogate models based on LightGBM, FCNN, graph convolutional neural networks introduced in [184] or sparse graph neural networks introduced in Section 8.3.1.9, which did not give satisfactory results in this case. The details were published in a scientific paper coauthored by the author of this thesis and published at a workshop of the NeurIPS 2020 conference [322].

These results showed the potential of language models and architectures based on transformers, so this potential was further explored in the next experiments trying to combine graph neural networks with the attention mechanism in the form of graph attention networks (GAT) proposed in [338]. Initial experiments on the dataset for the "Ochota" region with 21 signal groups did not show any advantage over sparse graph neural networks introduced in Section 8.3.1.9, but the results were at the same level, which indicates that there might be potential to explore such architectures further.

Another important issue related to the limitation 1 is a consequence of the fact that all measurements of physical devices are subject to errors, and the sensitive dependence on initial conditions can lead to increasing errors of the models over time. This issue was partially addressed in Section 2.4, where 10 (similar) tumors developed from an initial seed population of 200 cancer cells were considered. A similar approach based on considering several initial states (close to the result of the measurement) can be applied to other complex processes, but its efficiency should be investigated further.

Also, one can think about more adaptive approaches, in which default solutions found offline are adjusted to dynamically changing conditions. There are also many ways to implement them, ranging from designing problem-specific rules on how to act in certain situations (and this is a standard approach in classical traffic signal control with adaptive controllers [256, 106]), to more general approaches like reinforcement learning, where an agent (or agents) controlling a complex process at hand is trained to make good decisions based on the observed state of the process. This approach has also been investigated by the author of this thesis in collaboration with the members of the TensorCell group and researchers from the Technical University of Melaka (in the case of traffic signal control and solving routing problems), the University of Warsaw and ETH Zurich (in the case of optimizing cancer treatment), with very promising results. In the context of adaptive traffic signal control, it is discussed in detail in Section 9.2.1.

Another issue related to the limitation 1 is caused by the fact that the optima of the optimized functions are usually underrepresented in the training set for surrogate models based on machine learning, so the errors of the surrogate model approximations close to the optima may be too large. This issue was identified and described in Section 8.3.1.6 and in the case of the traffic signal use case was mitigated using the introduced graph neural networks 8.3.1.9. Another way to improve these results could be to use the active learning approach [298] and to retrain the trained models online, during the optimization process, based on the results of evaluations close to local optima. However, this can lead to an increase in the required computational time of the optimization algorithms, and in some preliminary experiments did not

give any observable advantage. Nevertheless, active learning is still a relatively new concept, so it is possible that with advances in machine learning, these techniques may provide better results. If the number of good solutions (found using metaheuristics) to a given optimization problem is large, using them in the future training of surrogate models may lead to better performance. And if the machine learning models are used directly as models of complex processes, and not only as surrogate models, there may be potential for even more efficient training based on points close to local optima.

Some improvement in terms of reducing the size of the required training set and increasing accuracy near local optima (limitation 1 and 3) could potentially be achieved by using better quasi-random number generators like Sobol sequence and there have already been research papers dealing with this topic and showing a potential advantage [234].

In terms of metaheuristics, there are still many approaches and variants of classical metaheuristics that could be explored to improve the quality of solutions and reduce the computation time. For example, in the case of genetic algorithms and other population-based metaheuristics, one can think about applying *island models* [309] in which multiple subpopulations evolve in parallel in isolation. Some promising individuals, called migrants, periodically move from their islands to others according to a predefined communication topology. This approach has been tested by the author of the thesis in the case of genetic algorithms applied to the Traffic Signal Setting problem, in combination with the graph neural networks introduced in Section 8.3.1.9. It did not bring observable improvements in terms of the quality of the found solutions, but there is a potential for improvement in terms of performance by parallelizing computations.

Another approach that could be further investigated is the development of domain-specific operators of metaheuristics. This will not fulfill the goal of building universal methods for optimizing complex processes, but it can potentially improve the performance of metaheuristics in some cases. For example, in the case of the Traffic Signal Setting problem, one can think of crossover or mutation operators based on the proximity of signal groups. Intuitively, signal groups that are geographically close should have more influence on each other than signal groups that are distant. Therefore, good coordination of close signal groups seems to be more important, and traffic engineering practices like creating "green waves" seem to assume it too. Based on this, the author of the thesis investigated the idea of clustering signal groups that are geographically close, assuming that in the crossover process, signal settings belonging to the same cluster are not exchanged. In this approach, for signal groups belonging to the same cluster, the new chromosomes may possess settings inherited from only one parent. To build the

clusters, one can also think about taking into account other factors than just geographic proximity. Intuitively, the number of internal connections between signal groups within each cluster should be large, while the number of external connections between different clusters should be small. Therefore, it might be useful to consider some metrics that measure how good the clustering is in terms of the number of intra-cluster connections and inter-cluster connections. One such metric is the Cheeger constant, which measures the balance between intra-cluster and inter-cluster connections [279] and has already found applications in clustering urban road networks [39]. Similarly, mutation operators can be also domain-specific, and in the case of the Traffic Signal Settings problem, a variant of *swap mutation* can take into account the proximity of signal groups. The author of this thesis has run initial experiments with such proximity-based crossover and mutation operators used in genetic algorithms applied to solving the Traffic Signal Setting problem. They did not yield observable improvements in terms of the quality of the found solutions or speed, but it is possible that combining such operators with other approaches and techniques, like memetic algorithms, can give some improvements, so it is considered an area for further studies.

A similar conclusion holds for the mesoscopic models. The difference between the outcomes of a simple mesoscopic model introduced in Section 5.4 and the microscopic model implemented in TSF (Section 5.3) was too large to consider this model as a good surrogate model, and the time of its evaluation was still significantly longer compared to the surrogate models based on machine learning. Also, designing good mesoscopic models for some complex processes might be challenging, so this method cannot be considered as a universal approach. However, as explained in Section 8.3.1.2, it is still worth considering mesoscopic models for the evaluation of complex processes, as for some complex processes their correspondence with real-world data can be acceptable, and with the increase in computational power, their usage could be valuable. An undisputable advantage of mesoscopic models is that after calibration they do not require as large datasets as surrogate models based on machine learning do.

In addition, it might be also worth to study combinations of the presented approach with interaction with the domain experts, e.g., as in the case of the IGrC approach presented in Section 4.4. Potentially, IGrC can be also applied to build surrogate models [69]. In this thesis, the focus was on the application of metaheuristics and the interaction between the optimization algorithms (like metaheuristics) or a mathematical model of a complex process and the human expert was beyond its scope, but can be an interesting area of future research. Nevertheless, some elements of including the human expert's domain knowledge in the computational model have been already studied in

the case of the graph-based neural networks introduced in Section 8.3.1.9. The interaction with domain experts was also considered in the research on the acquisition of traffic-related knowledge is described in Section 5.7.2. The goal of that research was to learn conceptual levels of traffic congestion and a concept of a traffic jam at a single crossroad by means of a dialog with experts. Such learned high-level concepts and their approximation from low-level traffic data using machine learning models can be also used to infer high-level traffic states (e.g., *traffic jams*) from low-level sensory data and to trigger the process of reconfiguring traffic signal settings (e.g., using metaheuristics).

In the case of surrogate models based on machine learning, there are also potential areas for improvement. For example, another way to mitigate the limitation 1 is to ensure that the surrogate models used to evaluate settings are monotonic in the sense that if one setting is better than another in the real world, the surrogate models should preserve this relation. This can be done using the Margin Ranking Loss [247] that was already used in experiments presented in Section 8.3.2.2. These experiments also showed that constructing ensembles that integrate and average the outcomes of several surrogate models may also lead to better results. Besides MRL, it might be also worth studying other loss functions and potentially even designing new loss functions tailored to the proposed applications.

Another improvement may potentially come thanks to applying explainability techniques to the trained models. One of the advantages of the microscopic or mesoscopic models over surrogate models based on machine learning is that it is relatively easy to understand and explain the process of their evaluation in terms of certain traffic characteristics, in particular, it is possible to monitor (and often even visualize) traffic states. In the case of machine learning models, it is more difficult, but explainable AI (XAI) techniques can help get some useful insights. The author of this thesis has conducted several experiments in collaboration with the members of the TensorCell group, in which two XAI techniques were applied to sparse graph neural networks introduced in Section 8.3.1.9 to understand the importance of signal groups and their impact on the times of waiting at red signals in the considered regions in Warsaw. The first technique was SHAP [219], which is based on the Shapley value, a concept from cooperative game theory that measures the contribution of each player to a game. The method assigns a value to a given player (in this case: input feature to the machine learning model) in the all-player-coalition based on the contribution it would make by joining a coalition of other players, averaging over all possible other-player-collaborations. This method was used to assess the importance of input features (in this case: offsets of traffic signal settings) in deriving the final prediction of the model.

An individual feature (offset) was treated as a player joining the coalition of features and was assigned the Shapley value. The second method was the Zorro method [99], which uses a hard mask to identify important features. It sets certain features as constant and perturbs the values of other features to calculate the change in the outcomes of the surrogate model. The assumption is that if a feature has low importance, changing its value with a random noise from input space should change the prediction less compared to a feature with high importance. For the experiments XAI and testing both techniques, 5 GNN models were selected for each of the 3 districts based on their performance on the test set in the experiments described in Section 8.3.1.9, resulting in 15 models in total. Although the investigated techniques (Zorro method and calculating the Shapley value) are different measures of feature importance, their conclusions were similar and they identified the same signal groups as important. The results of this analysis will be published soon in [127].

Explainability techniques are becoming very popular and important for machine learning but instead of using surrogate models based on machine learning, one can also think about using totally different surrogate models, e.g., the techniques presented in Section 2.2.

Another potential improvement in optimization algorithms is related to the concept of hyper-heuristics [47]. These techniques use AI to automatically generate (meta)heuristics instead of manually designing them. They typically work by searching a space of candidate heuristics using machine learning methods to evaluate their performance and identify promising ones. They can then generate new heuristics by combining or modifying the best-performing ones, or by using machine learning techniques to create entirely new approaches.

In general, most of the introduced methods have also the potential for linear acceleration thanks to parallelization, and with access to more and cheaper computing power, the proposed methods can be run much faster. Finally, significant benefits can be expected from new computational paradigms such as:

1. Quantum computing: It assumes performing computations using quantum-mechanical phenomena such as superposition and entanglement. As mentioned in Section 6.9, there already exist quantum computing approaches and algorithms to optimize some complex processes like road traffic in cities.

2. Neuromorphic computing: This approach is inspired by the structure and function of the human brain. It uses systems that physically implement artificial neural networks to perform computations [295].

3. DNA computing [1]: It uses the properties of DNA molecules to perform computations.

4. Edge computing [303]: In this approach, data is processed by the controlled device itself or by a local computer or server, rather than being transmitted to a data center. For example, one can think about a traffic signal control system in which computations are performed in a distributed fashion directly by controllers and detectors, or by computers installed in vehicles participating in traffic.

9.2 Potential applications and impact of the introduced methodology

Since the introduced methodology gave relatively good results in experiments, it is natural to consider if and under which conditions it can be applied to optimize complex real-world processes. Therefore, Section 9.2.1 discusses the potential applications in traffic management, while Section 9.2.2 discusses the potential applications in cancer treatment.

9.2.1 Traffic management

As discussed in Section 2.3, there are already many approaches to traffic signal control, and many traffic management systems have been implemented, but they still have some drawbacks, so it is worth considering new approaches.

Despite the presented limitations of the introduced methodology (cf. Section 9.1), it can already be used to find default traffic signal settings for typical, repeatable traffic conditions. In such cases, traffic engineers can collect real-world traffic data using detectors like inductive loops, radars, cameras, floating car data from mobile devices, or using vehicle-to-infrastructure communication, and calibrate accurate traffic simulation models that can be used in further experiments. However, real-world data can be usually collected only for those traffic signal settings that are implemented in the traffic control system. Therefore, in order to evaluate the quality of other settings, traffic simulations must be run using the calibrated model. Since this operation can be performed offline, it would be possible to generate a large dataset in an acceptable time and use it to train surrogate models based on machine learning. These surrogate models can then be used to evaluate the quality of traffic signal settings to find sufficiently good solutions, according to the introduced methodology.

This procedure can be applied to all typical repeatable traffic conditions. However, even in such cases, the traffic data collected on a particular day will differ from the data from the analogous period in the past. For example, traffic in a single time period of a given day of a week, like Monday morning rush hours, has some similarities to traffic on other Monday mornings. People travel for the same purposes (e.g., to work or to school), from the same areas, to the same areas, using the same modes of transport, they depart at similar times, etc., so that traffic data reveals some repeatable patterns [3]. However, there is always some randomness and unpredictability, as people usually do not depart at exactly the same time, sometimes they use other modes of transport, sometimes they decide to stay at home because they are sick or work remotely, etc. Therefore, like many complex systems, traffic is a combination of some typical patterns and fluctuations, so the first step should be to identify the patterns, neglect the fluctuations and noise, and then train the machine learning models and find good traffic signal control settings for the identified typical, repeatable patterns. Since the newly encountered traffic will include fluctuations, the traffic control system should adapt to these new conditions. This can potentially be implemented in several ways:

1. Using classical adaptation techniques (e.g., the ones presented in [256] or [106]) that use mathematical formulas and calculations to determine the appropriate signal timings, taking into account factors such as traffic volumes, signal phasing, and cycle lengths;

2. Using transfer learning approaches [371] to improve the existing surrogate models and then run the metaheuristics again;

3. Using reinforcement learning to train AI agents how to adapt the default signal settings to changing traffic conditions.

The first option is the default one and similar approaches are already used in some adaptive traffic signal control systems (e.g., [217]). The second approach is similar to the main methodology presented in this thesis. However, unlike the method used to find good settings for default traffic conditions offline, this time the algorithm should operate in real-time. Therefore, generating large datasets, training the model, and running metaheuristics may be too time-consuming. Perhaps instead of training the surrogate model, traffic simulation could be used explicitly in combination with metaheuristics to evaluate explored traffic signal settings. This would require considerable computing power, but should be technically feasible.

However, the third approach seems to be even more promising. As discussed in Section 2.3, applications of reinforcement learning to traffic management systems are already being considered, mostly in research papers and

pilot projects, but it is likely that with further advances in the development of AI and with greater availability of computational power, these techniques may become more and more popular in practice. The author of this thesis has also started to investigate this approach in collaboration with students from the University of Warsaw [33], members of the TensorCell research group, and scientists from the Technical University of Melaka in Malaysia, who developed a microscopic traffic simulation model based on measurements from cameras installed at several intersections in Melaka. The model was later implemented in the simulation framework SUMO [22] and integrated with a RESCO framework for benchmarking reinforcement learning algorithms for traffic signal control [11]. The initial experiments carried out using popular reinforcement learning methods like PPO [294] or DQN [238] show that it is possible to outperform classical methods like the Webster method [256] which was the main benchmark algorithm. Therefore, it seems that reinforcement learning can potentially lead to better traffic signal control systems, while its integration with metaheuristics and the methodology introduced in this thesis can potentially reduce the size of the space of possible actions, and improve the control's performance.

On the other hand, if the traffic data suggest that conditions are far from any typical patterns, the system should try to find good signal settings again using metaheuristics, and retrain the surrogate models, if necessary.

The schema of the idea for the traffic management system taking advantage of the introduced methodology is shown in Figure 9.1.

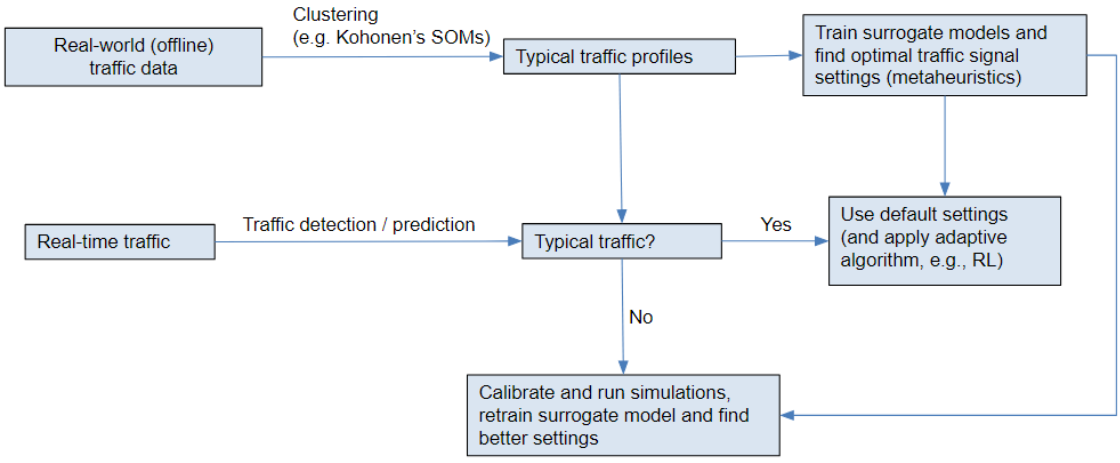


Figure 9.1: Schema presenting the idea for the traffic management system taking advantage of the introduced methodology.

In addition to the use case in Melaka, there are ongoing discussions with leading providers of traffic signal controllers in Poland regarding the implementation of a pilot project. There is also an interest to use the Traffic Simulation Framework and the methodology introduced in this thesis in the SmartCity Lab, currently under development in Chełm (in Poland) [312]. Finally, since the TSF software has been developed using the road network and traffic data from the measurements in Warsaw, there is also a potential to run some pilot projects and plan the possible implementations in this city.

9.2.2 Cancer treatment

The application of the presented methodology in the practice of cancer treatment is much further away from implementation. The reason is that the mathematical model of cancer growth has not been validated in the case of human cancer. It is a model of the EMT6/Ro cell line isolated from the breast of a mouse with a mammary tumor. Also, the model only takes into account radiotherapy, but some initial contacts with professional radiologists revealed that in medical practice, radiotherapy is usually combined with other treatment methods, so it is not possible to measure the true impact of radiotherapy in isolation. As a result, it is currently difficult to develop a reliable method for evaluating the quality of treatment protocols.

One of the possible approaches to overcome these difficulties is to test the developed technique first in an environment similar to the one in which the EMT6/Ro model was developed, i.e. in mammary tumor mice. However, it hard to assess whether transferring the results to people would be possible. Perhaps one can think about testing on mice exactly the same treatment methods as those that are used for people (e.g., radiotherapy in combination with chemotherapy). However, it is not certain that the conclusions that can be drawn from such experiments would be sufficient to test these techniques on humans. In any case, further experiments leading to the possible real applications should be consulted with specialists: radiologists and experts in clinical trials.

However, it is expected that in the future, with greater availability of medical data, new treatment methods can potentially be developed and the methodology introduced in this thesis may become useful.

Chapter 10

Conclusions

The objectives specified at the start of the research presented in this thesis (outlined in Section 1.2) turned out to be challenging. The author gained a deeper understanding of the nature of complex processes and complexity science and summarized his studies in Chapters 3 and 4. He also gained knowledge and experience in other scientific domains such as metaheuristics, machine learning, intelligent transportation systems, mathematical modeling, scientific computing, combinatorial optimization, and quantum computing. These new knowledge and skills have already paid off in the research summarized in this thesis as well as in other research projects in which the author has been involved.

One of the outcomes was proposing a universal method for optimizing complex processes by combining computer simulations, metaheuristics, and surrogate models based on machine learning, which was the second goal of the presented research (see section 1.2). The method was advantageous in terms of computational speed and quality of the solutions found. It was possible to test this method for two use cases from (apparently) completely different domains: traffic signal control for vehicular traffic in cities, and optimization of radiotherapy for cancer treatment.

Several metaheuristics were studied and compared with other optimization algorithms, and it turned out that, in general, population-based metaheuristics (genetic algorithms, memetic algorithms, and CMA-ES) give the best results, but some other algorithms (e.g., gradient descent) are also good.

Several surrogate models based on neural networks and LightGBM were also investigated, and a new architecture of sparse graph neural networks was proposed, which turned out to be quite a good approximation of the traffic simulation results. To approximate the results of cancer growth under radiotherapy treatment, LightGBM also proved to be quite suitable, but even better results were obtained for convolutional neural networks with

an attention mechanism. Further improvement was achieved by using an ensemble of several models and a new loss function (Margin Ranking Loss).

Despite initial computational difficulties, the author managed to develop and study the method and perform many series of experiments. Most of the generated datasets and some software tools and source codes have been made publicly available to facilitate further work on this topic.

Finally, Chapter 9 discusses the identified limitations of the presented method and some ways to overcome them and to further extend this method in order to apply it in practice.

It is worth emphasizing that there are already ongoing discussions about the applications of the considered method in real traffic management systems and the use of the Traffic Simulation Framework in the recently established SmartCity Lab in Chełm (Poland). Since the TSF software has been developed using the road network and traffic data from the measurements in Warsaw, there is also a potential to run some pilot projects and plan the possible implementations in this city. There is also interest from other scientists to test the developed method in a new field - materials science.

The author of this thesis has already established a research group TensorCell [325], which aims to apply AI-based techniques to optimize complex processes and solve combinatorial optimization problems. This group, led by the author of this thesis, has already prepared several research papers and presented them at top transportation and AI conferences (e.g., MT-ITS, NIPS / NeurIPS workshops).

In total, the research covered in this thesis has been summarized and published in 25 research publications and awarded with several prestigious awards, including "LIDER ITS" award for the best R&D work in Intelligent Transportation Systems in Poland in 2015 and 2017 [205, 206], "Top 10 Polish Talents" by MIT Technology Review [235], and "New Europe 100" [252].

Appendix A

Datasets

During the work on this thesis, many datasets were generated to conduct experiments, and the most important datasets have been released to facilitate further work on the presented topics.

A.1 The dataset for Ochota (initial experiments)

The dataset for experiments described in Sections 8.3.1.3-8.3.1.5 can be found at `https://tinyurl.com/pgphdochota15`. Each row contains a traffic signal setting for 15 signal groups, followed by the result of evaluating that setting using the deterministic version of the TSF's microscopic model. The result is the total time of waiting at red signals in a given area during 10 minutes of a simulation using TSF. For each traffic signal setting, the same scenario (with the same vehicles and the same routes) was simulated.

This dataset was generated by the author of this thesis using the TSF software [125].

A.2 The datasets for Ochota, Centrum, and Mokotów

The following datasets were used in experiments described in Sections 8.3.1.6-8.3.1.10 to train surrogate models based on machine learning:

- The dataset with traffic signal settings for Ochota: `https://tinyurl.com/pgphdochota`;

- The dataset with traffic signal settings for Centrum: `https://tinyurl.com/pgphdcentrum`;

- The dataset with traffic signal settings for Mokotów: `https://tinyurl.com/pgphdmokotow`.

In each file, each row contains a single traffic signal setting (as defined in Section 8.1.1) represented as N offsets corresponding to N signal groups (where N is equal to 21 for Ochota, 11 for Centrum, and 42 for Mokotów) with values from the set $\{0, 1, \ldots, 119\}$, followed by the result of evaluating that setting using the deterministic version of the TSF's microscopic model. The result is the total time of waiting at red signals in a given area during 10 minutes of a simulation using TSF (with the given traffic signal setting). For each traffic signal setting, the same scenario (with the same vehicles and the same routes) was simulated.

These datasets were generated by the author of this thesis using the TSF software [125].

A.3 The dataset for experiments with various durations of signal phases

This is the largest dataset that was used to test several architectures of neural networks in additional experiments described in Chapter 9 and in [322]: `https://tinyurl.com/pgphdbert`.

Each row contains a traffic signal setting for the Ochota region with 21 signal groups. However, this time there are 3 values for each signal group: offsets and durations of green signal phases in 2 directions. These durations ($green1$ and $green2$) are from the set $\{20, 21, \ldots, 79, 80\}$, while offsets are from the set $\{0, 1, 2, \ldots, green1 + green2 + 3\}$. In each row, a traffic signal setting consisting of 63 values is followed by the result of evaluating that setting using the deterministic version of the TSF's microscopic model. The result is the total time of waiting at red signals in a given area during 10 minutes of a simulation using TSF. For each traffic signal setting, the same scenario (with the same vehicles and the same routes) was simulated.

This dataset was generated by the author of this thesis using the TSF software [125].

A.4 The dataset used in experiments with cancer treatment

This is the dataset that was used to train surrogate models in experiments with cancer treatment optimization described in Section 8.3.2.2: `https://tinyurl.com/pgphdcto`. It consists of radiotherapy protocols and the results of their evaluation using the simulator of cancer evolution (presented in [264]). The dataset was generated by a group of students supervised by the author of this thesis [15, 117].

Each row contains a description of a single protocol followed by the results of its evaluation using the simulator of cancer evolution under radiotherapy treatment. A single dose should be greater than 0.5 Gy, while the sum of all doses cannot exceed 10 Gy. Therefore, there can be at most 20 doses, which gives 20 slots for the doses and 20 slots for the times of their administration. Thus, each protocol can be represented using 40 values (20 pairs (dose, time of their administration)), so in each row, there are 40 slots for the protocol's representation. They are preceded by the protocol's ID and are followed by the results of the protocol's evaluation (the average number of cancer cells at the end of the treatment).

Appendix B

Software tools and codes

The demo version of the Traffic Simulation Framework software used by the author of this thesis to generate datasets for most of the presented experiments can be found at `https://tinyurl.com/pgphdtsf`.

The code used in experiments with graph neural networks (cf. Section 8.3.1.9) can be found at `https://tinyurl.com/pgphdgnn`. The code was prepared by members of the TensorCell research group led by the author of this thesis and was released with the first publication on using the sparse graph neural networks as surrogate models for traffic simulations [310].

The codes used in experiments aiming to optimize cancer treatment (Section 8.3.2.1) can be found in a repository of Rafał Banaś (one of the authors of the newest version of the simulator of cancer evolution under radiotherapy treatment): `https://tinyurl.com/pgphdemt6ro`. The code was made publicly available together with a research publication in which the tool was used [264].

The code used by a member of the TensorCell group, Anna Warno, to train surrogate models substituting the simulator of cancer evolution under radiotherapy treatment can be found in her Github repository: `https://tinyurl.com/pgphdsmcto`. These models were later used by the author of this thesis in experiments described in Section 8.3.2.2.

The other codes used in the experiments presented in this thesis have not been made publicly available yet but can be made available to interested readers upon request (in such a case please contact the author of this thesis directly at p.gora@mimuw.edu.pl).

Bibliography

[1] L.M. Adleman, "Molecular computation of solutions to combinatorial problems", *Science*, vol. 266, no. 5187, pp. 1021-1024, 1994. DOI: 10.1126/science.7973651.

[2] R. Agustaniah, A. Wicaksono, "Logit Model for Transportation Mode Choice in Berau Regency East Kalimantan", *Journal of Physics: Conference Series*, vol. 1569, 2020. DOI: 10.1088/1742-6596/1569/4/042014.

[3] K.M. Almatar, "Traffic congestion patterns in the urban road network: (Dammam metropolitan area)", *Ain Shams Engineering Journal*, vol. 14, no. 3, 2023. DOI: 10.1016/j.asej.2022.101886.

[4] Amazon, AWS Cloud Credit for Research, official website: `https://aws.amazon.com/government-education/research-and-technical-computing/cloud-credit-for-research`. Last accessed: 15.05.2023.

[5] A.R.A. Anderson and M.A.J. Chaplain, "Continuous and discrete mathematical models of tumor-induced angiogenesis", *Bulletin of Mathematical Biology*, vol. 60, no. 5, pp. 857-899, 1998. DOI: 10.1006/bulm.1998.0042.

[6] C. Angione, E. Silverman, and E. Yaneske, "Using machine learning as a surrogate model for agent-based simulations". *PLOS ONE*, vol. 17, no. 2, 2022. DOI: 10.1371/journal.pone.0263150.

[7] S.D. Angus and M.J. Piotrowska, "A Matter of Timing: Identifying Significant Multi-Dose Radiotherapy Improvements by Numerical Simulation and Genetic Algorithm Search". *PLoS ONE*, vol. 9, no. 12, e114098, 2014. DOI: 10.1371/journal.pone.0114098.

[8] S.D. Angus and M.J. Piotrowska, "A numerical model of EMT6/Ro spheroid dynamics under irradiation: Calibration and estimation

of the underlying irradiation-induced cell survival probability", *Journal of Theoretical Biology*, vol. 320, pp. 23–32, 2013. DOI: 10.1016/j.jtbi.2012.11.035.

[9] J.S. Apte, M. Brauer, A.J. Cohen, M. Ezzati, and C. Arden Pope, "Ambient PM2.5 Reduces Global and Regional Life Expectancy", *Environ. Sci. Technol. Lett.*, vol. 5, no. 9, pp. 546–551, 2018, DOI: 10.1021/acs.estlett.8b00360.

[10] W.B. Arthur, "Complexity and the economy", *Science*, vol. 284, no. 5411, pp. 107-109, 1999. DOI: 10.1126/science.284.5411.107.

[11] J. Ault and G. Sharon, "Reinforcement Learning Benchmarks for Traffic Signal Control", *Proceedings of the Thirty-fifth Conference on Neural Information Processing Systems (NeurIPS 2021) Datasets and Benchmarks Track*, 2021.

[12] H. Bai, J. Zheng, G. Yu, S. Yang, and J. Zou, "A Pareto-based many-objective evolutionary algorithm using space partitioning selection and angle-based truncation", *Information Sciences*, vol. 478, pp. 186-207, 2019. DOI: 10.1016/j.ins.2018.10.027.

[13] Y. Bai, A. Jones, K. Ndousse, A. Askell, A. Chen, N. DasSarma, et al., "Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback", arXiv:2204.05862, 2022. DOI: 10.48550/arXiv.2204.05862.

[14] P. Ball, "Why Society is a Complex Matter", *Meeting Twenty-first Century Challenges with a New Kind of Science*, Springer, 2012. DOI: 10.1007/978-3-642-29000-8.

[15] R. Banaś, J. Bazińska, O. Łobożewicz, and A. Strzałka, "Optimization of cancer treatment", Bachelor thesis in Computer Science at University of Warsaw, supervised by P. Gora, 2019.

[16] Y. Bar-Yam, "Dynamics of Complex Systems", Westview Press, 1997. DOI: 10.1201/9780429034961.

[17] A.L. Barabási and Z. Oltvai, "Network biology: understanding the cell's functional organization", *Nat. Rev. Genet.*, vol. 5, pp. 101–113, 2004. DOI: 10.1038/nrg1272.

[18] A. Bargiela and W. Pedrycz, "Granular Computing. An introduction", Kluwer Academic Publishers, 2003. DOI: 10.1007/978-1-4615-1033-8.

[19] R. Battiti and G. Tecchiolli, "The Reactive Tabu Search", *"ORSA Journal on Computing"*, vol. 6, no. 2, pp. 126–140, 1994. DOI: 10.1287/ijoc.6.2.126.

[20] R. Battiti and G. Tecchiolli, "Training Neural Nets with Reactive Tabu Search", *IEEE transactions on neural networks*, vol. 6, no. 5, pp. 1185–2000, 1995. DOI: 10.1109/72.410361.

[21] M. Baumann and C. Petersen C., "TCP and NTCP: a basic introduction", *Rays*, vol. 30, no. 2, pp. 99-104, 2005.

[22] M. Behrisch, L. Bieker-Walz, J. Erdmann, and D. Krajzewicz, "SUMO – Simulation of Urban Mobility: An Overview", *Proceedings of SIMUL*, 2011.

[23] M. Ben-Akiva, H.N. Koutsopoulos, C. Antoniou and R. Balakrishna, "Traffic Simulation with DynaMIT", *Barceló, J. (eds) Fundamentals of Traffic Simulation. International Series in Operations Research & Management Science*, vol 145, Springer, New York, NY, 2010. DOI: 10.1007/978-1-4419-6142-6_10.

[24] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kegl, "Algorithms for Hyper-Parameter Optimization", *NIPS'11 Proceedings of the 24th International Conference on Neural Information Processing Systems*, pp. 2546–2554, 2011.

[25] C. Berner, G. Brockman, S. Zhang, B. Chan, C. Vicki, P. Dębiak, et al., *"Dota 2 with Large Scale Deep Reinforcement Learning"*, arXiv:1912.06680, 2019.

[26] P.J. Bickel and K.A. Doksum, "Mathematical Statistics: Basic Ideas and Selected Topics", vol. I (Second ed.), 2015. DOI: 10.1201/9781315369266.

[27] O. Biham, A.A. Middleton, and D. Levine, "Self-organization and a dynamical transition in traffic-flow models", *Phys. Rev. A. American Physical Society*, vol. 46, no. 10, pp. 6124-6127, 1992. DOI: 10.1103/PhysRevA.46.R6124.

[28] L. Blincoe, T. Miller, J.-S. Wang, D. Swedler, T. Coughlin, B. Lawrence, et al., "The Economic and Societal Impact of Motor Vehicle Crashes, 2019 (Revised)", Report number: DOT HS 813 403, US Department of Transportation, 2019.

[29] T. Bloom and J. Calvi, "On multivariate minimal polynomials", *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 129, no. 3, pp. 417-432, 2000. DOI: 10.1017/S0305004100004606.

[30] C. Blum and A. Roli, "Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison", *ACM Comput. Surv.*, vol. 35, no. 3, pp. 268-308, 2001. DOI: 10.1145/937503.937505.

[31] R. Bommasani, D.A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, et al., "On the Opportunities and Risks of Foundation Models", arXiv:2108.07258, 2022. DOI: 10.48550/arXiv.2108.07258.

[32] M. Borowski, P. Gora, K. Karnas, M. Błajda, K. Król, A. Matyjasek, et al., "New Hybrid Quantum Annealing Algorithms for Solving Vehicle Routing Problem", *ICCS 2020, LNCS 12142*, pp. 546-561, 2020.

[33] D. Borys, M. Oparka, Z. Partyka, and J. Paszkowska, "Optimising road traffic using Reinforcement Learning", Bachelor thesis at the University of Warsaw, supervised by P. Gora, 2020.

[34] P. Botella, P. Gora, M. Sosnowska, I. Karsznia, and S. Carvajal Querol, "Modelling mobility and visualizing people's flow patterns in rural areas for future infrastructure development as a good transnational land-governance practice", arXiv:2103.01777, 2021.

[35] L. Bottou, "Stochastic Gradient Descent Tricks", *Montavon, G., Orr, G.B., Müller, KR. (eds) Neural Networks: Tricks of the Trade. Lecture Notes in Computer Science*, vol. 7700, 2012. DOI: 10.1007/978-3-642-35289-8_25.

[36] G.E.P. Box and N.R. Draper, "Empirical Model-Building and Response Surfaces". John Wiley & Sons, 1987.

[37] G.E.P. Box and K.B. Wilson, "On the Experimental Attainment of Optimum Conditions", *Journal of the Royal Statistical Society, Series B*, vol. 13, no. 1, pp. 1–45, 1951. DOI: 10.1007/978-1-4612-4380-9_23.

[38] P. Brach, K. Choromański, P. Gora, and A. Jankowski, "Urban Transport Control System - system wspierający organizację ruchu drogowego", B.Sc. thesis at the University of Warsaw, 2008.

[39] P. Brach and K. Choromański, "Urban Transport Control System - innowacyjny system zarządzania ruchem pojazdów", M.Sc. thesis at the University of Warsaw, 2009.

[40] D. Braess, "Über ein Paradoxon aus der Verkehrsplanung" [On a Paradox of Traffic Planning]. Unternehmensforschung, vol. 12, pp. 258–268, 1969.

[41] D. Branston, "Models of Single Lane Time Headway Distributions", *Transportation Science*, vol. 10,no. 2, pp. 125-148, 1976. DOI: 10.1287/trsc.10.2.125.

[42] R.D. Bretherton, "Scoot Urban Traffic Control System — Philosophy and Evaluation", *IFAC Proceedings Volumes*, vol. 23, no. 2, pp. 237-239, 1990. DOI: 10.1016/S1474-6670(17)52676-2.

[43] F. Brooks, "The Mythical Man-Month: Essays on Software Engineering", Addison-Wesley, Boston, 1975 (extended Anniversary Edition in 1995). DOI: 10.1145/390016.808439.

[44] N. Brown and T. Sandholm, "Superhuman AI for multiplayer poker", *Science*, vol. 365, np. 6456, pp. 885–890, 2019. DOI: 10.1126/science.aay2400.

[45] D.J. Buckley, "A Semi-Poisson Model of Traffic Flow", *Transporation Science*, vol. 2, no. 2, pp. 107-132, 1968. DOI: 10.1287/trsc.2.2.107.

[46] W. Buckley, "Modern Systems Research for the Behavioral Scientist: A Sourcebook", Chicago: Aldine Publishing Company, 1968.

[47] E. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and R. Qu, "Hyper-heuristics: a survey of the state of the art", *J Oper Res Soc*, vol. 64, pp. 1695–1724, 2013, DOI: 10.1057/jors.2013.71.

[48] L. Bussolari, P. Contucci, C. Giardina, C. Giberti, F. Unguendoli, and C. Vernia, "Optimization Strategies in Complex Systems", arXiv: math/0309058, 2003.

[49] I.C. Cardenas, "On the use of Bayesian networks as a meta-modeling approach to analyse uncertainties in slope stability analysis", *Georisk: Assessment and Management of Risk for Engineered Systems and Geohazards*, vol. 13, no. 1, pp. 53–65, 2019. DOI: 10.1080/17499518.2018.1498524.

[50] CEBR, prediction of a future cost of traffic gridlocks, `http://www.cebr.com/reports/the-future-economic-and-environmental-costs-of-gridlock`. Last accessed: 10.06.2023.

[51] S. Celtek, A. Durdu, and M.E.M. Ali, "Real-time Traffic Signal Control with Swarm Optimization Methods", *Measurement*, vol. 166, 108206, 2020. DOI: 10.1016/j.measurement.2020.108206.

[52] E.J. Chaisson, "Energy Rate Density as a Complexity Metric and Evolutionary Driver", *Complexity*, vol. 16, no. 3, pp. 27-40, 2011. DOI: 10.1002/cplx.20323.

[53] S. Chan, "Complex adaptive systems", ESD.83 Research Seminar in Engineering Systems, 2001. `http://web.mit.edu/esd.83/www/notebook/ComplexAdaptiveSystems.pdf`. Last accessed: 18.12.2022.

[54] S.W. Chen, C.B. Yang, and Y.H. Peng, "Algorithms for the Traffic Light Setting Problem on the Graph Model", *Proceedings of the 12th Conference on Artificial Intelligence and Applications*, TAAI 2007.

[55] Y. Chen, "The Distance-Decay Function of Geographical Gravity Model: Power Law or Exponential Law?", *Chaos, Solitons & Fractals*, vol. 77, pp. 174-189, 2015. DOI: 10.1016/j.chaos.2015.05.022.

[56] J. Chen, B. Yuan, and M. Tomizuka, "Model-free Deep Reinforcement Learning for Urban Autonomous Driving", pp. 2765–2771, 2019. DOI: 10.1109/ITSC.2019.8917306.

[57] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System", *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794, 2016.

[58] W. Chmiel, "Origin-Destination matrix estimation based on real traffic data", M.Sc. thesis, University of Warsaw, 2015.

[59] F. Chollet et al., repository of the Keras project, `https://github.com/keras-team/keras`. Last accessed: 10.06.2023.

[60] L. Chong and C. Osorio, "A Simulation-Based Optimization Algorithm for Dynamic Large-Scale Urban Transportation Problems", *Transportation Science*, vol. 52, no. 3, 2017. DOI: 10.1287/trsc.2016.0717.

[61] B. Chopard and M. Tomassini, "An Introduction to Metaheuristics for Optimization", Natural Computing Series (NCS), 2018.

[62] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, et al, "PaLM: Scaling Language Modeling with Pathways", arXiv:2204.02311, 2022.

[63] D. Chowdhury and A. Schadschneider, "Self-organization of traffic jams in cities: effects of stochastic dynamics and signal periods", *Physical Review E (Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics)*, vol. 59, no. 2, pp. R1311-R1314, 1999. DOI: 10.1103/PhysRevE.59.R1311.

[64] A. Ciccazzo, G.D. Pillo and V. Latorre, "Support vector machines for surrogate modeling of electronic circuits", *Neural Computing & Applications*, vol. 24, pp. 69–76, 2014. DOI: 10.1007/s00521-013-1509-5.

[65] M. Cook, "Universality in Elementary Cellular Automata", *Complex Systems*, vol. 15, no. 1, pp. 1–40, 2004. DOI: 10.25088/ComplexSystems.15.1.1.

[66] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, "An introduction to algorithms", (3rd edition), MIT Press and McGraw-Hill, 2009. ISBN: 0-262-03384-4.

[67] C. Cotta, J. Gallardo, L. Mathieson, and P. Moscato, "Memetic Algorithms: A Contemporary Introduction", *Wiley Encyclopedia of Electrical and Electronics Engineering*, John Wiley & Sons, pp.1-15, 2016. DOI: 10.1002/047134608X.W8330.

[68] T.M. Cover and J.A. Thomas, "Elements of Information Theory", John Wiley & Sons, Inc., 2005. ISBN: 0-471-20061-1. DOI: 10.1002/047174882X.

[69] I. Cruz-Vega, C. Reyes Garcia, H. Jair Escalante, J. de Jesus Rangel-Magdaleno, and J.M. Ramirez Cortes, "Surrogate modeling based on granular models and fuzzy aptitude functions", *Applied Soft Computing*, vol. 65, pp. 21-32, 2018. DOI: 10.1016/j.asoc.2017.12.016.

[70] G. Cybenko, "Approximation by superpositions of a sigmoidal function", *Mathematics of Control, Signals, and Systems*, vol. 2, no. 4, pp. 303–314, 1989. DOI: 10.1007/BF02551274.

[71] C.F. Daganzo, "The cell transmission model: A dynamic representation of highway traffic consistent with the hydrodynamic theory", *Transportation Research Part B: Methodological*, vol. 28, no. 4, pp. 269–287, 1994. DOI: 10.1016/0191-2615(94)90002-7.

[72] C.F. Daganzo, "The Lagged Cell-Transmission Model", *Proceedings of the 14th Internation Symposium on Transportation and Traffic Theory*, pp. 81–104, 1999.

[73] G.B. Dantzig and J.H. Ramser, "The Truck Dispatching Problem", *Management Science*, vol. 6, no. 1, pp. 80—91, 1959. DOI: 10.1287/mnsc.6.1.80.

[74] K.A. De Jong, *"Evolutionary Computation: A Unified Approach"*, MIT Press, 2006. ISBN: 978-0-262-52960-0.

[75] Deloitte & Targeo.pl, Traffic Jams Report ("Raport o korkach w 7 największych miastach Polski"), 2016, `http://korkometr.targeo.pl/Raport_Korki_2015.pdf`. Last accessed: 10.06.2023.

[76] J. Devlin, M.W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", *NAACL-HLT*, vol. 1, pp. 4171-4186, 2019. DOI: 10.18653/v1/N19-1423.

[77] E.W. Dijkstra, "A note on two problems in connexion with graphs", *Numerische Mathematik*, vol. 1: pp. 269–271, 1959. DOI: 10.1007/BF01386390.

[78] P.A.M. Dirac, "The Principles Of Quantum Mechanics", Oxford University Press, 1936.

[79] K.F. Doerner, M. Gendreau, P. Greistorfer, W. Gutjahr, R.F. Hartl, and M. Reimann, "Metaheuristics: Progress in Complex Systems Optimization", Springer, 2007. DOI: 10.1007/978-0-387-71921-4.

[80] S. Dutta and A. Skowron, "Interactive Granular Computing Model for Intelligent Systems", *Intelligence Science III: 4th IFIP TC 12 International Conference, ICIS 2020, Durgapur, India, February 24-27, 2021, Revised Selected Papers*, Springer International Publishing, vol. 623, pp. 37–48, 2020, DOI: 10.1007/978-3-030-74826-5_4.

[81] J.N. Eckardt, K. Wendt, M. Bornhäuser, and J.M. Middeke, "Reinforcement Learning for Precision Oncology", *Cancers (Basel). 2021 Sep 15*, vol. 13, no. 18, 4624, 2021. DOI: 10.3390/cancers13184624.

[82] B. Edmonds, "What is Complexity? - The philosophy of complexity per se with application to some examples in evolution", Discussion Papers 07, Manchester Metropolitan University, Centre for Policy Modelling., 1995.

[83] A. Einstein, B. Podolsky, and N. Rosen N., "Can Quantum-Mechanical Description of Physical Reality Be Considered Complete?", *Phys. Rev.*, vol. 47, no. 10, pp. 777–780, 1935. DOI: 10.1103/PhysRev.47.777.

[84] S. El-Tantawy, B. Abdulhai, and H. Abdelgawad, "Multiagent Reinforcement Learning for Integrated Network of Adaptive Traffic Signal Controllers (MARLIN-ATSC): Methodology and Large-Scale Application on Downtown Toronto", in *IEEE Conference on Intelligent Transportation Systems*, vol. 14, no. 3, pp. 1140-1150, 2013. DOI: 10.1109/TITS.2013.2255286.

[85] H. Enderling and M.A., Chaplain, "Mathematical modeling of tumor growth and treatment", *Current Pharmaceutical Design*, vol. 20, no. 30, pp. 4934-4940, 2014. DOI: 10.2174/1381612819666131125150434.

[86] A. Engelbrecht, "Particle Swarm Optimization: Global Best or Local Best?", *Proceedings - 1st BRICS Countries Congress on Computational Intelligence*, BRICS-CCI 2013, pp. 124-135, 2013. DOI: 10.1109/BRICS-CCI-CBIC.2013.31.

[87] S. Erlander, N.F. Stewart, "The Gravity Model in Transportation Analysis: Theory and Extensions", *Topics in Transportation*, Taylor & Francis, 1990.

[88] S.N. Ershov, "B-Splines and Bernstein Basis Polynomials", *Phys. Part. Nuclei Lett.*, vol. 16, pp. 593–601, 2019. DOI: 10.1134/S154747711906013X.

[89] M.R. Eslahchi, M. Dehghan, and S. Amani, "The third and fourth kinds of Chebyshev polynomials and best uniform approximation", in *Mathematical and Computer Modelling*, vol. 55, no. 5–6, pp. 1746-1762, 2012. DOI: 10.1016/j.mcm.2011.11.023.

[90] Etymology dictionary, `http://www.etymonline.com/index.php?term=complex`. Last accessed: 18.12.2022. 26.03.2014.

[91] European Environment Agency, "Emissions of air pollutants from transport", `https://www.eea.europa.eu/data-and-maps/indicators/transport-emissions-of-air-pollutants-8/transport-emissions-of-air-pollutants-8`. Last accessed: 10.06.2023.

[92] A.E. Ezugwu, A.K. Shukla, R. Nath, A.A. Akinyelu, J.O. Agushaka, H. Chiroma, and P.K. Muhuri, "Metaheuristics: a comprehensive overview and classification along with bibliometric analysis", in *Artificial Intelligence Review*, vol. 54, pp. 4237–4316, 2021 DOI: 10.1007/s10462-020-09952-0.

[93] R. Fagin, ‘‘Generalized First-Order Spectra and Polynomial-Time Recognizable Sets”, *SIAM-AMS Proceedings*, vol. 7, pp. 27–41, 1974.

[94] A. Ferrara, S. Sacone, S. Siri, “Microscopic and Mesoscopic Traffic Models”, *Freeway Traffic Modelling and Control. Advances in Industrial Control*, Sringer, Cham, 2018, DOI: 10.1007/978-3-319-75961-6_5.

[95] J. Fish, *“Practical Multiscaling”*, John Wiley & Sons, 2014. ISBN: 978-1-118-41068-4.

[96] R. Foote, “Mathematics and complex systems, *Science*, vol. 318, no. 5849, pp. 410–412, 2007. DOI: 10.1126/science.1141754.

[97] A. Forrester and A. Keane, “Recent advances in surrogate-based optimization”, *Progress in Aerospace Sciences*, vol. 45. pp. 50-79, 2009. DOI: 10.1016/j.paerosci.2008.11.001.

[98] P.I. Frazier, “A Tutorial on Bayesian Optimization”, arXiv:1807.02811, 2018.

[99] T. Funke, M. Khosla, and A. Anand, “Hard masking for explaining graph neural networks”, 2021.

[100] M. Gardner, “Mathematical games: The fantastic combinations of John Conway’s new solitaire game ”life””, *Scientific American*, vol. 223, no. 4, pp. 120-123, 1970.

[101] R. Garnett, “Bayesian Optimization”, Cambridge University Press, 2023.

[102] N.H. Gartner, “Demand-Responsive Decentralized Urban Traffic Control”, *IFAC Proceedings Volumes*, vol. 23, no. 2, pp. 241-244, 1990. DOI: 10.1016/S1474-6670(17)52677-4.

[103] M. Gendreau and J.-Y. Potvin., *“Handbook of Metaheuristics”*, Springer, 2010. DOI: 10.1007/978-1-4419-1665-5.

[104] C. Gershenson, *“Design and Control of Self-organizing Systems”*, PhD dissertation, 2007.

[105] C. Gershenson and F. Heylighen, “How can we think the complex?”, *Managing Organizational Complexity: Philosophy, Theory and Application, K. Richardson, (Ed.)”*, pp. 47-62, 2005.

[106] C. Gershenson and D.A. Rosenblueth, "Self-organizing traffic lights at multiple-street intersections", in *Complexity*, vol. 17, no. 4, pp. 23-39, 2012. DOI: 10.1002/cplx.20392.

[107] M. Ghosh, N. Dey, D. Mitra, and A. Chakrabarti, "A Novel Quantum Algorithm for Ant Colony Optimization", *IET Quantum Communication*, vol. 3, pp. 13-29, 2020. DOI: 10.1049/qtc2.12023.

[108] P.G. Gipps, "A behavioral car-following model for computer simulation", *Transportation Research Part B: Methodological*, vol. 15, no. 2, pp. 105-111, 1981. DOI: 10.1016/0191-2615(81)90037-0.

[109] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks", *AISTATS 2010*, 2010.

[110] X. Glorot, A. Bordes, and Y. Bengio, "Deep Sparse Rectifier Neural Networks", *International Conference on Artificial Intelligence and Statistics*, vol. 15, pp. 315-323, 2011.

[111] F. Glover, "Future Paths for Integer Programming and Links to Artificial Intelligence", *Computers and Operations Research*, vol. 13, no. 5, pp. 533—549, 1986. DOI: 10.1016/0305-0548(86)90048-1.

[112] F. Glover, "Tabu Search – Part 1", *ORSA Journal on Computing*, vol. 1, no. 2, pp. 190-–206, 1989. DOI: 10.1287/ijoc.1.3.190.

[113] F. Glover, G.A. Kochenberger, and B. Alidaee, "Adaptive Memory Tabu Search for Binary Quadratic Programs", *Management Science*, vol. 44, no. 3, pp. 336–345, 1998. DOI: 10.1287/mnsc.44.3.336.

[114] Główny Urząd Statystyczny, `https://stat.gov.pl`. Last accessed: 23.12.2022.

[115] A. Godunov, "Difference Scheme for Numerical Solution of Discontinuous Solution of Hydrodynamic Equations", *Math. Sbornik*, vol. 47, pp. 271–306, 1959.

[116] N. Goldenfeld and L.P. Kadano, "Simple lessons from complexity", *Science*, vol. 284, no. 5411, pp. 87–89, 1999. DOI: 10.1126/science.284.5411.87.

[117] A. Goldstein, M. Kardas, P. Kowalkowski, M. Molenda, and Ł. Piekarski, "A machine learning approach for cancer treatment optimization", Bachelor Thesis at the University of Warsaw, supervised by Gora P., 2020.

[118] G. Gomes, J. Ugirumurera, and X.S. Li, "Distributed macroscopic traffic simulation with Open Traffic Models", *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, Rhodes, Greece, pp. 1-6, 2020. DOI: 10.1109/ITSC45102.2020.9294316.

[119] Google Cloud Research Credits program, `https://cloud.google.com/edu/faculty`. Last accessed: 15.05.2023.

[120] P. Gora, "A genetic algorithm approach to optimization of vehicular traffic in cities by means of configuring traffic lights", *Emergent Intelligent Technologies in the Industry*, pp. 1-10, 2011. DOI: 10.1007/978-3-642-22732-5_1.

[121] P. Gora, "Adaptive planning of vehicular traffic", M.Sc. thesis in Computer Science, University of Warsaw, 2010.

[122] P. Gora, "Designing urban areas using traffic simulations, artificial intelligence and acquiring feedback from stakeholders", *Transportation Research Procedia*, vol. 41, pp. 532–534, 2009. DOI: 10.1016/j.trpro.2019.09.089.

[123] P. Gora, "Complex process modelling based on vehicular traffic simulation", M.Sc. thesis in Mathematics, University of Warsaw, 2009.

[124] P. Gora, "Simulation-based traffic management system for connected and autonomous vehicles", *Road Vehicle Automation 4*, Springer, pp. 257-266, 2017. DOI: 10.1007/978-3-319-60934-8_21.

[125] P. Gora, "Traffic Simulation Framework - a Cellular Automaton based tool for simulating and investigating real city traffic", *Recent Advances in Intelligent Information Systems*, pp. 641-653, 2009.

[126] P. Gora and M. Bardoński, "Training neural networks to approximate traffic simulation outcomes", *2017 5th IEEE International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS), Naples, Italy*, pp. 889-894, 2017. DOI: 10.1109/MTITS.2017.8005639.

[127] P. Gora, D. Bogucki, and M.L. Bölüm, "Explainability of surrogate models for traffic signal control", accepted for publication as a chapter in the book *Explainable AI for Intelligent Transportation System*.

[128] P. Gora, C. Katrakazas, A. Drabicki, F. Islam, and P. Ostaszewski, "Microscopic traffic simulation models for connected and automated

vehicles (CAVs) – state-of-the-art", *Procedia Computer Science*, vol. 170, pp. 474-481, 2020. DOI: 10.1016/j.procs.2020.03.091.

[129] P. Gora and K. Kurach, "Approximating Traffic Simulation using Neural Networks and its Application in Traffic Optimization", *'NIPS 2016 Workshop on Nonconvex Optimization for Machine Learning: Theory and Practice*, 2016.

[130] P. Gora, M. Możejko, M. Brzeski, A. Klemenko, and A. Kochański, "Investigating performance of neural networks and gradient boosting models approximating microscopic traffic simulations in traffic optimization tasks", presented at *NeurIPS 2018 Workshop "Machine Learning for Intelligent Transportation Systems*, arXiv:1812.00401, 2018.

[131] P. Gora and P. Pardel, "Application of genetic algorithms and high-performance computing to the Traffic Signal Setting problem", *24th International Workshop, CS&P 2015*, vol. 1, Rzeszów, pp. 146-157, 205.

[132] P. Gora P. and I. Rüb, "Traffic Models For Self-driving Connected Cars", *Transportation Research Procedia*, vol. 14, pp. 2207-2216, 2016. DOI: 10.1016/j.trpro.2016.05.236.

[133] P. Gora and P. Wasilewski, "Inducing Models of Vehicular Traffic Complex Vague Concepts by Interaction with Domain Experts", *COGNITIVE 2013, The Fifth International Conference on Advanced Cognitive Technologies and Applications*, pp. 120-125, 2013.

[134] R.L. Gordon, W. Tighe, "Traffic Control Systems Handbook", U.S. Department of Transportation, Federal Highway Administration, 2005.

[135] B.D. Greenshields, "A Study of Traffic Capacity". *Proceedings of the Highway Research Board*, vol. 14, pp. 448-477, 1935.

[136] S. Gulliford and I. El Naqa, "Modelling of Radiotherapy Response (TCP/NTCP)", *El Naqa, I., Murphy, M.J. (eds) Machine and Deep Learning in Oncology, Medical Physics and Radiology*, Springer, Cham, 2022. DOI: 10.1007/978-3-030-83047-2_17.

[137] T. Haarnoja, A. Zhou, S. Ha, J. Tan, G. Tucker, and S. Levine, "Learning to Walk via Deep Reinforcement Learning", arXiv:1812.11103, 2018. DOI: 10.15607/RSS.2019.XV.011.

[138] S. Haldenbilen, O. Baskan, and C. Oz, "An Ant Colony Optimization Algorithm for Area Traffic Control", *Ant Colony Optimization - Techniques and Applications*. InTech, Feb. 20, 2013. DOI: 10.5772/51695.

[139] M.H. Halstead, "Elements of Software Science", Elsevier Science Inc., 1977. ISBN: 978-0-444-00205-1.

[140] N. Hansen, "The CMA Evolution Strategy: A Tutorial", arXiv:1604.00772, 2023.

[141] N. Hansen, S.D. Muller, and P. Koumoutsakos, "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES)", in *Evolutionary Computation*, vol. 11, no. 1, pp. 1-–18, 2003. DOI: 10.1162/106365603321828970.

[142] P.E. Hart, N.J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths", *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968. DOI: 10.1109/TSSC.1968.300136.

[143] S. Hawking, "Unified Theory Is Getting Closer, Hawking Predicts", interview with S. Hawking, *SAN JOSE MERCURY NEWS*, 23 January 2000.

[144] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition", *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, pp. 770-778, 2016. DOI: 10.1109/CVPR.2016.90.

[145] K.L. Head, P.B. Mirchandani, and D. Shepherd, "A Hierarchical Framework for Real-Time Traffic Control", *Transportation Research Record 1360, Transportation Research Board, National Research Council*, Washington, D.C., pp. 82-88, 1992.

[146] A. Heinecke, J. Ho, and W.-L. Hwang, "Refinement and Universal Approximation via Sparsely Connected ReLU Convolution Nets", *IEEE Signal Processing Letters*, vol. 27, pp. 1175–1179, 2020. DOI: 10.1109/LSP.2020.3005051.

[147] D. Helbing, "FuturICT – New Science and Technology to Manage Our Complex, Strongly Connected World", arXiv:1108.6131, 2011.

[148] D. Helbing, "Gas-kinetic derivation of Navier-Stokes-like traffic equations", *Physical Review E*, vol. 53, no. 3, pp. 2266–2381, 1996. DOI: 10.1103/PhysRevE.53.2366.

[149] D. Helbing, "Verkehrsdynamik – neue Physikalische Modellieringskonzepte", Springer-Verlag, 1997.

[150] S. Henry and D. Kafura, "Software Structure Metrics Based on Information Flow", *IEEE Transactions on Software Engineering*, vol. SE-7, no. 5, pp. 510-518, 1981. DOI: 10.1109/TSE.1981.231113.

[151] P. Heyken Soares, L. Ahmed, Y. Mao, and C.L. Mumford, "Public transport network optimisation in PTV Visum using selection hyper-heuristics", *Public Transp*, vol. 13, pp. 163–196, 2021. DOI: 10.1007/s12469-020-00249-7.

[152] F. Heylighen, "Classical and Non-classical Representations in Physics I", *Cybernetics and Systems*, vol. 21, no. 4, pp. 423-444, 1990. DOI: 10.1080/01969729008902251.

[153] K. Higashi, J. Satsuma, and T. Tokihiro, "Rule 184 fuzzy cellular automaton as a mathematical model for traffic flow", *Japan J. Indust. Appl. Math.*, vol. 38, pp. 579–609, 2021. DOI: 10.1007/s13160-021-00461-3.

[154] S. Hochreiter and J. Schmidhuber, "Long short-term memory", *Neural Computation*, vol. 9, no. 7, pp. 1735–1780, 1997. DOI: 10.1162/neco.1997.9.8.1735.

[155] J.H. Holland, "Signals and Boundaries: Building Blocks for Complex Adaptive Systems", Cambridge, MA: MIT Press, 2012. DOI: 10.7551/mitpress/9412.001.0001.

[156] S.P. Hoogendoorn and P.H.L. Bovy, "State-of-the-art of Vehicular Traffic Flow Modelling", *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 215, no. 4, pp. 283-303, 2001. DOI: 10.1177/095965180121500402.

[157] S.P. Hoogendoorn and P.H.L. Bovy, "Modelling Multiple User-Class Traffic Flow", *Transportation Research B"*, vol. 34, no. 2, pp. 123-146, 2000.

[158] G. Horn, P. Skrzypek, K. Materka, and T. Przeździęk, "Cost Benefits of Multi-cloud Deployment of Dynamic Computational Intelligence Applications", *Web, Artificial Intelligence and Network Applications*, vol. 927, Springer, Cham, 2019. DOI: 10.1007/978-3-030-15035-8_102.

[159] N. Immerman, "Languages Which Capture Complexity Classes", *STOC '83: Proceedings of the fifteenth annual ACM symposium on Theory of computing*, pp. 347–354, 1983. DOI: 10.1145/800061.808765.

[160] D. Inoue, A. Okada, T. Matsumori, K. Aihara, and H. Yoshida, "Traffic signal optimization on a square lattice with quantum annealing", *Sci. Rep.*, vol. 11, 3303, 2021. DOI: 10.1038/s41598-021-82740-0.

[161] Interdisciplinary Centre for Mathematical and Computational Modelling at University of Warsaw, website: `https://icm.edu.pl/en`. Last accessed: 15.05.2023.

[162] International Conference on Data Mining series, website: `http://www.wikicfp.com/cfp/program?id=1337&s=ICDM&f=International%20Conference%20on%20Data%20Mining`. Last accesses: 22.12.2022.

[163] N. Israeli and N. Goldenfeld, "Computational Irreducibility and the Predictability of Complex Physical Systems", *Physical Review Letters*, vol. 92, no. 7, 074105, 2004. DOI: 10.1103/PhysRevLett.92.074105.

[164] A. Jadon, A. Patil, and S. Jadon, "A Comprehensive Survey of Regression Based Loss Functions for Time Series Forecasting", arXiv:2211.02989, 2022.

[165] A. Jamal, H.M. Al-Ahmadi, F.M. Butt, M. Iqbal, M. Almoshaogeh, and S. Ali, "Metaheuristics for Traffic Control and Optimization: Current Challenges and Prospects", *Search Algorithm - Essence of Optimization*, 2023. DOI: 10.5772/intechopen.99395.

[166] U. Jang, W. Xi, and S. Jha, "Objective metrics and gradient descent algorithms for adversarial examples in machine learning", *Proceedings of the 33rd Annual Computer Security Applications Conference*, pp. 262–277, 2017. DOI: 10.1145/3134600.3134635.

[167] A. Jankowski, "Interactive Granular Computations in Networks and Systems Engineering: A Practical Perspective", *Lecture Notes in Networks and Systems*, vol. 17, Springer 2017. DOI: 10.1007/978-3-319-57627-5.

[168] A. Jankowski, A. Skowron, and M. Szczuka, "Interactive Granular Computing in Rightly Judging Systems", *Rough Sets and Knowledge Technology*, vol. 5589, pp. 1–16, 2009. DOI: 10.1007/978-3-642-02962-2_1.

[169] A.M. Jarrett, D. Faghihi, D.A. Hormuth, E.A.B.F. Lima, J. Virostko, G. Biros, D. Patt, and T.E. Yankeelov, "Optimal Control Theory for Personalized Therapeutic Regimens in Oncology: Background, History,

Challenges, and Opportunities", *Journal of Clinical Medicine*, vol. 9, no. 5, pp. 1314, 2020. DOI: 10.3390/jcm9051314.

[170] Y.-Q. Jiang, P.-J Ma., and S.-G. Zhou, "Macroscopic modeling approach to estimate traffic-related emissions in urban areas", *Transportation Research Part D: Transport and Environment*, vol. 60, pp. 41-55, 2018. DOI: 10.1016/j.trd.2015.10.022.

[171] N.F. Johnson, *"Simply complexity: A clear guide to complexity theory"*, Oneworld Publications, ISBN 9781851686308, 2009.

[172] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, et al., "Highly accurate protein structure prediction with AlphaFold", *Nature*, vol. 596, no. 7837, pp. 583–589, 2021. DOI: 10.1038/s41586-021-03819-2.

[173] T. Kadowaki and H. Nishimori, "Quantum annealing in the transverse Ising model", *Phys. Rev. E.*, vol. 58, no. 5, 5355, 1998. DOI: 10.1103/PhysRevE.58.5355.

[174] G.O. Kagho, M. Balac, K.W. Axhausen, "Agent-Based Models in Transport Planning: Current State, Issues, and Expectations", *Procedia Computer Science*, vol. 170, pp. 726-732, 2020. DOI: 10.1016/j.procs.2020.03.164.

[175] D. Kalashnikov, A. Irpan, P. Pastor, S. Ibarz, A. Herzog, E. Jang, et al., "QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation", arXiv: 1806.10293, 2018. DOI: 10.48550/arXiv.1806.10293.

[176] M. Karplus, "Development of Multiscale Models for Complex Chemical Systems: From H+H2 to Biomolecules (Nobel Lecture)", *Angewandte Chemie International Edition*, vol. 53, no. 38, pp. 9992-10005, 2014. DOI: 10.1002/anie.201403924.

[177] H. Kaths, A. Keler, and K. Bogenberger, "Calibrating the Wiedemann 99 Car-Following Model for Bicycle Traffic", *Sustainability*, vol. 13, no. 6, 3487, 2021. DOI: 10.3390/su13063487.

[178] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, et. al., "LightGBM: A Highly Efficient Gradient Boosting Decision Tree", *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[179] R. Keefe, "Theories of Vagueness", *Philosophy and Phenomenological Research*, vol. 67, no. 2, pp. 491-494, 2003. DOI: 10.1111/j.1933-1592.2003.tb00305.x.

[180] J. Kennedy and R. Eberhart, "Particle Swarm Optimization", *Proceedings of IEEE International Conference on Neural Networks*, vol. IV, pp. 1942–1948, 1995. DOI: 10.1109/ICNN.1995.488968.

[181] M. Khalid, J. Baber, M.K. Kasi, M. Bakhtyar, V. Devi, and N. Sheikh, "Empirical Evaluation of Activation Functions in Deep Convolution Neural Network for Facial Expression Recognition", *2020 43rd International Conference on Telecommunications and Signal Processing (TSP)*, pp. 204-207, 2020, DOI: 10.1109/TSP49548.2020.9163446.

[182] M. Kim, A. Ghate, and M.H. Phillips, "A Markov decision process approach to temporal modulation of dose fractions in radiation therapy planning," *Phys. Med. Biol.*, vol. 54, no. 14, pp. 4455-4476, 2009. DOI: 10.1088/0031-9155/54/14/007.

[183] D.P. Kingma and J.L. Ba, "Adam: a method for stochastic optimization", *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015)*, pp. 1–13, 2015.

[184] T.N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks", *Proceedings of the 5th International Conference on Learning Representations*, 2017.

[185] S. Kirkpatrick, C.D. Gelatt Jr, M.P. Vecchi, "Optimization by Simulated Annealing", *Science*, vol. 220, no. 4598, pp. 671-–680, 1983. DOI: 10.1126/science.220.4598.671.

[186] F. Knight, "Some fallacies in the Interpretation of Social Cost", *Quarterly Journal of Economics*, vol. 38, no. 4, pp. 582-604, 1924. DOI: 10.2307/1884592.

[187] J. Kober, J.A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey", *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238-1274, 2013. DOI: 10.1177/0278364913495721.

[188] A.N. Kolmogorov, "On Tables of Random Numbers", *Theoretical Computer Science*, vol. 207, no. 2, pp. 387-395, 1963. DOI: 10.1016/S0304-3975(98)00075-9.

[189] P. Koonce and L. Rodegerdts, "Traffic signal timing manual", Tech report FHWA-HOP-08-024, United States, Federal Highway Administration, 2008.

[190] A. Kratsios and E. Bilokopytov, "Non-Euclidean Universal Approximation". *Proceedings of the 34th International Conference on Neural Information Processing Systems*, ISBN: 9781713829546, 2020.

[191] O. Krause, D.R. Arbonès, and C. Igel, "CMA-ES with Optimal Covariance Update and Storage Complexity", *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[192] D.G. Krige, "A statistical approach to some mine valuations and allied problems at the Witwatersrand", Master's thesis at the University of Witwatersrand, 1951.

[193] K.R. Krohn and J.L. Rhodes, "Algebraic theory of machines", *Proceedings of the Symposium on Mathematical Theory of Automata*, 1962.

[194] J. Ladyman, J. Lambert, and K. Wiesner, "What is a complex system?", *European Journal for Philosophy of Science*, vol. 3, no. 1, pp 33-67, 2013. DOI: 10.1007/s13194-012-0056-8.

[195] E. Lander, "Initial impact of the sequencing of the human genome", *Nature*, vol. 470, no. 7333, pp. 187–197, 2011. DOI: 10.1038/nature09792.

[196] T. Lanting, A.J. Przybysz, A. Yu. Smirnov, F.M. Spedalieri, M.H. Amin, A.J. Berkley, et al., "Entanglement in a quantum annealing processor", arXiv:1401.3500, 2014.

[197] P.S. Laplace, *"A Philosophical Essay on Probabilities"*, 1814.

[198] E.L. Lawler, "The Travelling Salesman Problem: A Guided Tour of Combinatorial Optimization", John Wiley & sons, 1985.

[199] Y. LeCun and Y. Bengio, "Convolutional Networks for Images, Speech, and Time Series", *The Handbook of Brain Theory and Neural Networks*, MIT Press, pp. 255-258, 1998.

[200] J.P. Lebacque, "The Godunov scheme and what it means for first order traffic flow models", *Lesort, J.B. (ed), Proceedings of the 13th International Symposium of Transportation and Traffic Theory*, Lyon, pp. 647–677, 1996.

[201] D. Lewis, "Estimating the influence of public policy on road traffic levels in greater London", *J. Transport Econ. Policy*, vol. 11, no. 2, pp. 155–168, 1977.

[202] Z. Li and P. Schonfeld, "Hybrid simulated annealing and genetic algorithm for optimizing arterial signal timings under oversaturated traffic conditions", *Journal of Advanced Transportation*, vol. 49, pp. 153–170, 2015. DOI: 10.1002/atr.1274.

[203] M. Li and P.M.B. Vitányi, "An Introduction to Kolmogorov Complexity and Its Applications", *Texts in Computer Science, 3rd ed.*, 2008. DOI: 10.1007/978-0-387-49820-1.

[204] Z. Li and Q. Zhang, "A Simple Yet Efficient Rank One Update for Covariance Matrix Adaptation", arXiv:1710.03996, 2017.

[205] "LIDER ITS 2015", results of the contest, `https://www.itspolska.pl/2015/05/25/wynik-vi-edycji-konkursu-lider-its-2015`. Last accessed: 27.05.2023.

[206] "LIDER ITS 2017", results of the contest, `https://www.itspolska.pl/2017/05/16/wyniki-viii-edycji-konkursu-lider-its-2017`. Last accessed: 27.05.2023.

[207] LightGBM - a website with description of the LightGBM parameters, `https://lightgbm.readthedocs.io/en/latest/Parameters.html`. Last accessed: 31.05.2023.

[208] M.H. Lighthill, G.B. Whitham, "On kinematic waves II: a theory of traffic flow on long, crowded roads", *Proceedings of the Royal Society of London series A*, vol. 229, pp. 317-345. DOI: 10.1098/rspa.1955.0089.

[209] B. Lim, S.Ö. Arık, N. Loeff, and T. Pfister, "Temporal Fusion Transformers for interpretable multi-horizon time series forecasting", *International Journal of Forecasting*, vol. 37, no. 4, pp. 1748-1764, 2021, DOI: 10.1016/j.ijforecast.2021.03.012.

[210] S. Lin, B. De Schutter B., Y. Xi, and J. Hellendoorn, "A simplified macroscopic urban traffic network model for model-based predictive control", *IFAC Proceedings Volumes*, vol. 42, no. 15, pp. 286-291, 2009. DOI: 10.3182/20090902-3-US-2007.0023.

[211] S.P. Ling, L.C. Ming, J.S. Dhaliwal, M. Gupta, C. Ardianto, K.W. Goh, Z. Hussain, N. Shafqat, "Role of Immunotherapy in the Treatment of

Cancer: A Systematic Review", *Cancers*, vol. 14, no. 21, 5205, 2022. DOI: 10.3390/cancers14215205.

[212] X.-Y. Liu, H. Yang, J. Gao, and C. Dan Wang, "FinRL: Deep Reinforcement Learning Framework to Automate Trading in Quantitative Finance", *Proceedings of the Second ACM International Conference on AI in Finance*, no. 1, pp. 1-9, 2021. DOI: 10.1145/3490354.3494366.

[213] G. Liu, W. Chen, H. Chen, and J. Xie, " A Quantum Particle Swarm Optimization Algorithm with Teamwork Evolutionary Strategy", *Mathematical Problems in Engineering*, Hindawi, article ID 1805198, 2019. DOI: 10.1155/2019/1805198.

[214] S. Lloyd, "Measures of Complexity: a non-exhaustive list", Massachusetts Institute of Technology, 2001, Available online: `http://web.mit.edu/esd.83/www/notebook/Complexity.PDF`. Last accessed: 10.06.2023.

[215] K.-R. Lo and S.-L. Tung, "Traffic signal control based on particle swarm optimization", *21st World Congress on Intelligent Transport Systems, ITSWC 2014: Reinventing Transportation in Our Connected World*, 2014.

[216] J.S. Lowengrub, H.B. Frieboes, F. Jin, Y.-L. Chuang, X. Li, P. Macklin, et al., "Nonlinear modelling of cancer: bridging the gap between cells and tumours", *Nonlinearity*, vol. 23, pp. R1-R9, 2010. DOI: 10.1088/0951-7715/23/1/r01.

[217] P.R. Lowrie, "The Sydney Co-ordinated Adaptive Traffic System—Principles, Methodology, Algorithms", *Proceedings of the International Conference on Road Traffic Signaling*, Institution of Electrical Engineers, London, U.K., vol. 207, pp. 67–70, 1982.

[218] S. Luke, *"Essentials of Metaheuristics"*, 2nd edition, 2013, available at `http://cs.gmu.edu/~sean/book/metaheuristics`. Last accessed: 10.06.2023.

[219] S.M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions", *Advances in Neural Information Processing Systems*, vol. 30, pp. 4765-–4774, 2017.

[220] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.Y. Wang, "Traffic flow prediction with big data: a deep learning approach", *IEEE Transactions*

on Intelligent Transportation Systems, vol. 16, no. 2, pp. 865-873, 2015. DOI: 10.1109/TITS.2014.2345663.

[221] C.L. Magee and O.L. de Weck, "Complex system classification", INCOSE International Symposium, 2004. DOI: 14. 10.1002/j.2334-5837.2004.tb00510.x.

[222] V. Maiorov and A. Pinkus, "Lower bounds for approximation by MLP neural networks", *Neurocomputing*, vol. 25, no. 1–3, pp. 81–91, 1999. DOI: 10.1016/S0925-2312(98)00111-8.

[223] R. Martí, M. Laguna, and F. Glover, "Principles of scatter search", *European Journal of Operational Research*, vol. 169, no. 2, pp. 359-372, 2006. DOI: 10.1016/j.ejor.2004.08.004.

[224] G. Mathur, S. Nain, and P.K. Sharma, "Cancer: an overview", *Academic Journal of Cancer Research*, vol. 8. no. 1, pp. 1-9, 2015. DOI: 10.5829/idosi.ajcr.2015.8.1.9336.

[225] T.J. McCabe , "A Complexity Measure", *IEEE Transactions on Software Engineering*, vol. SE-2, no. 4, pp. 308-320, 1976. DOI: 10.1109/TSE.1976.233837.

[226] M.G. McNally, "The Four-Step Model", *Handbook of Transport Modelling*, vol. 1, pp. 35–53, 2007. DOI: 10.1108/9780857245670-003.

[227] Microsoft, description of virtual machines used in experiments, `https://learn.microsoft.com/en-us/azure/virtual-machines/nc-series`. Last accessed: 11.03.2023.

[228] Microsoft, description of the Azure N-series with K80 GPUs, `https://azure.microsoft.com/en-us/blog/azure-n-series-general-availability-on-december-1`. Last accessed: 1.02.2023.

[229] Microsoft .NET Framework, an official website, `https://dotnet.microsoft.com/en-us`. Last accessed: 18.12.2022.

[230] Microsoft, "AI for Earth" grant, official website: `https://www.microsoft.com/en-us/ai/ai-for-earth`. Last accessed: 15.05.2023.

[231] D.C. Mikulecky, *"Definition of complexity"*, `https://www.people.vcu.edu/~mikuleck/cmpxnat.html`. Last accessed: 18.12.2022.

[232] S.L. Miller, "A production of amino acids under possible primitive Earth conditions", *Science*, vol. 117, no. 3046, pp. 528-529, 1953. DOI: 10.1126/science.117.3046.528.

[233] J.H. Miller and S.E. Page, *"Complex adaptive systems: an introduction to computational models of social life"*, Princeton University Press, ISBN:978-0-691-12702-6, 2007.

[234] S. Mishra and T.K. Rusch, "Enhancing Accuracy of Deep Learning Algorithms by Training with Low-Discrepancy Sequences", *SIAM Journal on Numerical Analysis*, vol. 59, no. 3, pp. 1811-1834, 2021. DOI: 10.1137/20M1344883.

[235] MIT Technology Review, information about the announcement of the Top 10 Polish Talents of 2017 in the MIT Innovators Under 35 competition, `https://linktopoland.com/en/mit-technology-review-announces-top-10-polish-talents-2017`. Last accessed: 27.05.2023.

[236] M. Mitchell, "An Introduction to Genetic Algorithms", Cambridge, MA, MIT Press, 1996. DOI: 10.7551/mitpress/3927.001.0001.

[237] M. Mitchell and M. Newman, *"Complex Systems Theory and Evolution"*, in Encyclopedia of Evolution, New York: Oxford University Press, 2002.

[238] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou I., D. Wierstra, and M. Riedmiller, *"Playing Atari with Deep Reinforcement Learning'*, arXiv: 1312.5602, 2013.

[239] J. Mockus, "On Bayesian Methods for Seeking the Extremum and their Application", *IFIP Congress*, pp. 195–200, 1977.

[240] M.J.H. Mogridge, "Travel in towns: jam yesterday, jam today and jam tomorrow?", Macmillan Press, London, 1990. DOI: 10.1007/978-1-349-11798-7.

[241] J. Molina, M. Laguna, R. Marti, and R. Caballero, "SSPMO: A Scatter Tabu Search Procedure for Non-Linear Multiobjective Optimization", *INFORMS Journal on Computing*, vol. 19, no. 1, pp. 91-100, 2007. DOI: 10.1287/ijoc.1050.0149.

[242] *Mono.NET Project*, an official website: `https://www.mono-project.com`. Last accessed: 12.06.2023.

[243] A.J. Monticelli, R. Romero R., and E.N. Asada, "Fundamentals of Tabu Search", *Modern Heuristic Optimization Techniques: Theory and Applications to Power Systems*, pp. 101–122, 2007. DOI: 10.1002/9780470225868.ch6.

[244] G. Moreau, V. François-Lavet, P. Desbordes, and B. Macq, "Reinforcement Learning for Radiotherapy Dose Fractioning Automation", *Biomedicines*, vol. 9, no. 2, 214, 2021. DOI: 10.3390/biomedicines9020214.

[245] M. Możejko, M. Brzeski, Ł. Mądry, Ł. Skowronek, and P. Gora, "Traffic Signal Settings Optimization Using Gradient Descent", *Schedae Informaticae*, vol. 27, pp. 19–30, 2018. DOI: 10.4467/20838476SI.18.002.10407.

[246] K. Nagel and M. Schreckenberg, "A cellular automaton model for freeway traffic", *Journal de Physique I.*, vol. 2, no. 12, pp. 2221–2229, 1992. DOI: 10.1051/jp1:1992277.

[247] M. Nayyeri, X. Zhou, S. Vahdati, H.S. Yazdi, and J. Lehmann, "Adaptive Margin Ranking Loss for Knowledge Graph Embeddings via a Correntropy Objective Function", arXiv:1907.05336, 2019.

[248] F. Neri and C. Cotta, "Memetic algorithms and memetic computing optimization: A literature review", *Swarm and Evolutionary Computation*, vol. 2. pp. 1-14, 2012. DOI: 10.1016/j.swevo.2011.11.003.

[249] Y.E. Nesterov, "A method for solving the convex programming problem with convergence rate $o(1/k^2)$", *Dokl. Akad. Nauk SSSR*, vol. 269, pp. 543–547, 1983.

[250] F. Neukart, G. Compostella, C. Seidel, D. von Dollen, S. Yarkoni, and B. Parney, "Traffic Flow Optimization Using a Quantum Annealer", *Frontiers in ICT. Sec. Quantum Engineering and Technology*, vol. 4, 2017. DOI: 10.3389/fict.2017.00029.

[251] New England Complex Systems Institute, `https://necsi.edu/research`. Last accessed: 25.04.2023.

[252] New Europe 100 list 2017, information about the recognitions, `https://businessinsider.com.pl/rozwoj-osobisty/kariera/new-europe-100-nagrody-dla-innowatorow-nominacje-2017/pmkhqfm`. Last Accessed: 27.05.2023.

[253] D. Ni, "2DSIM: A prototype of nanoscopic traffic simulation", *IEEE IV2003 Intelligent Vehicles Symposium. Proceedings (Cat. No.03TH8683), Columbus, OH, USA*, 2003, pp. 47-52. DOI: 10.1109/IVS.2003.1212881.

[254] NVIDIA P100 GPU, `https://www.nvidia.com/en-us/data-center/tesla-p100`. Last accessed: 1.02.2023.

[255] NVIDIA V100 TENSOR CORE GPU, `https://www.nvidia.com/en-us/data-center/v100`. Last accessed: 3.04.2023.

[256] K. Ohno and H. Mine, "Optimal traffic signal settings—II. A refinement of Webster's method", *Transportation Research*, vol. 7, no. 3, pp. 269-292, 1973, DOI: 10.1016/0041-1647(73)90018-X.

[257] J.J. Olstam and A. Tapani, *"Comparison of Car-Following Models"*, Swedish National Road and Transport Research Institute: Linköping, Sweden, vol. 960, 2004.

[258] OpenAI, *"GPT-4 Technical Report"*, arXiv:2303.08774, 2023.

[259] OpenStreetMap Project, an official website: `http://www.openstreetmap.org`. Last accessed: 12.06.2023.

[260] Optuna framework for hyperparameter optimization, `https://github.com/optuna/optuna`. Last accessed: 15.05.2023.

[261] L.E. Orgel, "Prebiotic chemistry and the origin of the RNA world", *Critical Reviews in Biochemistry and Molecular Biology*, vol. 39, no. 2, pp. 99-123, 2004. DOI: 10.1080/10409230490460765.

[262] J.D. Ortúzar and L.G. Willumsen, "Modelling Transport", (4th ed.), Wiley, Hoboken, 2011. DOI: 10.1002/9781119993308.

[263] C. Osorio and M. Bierlaire, "A Simulation-Based Optimization Framework for Urban Transportation Problems", *Operations Research*, vol. 61, no. 6, pp. 1333-1345, 2013. DOI: 10.1287/opre.2013.1226.

[264] W. Ozimek, R. Banaś, P. Gora, S.D. Angus, and M.J. Piotrowska, "Identifying Promising Candidate Radiotherapy Protocols via GPU-GA in-silico", arXiv:2303.08123, 2023.

[265] B. Pan, U. Demiryurek, C. Shahabi, and C. Gupta, "Forecasting Spatiotemporal Impact of Traffic Incidents on Road Networks", *2013 IEEE 13th International Conference on Data Mining, Dallas, TX, USA*, 2013, pp. 587-596. DOI: 10.1109/ICDM.2013.44.

[266] M. Papageorgiou, "Some Remarks on Macroscopic Traffic Flow Modelling", *Transportation Research A*, vol. 32, no. 5, pp. 323–329, 1998. DOI: 10.1016/S0965-8564(97)00048-7.

[267] J.A. Parejo, A. Ruiz-Cortés, S. Lozano, and P. Fernandez, "Metaheuristic optimization frameworks: a survey and benchmarking", *Soft Comput*, vol. 16, pp. 527–561, 2012, DOI: 10.1007/s00500-011-0754-8.

[268] J.K. Parrish and L. Edelstein-Keshet, "Complexity, pattern, and evolutionary trade-offs in animal aggregation", *Science*, vol. 284, pp. 99–101, 1999. DOI: 10.1126/science.284.5411.99.

[269] S.L. Paveri-Fontana, "On Boltzmann-Like treatments for traffic flow: a critical review of the basic model and an alternative proposal for dilute traffic analysis", *Transportation Research B*, vol. 9, pp. 225-235, no. 4, 1975. DOI: 10.1016/0041-1647(75)90063-5.

[270] H.J. Payne, "Models of Freeway Traffic and Control", *Mathematical Models of Public Systems*, vol. 1, pp. 51–61, 1971.

[271] K. Pearson, "On Lines and Planes of Closest Fit to Systems of Points in Space", *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559-572, 1901. DOI: 10.1080/14786440109462720.

[272] T. Pham Dinh, "Optimization of Complex Systems: Theory, Models, Algorithms and Applications", *Advances in Intelligent Systems and Computing*, vol. 991. Springer, 2020. DOI: 10.1007/978-3-030-21803-4.

[273] M.J. Piotrowska and S.D. Angus, "A quantitative cellular automaton model of in vitro multicellular spheroid tumour growth", *Journal of Theoretical Biology*, vol. 258, no. 2, pp. 165–178, 2009. DOI: 10.1016/j.jtbi.2009.02.008.

[274] G.G. Powathil, M. Kohandel M., S. Sivaloganathan, A. Oza, and M. Milosevic, "Mathematical modeling of brain tumors: effects of radiotherapy and chemotherapy", *Physics in Medicine & Biology*, vol. 52, no. 11, pp. 3291-3306, 2007. DOI: 10.1088/0031-9155/52/11/023.

[275] R. Pragacz, P. Senchanka, N. Siwek N., and S. Zwara, "Approximation of cellular automata simulations using machine learning algorithms", Bachelor thesis at the University of Warsaw, supervised by P. Gora, 2019.

[276] I. Prigogine, R. Herman, and J. Chaiken, "Kinetic Theory of Vehicular Traffic". *Physics Today*, 1972, vol. 25, no. 2, pp. 56–57. DOI: 10.1063/1.3070729.

[277] H. Prothmann, S. Tomforde, J. Branke, J. Hähner, Ch. Mueller-Schloer, and H. Schmeck, "Organic Traffic Control", *Organic Computing - A Paradigm Shift for Complex Systems*, Autonomic Systems, vol. 1, pp. 431-446, 2011. DOI: 10.1007/978-3-0348-0130-0_28.

[278] Pyevolve - library for running experiments with genetic algorithms, `https://pyevolve.sourceforge.net/0_6rc1`. Last accessed: 12.06.2023.

[279] Z. Qiao, J.H. Koolen, and G. Markowsky, "On the Cheeger constant for distance-regular graphs", *Journal of Combinatorial Theory, Series A*, vol. 173, 2020. DOI: h10.1016/j.jcta.2020.105227.

[280] N.V. Queipo, R.T. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, and P.K. Tucker, "Surrogate-based analysis and optimization", *Progress in Aerospace Sciences*, vol. 41, no. 1, pp. 1—28, 2005. DOI: 10.1016/j.paerosci.2005.02.001.

[281] K. Rashmi and R. Gilad-Bachrach, "DART: Dropouts meet Multiple Additive Regression Trees", arXiv: 1505.01866, 2015.

[282] P. Rendell, "Turing machine universality of the Game of Life", *Genet Program Evolvable Mach*, vol. 18, pp. 115–117, 2017. DOI: 10.1007/s10710-016-9284-6.

[283] D. Renfrew and X.-H. Yu, "Traffic signal optimization using Ant Colony Algorithm", *The 2012 International Joint Conference on Neural Networks (IJCNN)*, pp. 1-7, 2012. DOI: 10.1109/IJCNN.2012.6252852.

[284] J. Rhodes, B. Steinberg, "The q-theory of finite semigroups", Springer Verlag, ISBN: 978-0-387-09780-0, 2008. DOI: 10.1007/b104443.

[285] P.I. Richards, "Shock waves on the highway", *Operations Research*, vol. 4, pp. 42–51, 1956. DOI: 10.1287/opre.4.1.42.

[286] D. Rind, "Complexity and climate", *Science*, vol. 284, no. 5411, pp. 105–107, 1999. DOI: science.284.5411.105.

[287] R. Rockne, J.K. Rockhill, M. Mrugala, A.M. Spence, I. Kalet, K. Hendrickson, "Predicting the efficacy of radiotherapy in individual glioblastoma patients in vivo: a mathematical modeling approach", *Physics*

in Medicine & Biology, vol. 55, no. 12, pp. 3271-3285, 2010. DOI: 10.1088/0031-9155/55/12/001.

[288] B. Rosenstein, S. Lymberis, and S.C. Formenti, "Biologic comparison of partial breast irradiation protocols", *International Journal of Radiation Oncology, Biology, Physics*, vol. 60, pp. 1393–1404, 2004. DOI: 10.1016/j.ijrobp.2004.05.072.

[289] S. Ruder, "An overview of gradient descent optimization algorithms", arXiv:1609.04747, 2016. DOI: 10.48550/arXiv.1609.04747.

[290] S.J. Russell and P. Norvig, "Artificial Intelligence: A Modern Approach (2nd ed.)", Upper Saddle River, New Jersey: Prentice Hall, ISBN: 0-13-790395-2, pp. 111–114, 2003.

[291] J. Rzeszótko and S.H. Nguyen, "Machine Learning for Traffic Prediction", *Fundamenta Informaticae*, vol. 119. pp. 407-420, 2012. DOI: 10.3233/FI-2012-745.

[292] N.R. Sabar, L.M. Kieu, E. Chung, T. Tsubota, and P.E.M. de Almeida, "A memetic algorithm for real world multi-intersection traffic signal optimisation problems", *Engineering Applications of Artificial Intelligence*, vol. 63, pp. 45-53, 2017. DOI: 10.1016/j.engappai.2017.04.021.

[293] A. Schadschneider, "Statistical physics of traffic flow", *Physica A: Statistical Mechanics and its Applications*, vol. 285, no. 1-2, pp. 101-120, 2000. DOI: 10.1016/S0378-4371(00)00274-0.

[294] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms", arXiv:1707.06347, 2017.

[295] C.D. Schuman, S.R. Kulkarni, M. Parsa, J.P. Mitchell, P. Date, B. Kay, "Opportunities for neuromorphic computing algorithms and applications", *Nat Comput Sci*, vol. 2, pp. 10–19, 2022. DOI: 10.1038/s43588-021-00184-y.

[296] M. Schuster and K.K. Paliwal, "Bidirectional recurrent neural networks", in *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673-2681, 1997. DOI: 10.1109/78.650093.

[297] Scikit-Optimize - library for bayesian optimization, `https://scikit-optimize.github.io`. Last accessed: 8.01.2023.

[298] B. Settles, "Active Learning Literature Survey", Computer Sciences Technical Report 1648, 2009.

[299] J.G. Shanahan and L. Dai, "Large Scale Distributed Data Science using Apache Spark", *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '15), Association for Computing Machinery, New York, NY, USA*, pp. 2323–2324, 2015. DOI: 10.1145/2783258.2789993.

[300] L.S. Shapley, "Notes on the n-person game – ii: The value of an n-person game", *RAND RM*, vol. 670, 1951.

[301] J. Shi, O. Alagoz, F.S. Erenay, and Q. Su, "A survey of optimization models on cancer chemotherapy treatment planning", *Ann Oper Res*, vol. 221, pp. 331–356, 2014. DOI: 10.1007/s10479-011-0869-4.

[302] M. Shi, L. Lv, W. Sun, and X. Song, "A multi-fidelity surrogate model based on support vector regression", *Struct Multidisc Optim*, vol. 61, pp. 2363–2375, 2020. DOI: 10.1007/s00158-020-02522-6.

[303] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge Computing: Vision and Challenges", *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637-646, 2016. DOI: 10.1109/JIOT.2016.2579198.

[304] Y. Shi and R.C. Eberhart, "Parameter selection in particle swarm optimization", *Proceedings of Evolutionary Programming VII*, vol. 1447, pp. 591–600, 1998. DOI: 10.1007/BFb0040810.

[305] A.F. Siegenfeld and Y. Bar-Yam, "An Introduction to Complex Systems Science and Its Applications", *Complexity*, vol. 2020, Article ID 6105872, 2020. DOI: 10.1155/2020/6105872.

[306] D. Silver, A. Huang, C.J. Maddison, A. Guez, L. Sifre, G. van den Driessche, et al., "Mastering the game of Go with deep neural networks and tree search", *Nature*, vol. 529, pp. 484–503, 2016. DOI: 10.1038/nature16961.

[307] H.A. Simon, "The Architecture of Complexity", *Proceedings of the American Philosophical Society*, vol. 106, no. 6, pp. 467-482, 1962. DOI: 10.1007/978-1-4899-0718-9_31.

[308] H. Situngkir, "On selfish memes: culture as complex adaptive system", *Journal of Social Complexity*, vol. 2, no. 1, pp. 20-–32, 2004. DOI: 10.1177/135050768101200107.

[309] Z. Skolicki and K. De Jong, "Improving Evolutionary Algorithms with Multi-representation Island Models", *Parallel Problem Solving from*

Nature - PPSN VIII, Lecture Notes in Computer Science, vol. 3242, Springer, Berlin, Heidelberg, 2004. DOI: 10.1007/978-3-540-30217-9_43.

[310] Ł. Skowronek, P. Gora, M. Możejko, and A. Klemenko, "Graph-based sparse neural networks for traffic signal optimization", in *Proceedings of the 29th International Workshop on Concurrency, Specification and Programming (CS&P 2021)*, pp. 145–155, 2021.

[311] Ł. Skowronek, P. Gora, M. Możejko, and A. Klemenko, "Graph-Based Sparse Neural Networks for Traffic Signal Optimization", *Schlingloff, BH., Vogel, T., Skowron, A. (eds) Concurrency, Specification and Programming. Studies in Computational Intelligence*, vol. 1091. Springer, Cham., 2023. DOI: 10.1007/978-3-031-26651-5_9.

[312] SmartCity Lab in Chełm, announcement, `https://www.itspolska.pl/2023/03/07/pierwsze-w-polsce-smartcity-lab-na-lubelszczyznie`. Last accessed: 12.06.2023.

[313] J. Song, Y. Wu, Z. Xu, X. Lin, "Research on car-following model based on SUMO", *The 7th IEEE/International Conference on Advanced Infocomm Technology*, Fuzhou, China, pp. 47-55, 2014. DOI: 10.1109/ICAIT.2014.7019528.

[314] K. Sörensen and F.W. Glover, "Metaheuristics", *Encyclopedia of Operations Research and Management Science*, Springer, 2013. DOI: 10.1007/978-1-4419-1153-7_1167.

[315] J.M. Springer and G.T. Kenyon, "It's Hard for Neural Networks to Learn the Game of Life", *2021 International Joint Conference on Neural Networks*, pp. 1-8, 2021, DOI: 10.1109/IJCNN52387.2021.9534060.

[316] N. Srivastava, G.E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting", *Journal of Machine Learning Research*, voll. 15, no. 1, pp. 1929-1958, 2014.

[317] K. Stanková, J.S. Brown, W.S. Dalton, and R.A. Gatenby, "Optimizing Cancer Treatment Using Game Theory: A Review", *JAMA Oncol*, vol. 5, no. 1, pp. 96-103, 2019. DOI: 10.1001/jamaoncol.2018.3395.

[318] T.E. Stuart, J.A. Slater, R. Colbeck, R. Renner, and W. Tittel, "Experimental Bound on the Maximum Predictive Power of Physical Theories", *Phys. Rev. Lett*, vol. 109, no. 2-13, 020402, 2012. DOI: 10.1103/PhysRevLett.109.020402.

[319] B. Suman and P. Kumar, "A survey of simulated annealing as a tool for single and multiobjective optimization", *Journal of the Operational Research Society*, vol. 57, 2006. DOI: 10.1057/palgrave.jors.2602068.

[320] B. Sun, J. Appiah, and B.B. Park, "Practical guidance for using mesoscopic simulation tools", *Transportation Research Procedia*, vol. 48, pp. 764-776, 2020. DOI: 10.1016/j.trpro.2020.08.078.

[321] R.S. Sutton and A.G. Barto, "Reinforcement Learning: An Introduction", *IEEE Transactions on Neural Networks*, vol. 9, no. 5, pp. 1054-1054, 1998, DOI: 10.1109/TNN.1998.712192.

[322] W. Szejgis, A. Warno, and P. Gora, "Predicting times of waiting on red signals using BERT", presented at *NeurIPS 2020 Workshop on Machine Learning for Autonomous Driving*, 2020. arXiv: 2102.12896.

[323] R. Tachet, P. Santi, S. Sobolevsky, L. Reyes-Castro, E. Frazzoli, D. Helbing, and C. Ratti, "Revisiting Street Intersections using Slot-Based Systems", *PLOS ONE*, vol. 11, no. 3, 2016. DOI: 10.1371/journal.pone.0149607.

[324] M.K. Tan, H.S.E. Chuo, R.K.Y. Chin, et al., "Optimization of traffic network signal timing using decentralized genetic algorithm", *2017 IEEE 2nd international conference on automatic control and intelligent systems, I2CACIS 2017*, pp. 62–67, 2017. DOI: 10.1109/I2CACIS.2017.8239034.

[325] TensorCell, the official group's website, `https://www.tensorcell.com`. Last accessed: 12.06.2023.

[326] TensorFlow project, official website `https://www.tensorflow.org`. Last accessed: 12.06.2023.

[327] M.K. Thompson, P. Poortmans, A.J. Chalmers, C. Faivre-Finn, E. Hall, R.A. Huddart, et al., "Practice-changing radiation therapy trials for the treatment of cancer: where are we 150 years after the birth of Marie Curie?", *British Journal of Cancer*, vol. 119, no. 4, pp. 389–407, 2018. DOI: 10.1038/s41416-018-0201-z.

[328] Torch framework, official website, `http://torch.ch`. Last accessed: 29.12.2022.

[329] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, et al., "LLaMA: Open and Efficient Foundation Language Models", arXiv:2302.13971, 2023.

[330] M.Q. Tran, M. Baharudin, and E. Kamioka, "MC-TES: An Efficient Mobile Phone Based Context-Aware Traffic State Estimation Framework", *Journal of Information Processing*, vol. 21. pp. 76-89, 2011. DOI: 10.2197/ipsjjip.21.76.

[331] TRANSIMS, traffic simulation tool, official website, `https://code.google.com/archive/p/transims`. Last accessed: 12.06.2023.

[332] Transportation Research Board, "Adaptive Traffic Control Systems: Domestic and Foreign State of Practice", NCHRP SYNTHESIS 403, Transportation Research Board, 2010.

[333] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations", *Physical Review E*, vol. 62, no. 2, pp. 1805–1824, 2000. DOI: 10.1103/PhysRevE.62.1805.

[334] UK Highways Agency, *"Design Manual for Roads and Bridges"*, vol. 12, section 2, 2005.

[335] S. Ulaganathan, I. Couckuyt, T. Dhaene, J. Degroote, and E. Laermans, "Performance study of gradient-enhanced Kriging", *Aerospace Science and Technology*, vol. 25, no. 1, pp. 177–189, 2016. DOI: 10.1007/s00366-015-0397-y.

[336] M. Vallati, D. Magazzeni, B. De Schutter, L. Chrpa, and T. McCluskey, "Efficient Macroscopic Urban Traffic Models for Reducing Congestion: A PDDL+ Planning Approach", *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016. DOI: 10.1609/aaai.v30i1.10399.

[337] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, N.A. Gomez, ŁKaiser, and I. Polosukhin, "Attention is All you Need", *Advances in Neural Information Processing Systems*, vol. 30, pp. 5998-6008, 2017.

[338] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio Y., "Graph Attention Networks", *International Conference on Learning Representations 2018*, 2017.

[339] Verkehr.nrw, traffic information system, `https://www.verkehr.nrw`. Last accessed: 12.06.2023.

[340] O. Vinyals, I. Babuschkin, W.M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, et al., "Grandmaster level in StarCraft II using multi-agent reinforcement learning", *Nature*, vol. 575, pp. 350–354, 2019. DOI: 10.1038/s41586-019-1724-z.

[341] Vision Zero Network, "What is Vision Zero?", `https://visionzeronetwork.org/about/what-is-vision-zero`. Last accessed: 12.06.2023.

[342] C. Voudouris, "Guided local search for combinatorial optimisation problems", PhD Thesis, Department of Computer Science, University of Essex, Colchester, UK, 1997.

[343] M. Wang, L. Wu, M. Li, D. Wu, X. Shi, and C. Ma, *"Meta-learning based spatial-temporal graph attention network for traffic signal control"*, Knowledge-Based Systems, vol. 250, 109166, 2022. DOI: 10.1016/j.knosys.2022.109166.

[344] Y. Wang, L. Li, J. Ni, and S. Huang, "Feature selection using Tabu search with long-term memories and probabilistic neural networks", *Pattern Recognition Letters*, vol. 30, no. 7, pp. 661–670, 2009. DOI: 10.1016/j.patrec.2009.02.001.

[345] P. Wasilewski and P. Gora, "Traffic-related Knowledge Acquired by Interaction with Experts", *Proceedings of the Workshop Concurrency, Specification & Programming'2014*, Chemnitz, Sept. 29 - Oct. 1, 2014, Informatik-Berichte der HUB Nr. 245, pp. 269-280, 2014.

[346] J. Wang, Q. Zhang, D. Zhao, and Y. Chen, "Lane Change Decision-making through Deep Reinforcement Learning with Rule-based Constraints", *2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1-6, 2019. DOI: 10.1109/IJCNN.2019.8852110.

[347] J.G. Wardrop and J.I. Whitehead, "Correspondence. Some Theoretical Aspects of Road Traffic Research", *ICE Proceedings: Engineering Divisions*, vol. 1, no. 5, pp. 767-768, 1952. DOI: 10.1680/ipeds.1952.11362.

[348] A. Warno, "Approximation of complex simulations using deep neural networks on example of tumour simulation", M.Sc. thesis at the University of Warsaw, 2022.

[349] W. Weaver, "Science and complexity", *American Scientist*, vol. 36, no. 4, pp. 536–544, 1948.

[350] H. Wei, N. Xu, H. Zhang, G. Zheng, X. Zang, X. Chen, et al., "CoLight: Learning Network-level Cooperation for Traffic Signal Control", *Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM '19)*, pp. 1913–1922, 2019. DOI: 10.1145/3357384.3357902.

[351] G. Weng, U.S. Bhalla, and R. Iyengar, "Complexity in biological signaling systems", *Science*, vol. no. 5411, 284, 1999, pp. 92-96. DOi: 10.1126/science.284.5411.92.

[352] G.M. Whitesides, R.F. Ismagilov, "Complexity in chemistry", *Science*, vol. 284, no. 5411, pp. 89-92, 1999. DOI: 10.1126/science.284.5411.89.

[353] R. Wiedemann, "Simulation des Strassenverkehrsflusses" (in German), *Schriftenreihe des Instituts für Verkehrswesen der Universität Karlsruhe*, Band 8, Karlsruhe, Germany. 1974.

[354] A. Williams, "Exploring the use of advanced traffic information system to manage traffic congestion in developing countries", *Scientific African*, vol. 4, 2019. DOI: 10.1016/j.sciaf.2019.e00079.

[355] M. Wojnarski, P. Gora, M. Szczuka, H. S. Nguyen, J. Świetlicka, and D. Zeinalipour, "IEEE ICDM 2010 Contest TomTom Traffic Prediction for Intelligent GPS Navigation", *IEEE ICDM 2010 Workshops*, pp. 1372-1376, 2010. DOI: 10.1109/ICDMW.2010.51.

[356] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, et al., *"HuggingFace's Transformers: State-of-the-art Natural Language Processing"*, arXiv:1910.03771, 2020.

[357] S. Wolfram, "A new kind of science", Wolfram Media, ISBN: 1-57955-008-8, 2002.

[358] World Health Organization, "Cancer", `https://www.who.int/news-room/fact-sheets/detail/cancer`. Last accessed: 21.11.2022.

[359] World Health Organization, "Road traffic injuries", `http://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries`. Last accessed: 12.06.2023.

[360] Z.-G. Wu, H. Wang, J. Liu, J. Zhi, and C. Fu, "The Improvement of Quantum Genetic Algorithm and Its Application on Function Optimization", *Mathematical Problems in Engineering*, Hindawi, article ID 730749, 2013. DOI: 10.1155/2013/730749.

[361] N. Wulff and J.A. Hertz, "Learning cellular automaton dynamics with neural networks", *Advances in Neural Information Processing Systems*, vol. 5, pp. 631–638, 1992.

[362] X.-S. Yang, "Nature-Inspired Optimization Algorithms", 2014. DOI: 10.1016/C2013-0-01368-0.

[363] P. Ye, "A Review on Surrogate-Based Global Optimization Methods for Computationally Expensive Functions", *Software Engineering*, vol. 7, no. 4, pp. 68-84, 2019. DOI: 10.11648/j.se.20190704.11.

[364] I. Yperman, S. Logghe S., and L.H. Immers, "The Link Transmission Model: An efficient implementation of the kinematic wave theory in traffic networks", *Proceedings of the 10th EWGT Meeting*, Poznań, Poland, 2005.

[365] ZDM (Road Authorities in Warsaw), Information about traffic on the roads of the basic system of the capital city of Warsaw according to Automatic Traffic Measurements, `https://zdm.waw.pl/dzialania/badania-i-analizy/analiza-ruchu-na-drogach` (in Polish). Last accessed: 12.06.2023.

[366] ZDM (Road Authorities in Warsaw), Information on the number of intersections in Warsaw, `https://zdm.waw.pl/aktualnosci/nowe-zastapi-stare-remontujemy-sygnalizacje-swietlne` (in Polish). Last accessed: 12.06.2022.

[367] L. Zhang and D. Levinson, "An Agent-Based Approach to Travel Demand Modeling: An Exploratory Analysis", *Transportation Research Record: Journal of the Transportation Research Board*, pp. 28–38, 2004. DOI: 10.3141/1898-04.

[368] S. Zhao, X. Lu, and X. Li, "Particle Swarm Optimization with Time Varying Parameters for Scheduling in Cloud Computing", *MATEC Web of Conferences*, vol. 28, 2015. DOI: 10.1051/matecconf/20152806001.

[369] D.-X. Zhou, "Universality of deep convolutional neural networks", *Applied and Computational Harmonic Analysis*, vol. 48, no. 2, pp. 787–794, 2020. DOI: 10.1016/j.acha.2019.06.004.

[370] Z. Zhou and M. Li, "Targeted therapies for cancer", *BMC Med*, vol. 20, no. 1, 90, 2022. DOI: 10.1186/s12916-022-02287-3.

[371] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H, Zhu, et al., "A Comprehensive Survey on Transfer Learning", *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43-76, 2021, DOI: 10.1109/JPROC.2020.3004555.