electronics

Advances in Public Transport Platform for the Development of Sustainability Cities

Edited by
Juan M. Corchado, Josep L. Larriba-Pey, Pablo Chamoso and Fernando De la Prieta

Printed Edition of the Special Issue Published in *Electronics*

www.mdpi.com/journal/electronics

MDPI

Advances in Public Transport Platform for the Development of Sustainability Cities

Advances in Public Transport Platform for the Development of Sustainability Cities

Editors

Juan M. Corchado
Josep L. Larriba-Pey
Pablo Chamoso
Fernando De la Prieta

MDPI • Basel • Beijing • Wuhan • Barcelona • Belgrade • Manchester • Tokyo • Cluj • Tianjin

Editors

Juan M. Corchado
BISITE Research Group
University of Salamanca
Salamanca
Spain

Josep L. Larriba-Pey
Data Maagement Group
Polytechnic University of Catalonia
Barcelona
Spain

Pablo Chamoso
BISITE Research Group
University of Salamanca
Salamanca
Spain

Fernando De la Prieta
BISITE Research Group
University of Salamanca
Salamanca
Spain

Editorial Office
MDPI
St. Alban-Anlage 66
4052 Basel, Switzerland

This is a reprint of articles from the Special Issue published online in the open access journal *Electronics* (ISSN 2079-9292) (available at: www.mdpi.com/journal/electronics/special_issues/transport_platform_cities).

For citation purposes, cite each article independently as indicated on the article page online and as indicated below:

LastName, A.A.; LastName, B.B.; LastName, C.C. Article Title. *Journal Name* **Year**, *Volume Number*, Page Range.

ISBN 978-3-0365-3980-5 (Hbk)
ISBN 978-3-0365-3979-9 (PDF)

Contents

About the Editors

Juan M. Corchado

Juan Manuel Corchado, Full Professor with Chair at the University of Salamanca. He was Vice President for Research from 2013 to 2017 and the Director of the Science Park of the University of Salamanca. Chosen twice as the Dean of the Faculty of Science, he holds a PhD in Computer Sciences from the University of Salamanca and a PhD in Artificial Intelligence from the University of the West of Scotland. He is the director of a renowned research group called BISITE (Bioinformatics, Intelligent Systems and Educational Technology), which was created in the year 2000.

J.M Corchado is also the Director of the IOT Digital Innovation Hub and the President of the AIR Institute. He has been a Visiting Professor at the Osaka Institute of Technology since January 2015, and has been a Visiting Professor at the Universiti Malaysia Kelantan and a Member of the Advisory group on Online Terrorist Propaganda of the European Counter Terrorism Centre (EUROPOL).

He currently combines all his activities with the direction of Master's Programs in Master in Security, Digital Animation, Multiplatform Mobile Application Development, Digital Intelligence, Financial Technology, Knowledge Transfer and R&D Management, Digital Transformation, ICT, Internet of Things, Social Media, 3D Printing, Blockchain, Industry 4.0 y Smart Cities and Intelligent Buildings, at the University of Salamanca, and with his work as Editor-in-Chief of Journals such as *ADCAIJ* (Advances in Distributed Computing and Artificial Intelligence Journal), *OJCST* (Oriental Journal of Computer Science and Technology) or *Electronics* MDPI (Computer Science and Engineering section).

J.M. Corchado works on projects in the fields of Artificial Intelligence, Machine Learning, Blockchain, IoT, Fog Computing, Edge Computing, Smart Cities, Smart Grids, Sentiment Analysis, etc.

Josep L. Larriba-Pey

Founder of the DAMA-UPC research group within BarcelonaTech and director over the past 15 years. This group deals with the management of big data and has a strong relationship with industry, both at local and international levels, with strong players such as IBM or Oracle.

Founder of Sparsity Technologies, BarcelonaTech spin-out, and CEO for the past 10 years. The company has been recognized by the European Commission as the SME with most innovative potential in the European Community in 2015.

Sparsity's star product, Sparksee, is a Graph Database used in Mobile and Embedded systems as a unique, small software and data footprint, persistent, high-performance and reliable device for the storage and analysis of large amounts of linked data.

Sparsity has also created CIGO!, a mobility management platform to plan routes, interact with fleets through mobile devices and analyses geolocated data without the need for sensors, intensive server use or expensive hardware. What makes CIGO! different from other route planning and fleet management systems is:

- Offline GPS tracking technology.
- Data integration from potentially any source: Open Data, Mobile Apps, private city or company data, sensors, etc.
- Optimal route calculation according to customized criteria (not just shortest path).

Scenarios where CIGO! makes the difference: tow away services, tourism, taxi, home medical services, police management, cycling safe and healthy, public transport, electric vehicle, delivery areas management.

Pablo Chamoso

Pablo Chamoso Santos holds a PhD in Computer Engineering from the University of Salamanca. He obtained the titles of Technical Engineer in Computer Systems, Computer Engineer (i3 Award for the best final project of Castilla y León) and Official Master' s Degree in Intelligent Systems. In addition to these official degrees, he holds a Master's Degree in Systems Development for Electronic Commerce and a Master's Degree in Information and Communication Systems Management. Currently, he is an Associate Professor at the Department of Computer Science of the University of Salamanca. His PhD thesis focused on the Development of Platforms for the Deployment and Management of Smart Cities. This research is co-funded by the Junta de Castilla y León and the European Social Fund through the program "Excellent Science and Technological Leadership". He has been a research member of the renowned research group BISITE since March 2011, and is collaborator at IBSAL, member of the University Institute of Research in Art and Technology of Animation, member of the Institute of Electrical and Electronics Engineers (IEEE), member of the IEEE Smart Cities Community, member of the Professional Association of Computer Engineers of Castilla y León, member of the University Institute of Research and Art and Technology of Animation and secretary of the academic and quality committees of the PhD Program in Energy and Marine Propulsion at the University of Salamanca.

Fernando De la Prieta

Fernando de la Prieta Pintado is an Associate Professor at the University of Salamanca Department of Computer Science and Automation.

Dr. De la Prieta is equally well experienced in research and teaching. Over recent years, he has followed a clearly defined line of research, focusing on the integration of multi-agent organizations, machine learning and advanced architectures in different fields. He applied the results in both his doctoral thesis (for which he obtained an international PhD mention and an extraordinary PhD award) and in the projects he has been involved in. He has more than 50 publications in international journals, many of which have a JCR impact factor on the Web of Science database. His H index in Google Scholar is 27. Furthermore, he has published more than 100 articles in books and in the proceedings of prestigious international conferences, and around thirty of these publications have been published in conferences indexed according to the CORE ranking. He has worked on more than 90 research projects (16 of them were international and in several he has been the principal investigator). In addition, he has participated in more than 30 research contracts (Art. 83), in some of them as the principal investigator. As a result of his work, around 40 intellectual properties have been registered. He has had several stays abroad (pre- and post-doctoral) in Portugal, Japan and South Korea. He has also taken an active part in the organization of international conferences, some of them included in the CORE ranking: IEEE-GLOBECOM (core B), ICCBR (Core B), CEDI, PAAMS (core C), ACM-SAC (core B), IEEE-FUSION (core C), and others.

Preface to "Advances in Public Transport Platform for the Development of Sustainability Cities"

Modern societies demand high and varied mobility, which in turn requires a complex transport system adapted to social needs that guarantees the movement of people and goods in an economically efficient and safe way, but all are subject to a new environmental rationality and the new logic of the paradigm of sustainability. From this perspective, an efficient and flexible transport system that provides intelligent and sustainable mobility patterns is essential to our economy and our quality of life. The current transport system poses growing and significant challenges for the environment, human health, and sustainability, while current mobility schemes have focused much more on the private vehicle that has conditioned both the lifestyles of citizens and cities, as well as urban and territorial sustainability.

Transport has a very considerable weight in the framework of sustainable development due to environmental pressures, associated social and economic effects, and interrelations with other sectors. The continuous growth that this sector has experienced over the last few years and its foreseeable increase, even considering the change in trends due to the current situation of generalized crisis, make the challenge of sustainable transport a strategic priority at local, national, European, and global levels.

This Special Issue will pay attention to all those research approaches focused on the relationship between evolution in the area of transport with a high incidence in the environment from the perspective of efficiency, which has become one of the neuralgic centers of sustainability. This relates to producing, consuming, and moving people and goods better, with fewer resources and less environmental impact.

Juan M. Corchado, Josep L. Larriba-Pey, Pablo Chamoso, and Fernando De la Prieta

Editors

electronics

MDPI

Editorial

Advances in Public Transport Platform for the Development of Sustainability Cities

Juan M. Corchado [1], Josep L. Larriba-Pey [2], Pablo Chamoso-Santos [1] and Fernando De la Prieta Pintado [1,*]

[1] BISITE Research Group, University of Salamanca, Edificio Multiusos I + D + I, 37007 Salamanca, Spain; corchado@usal.es (J.M.C.); chamoso@usal.es (P.C.-S.)
[2] Data Management Group, Polytechnic University of Catalonia, 08034 Barcelona, Spain; larri@ac.upc.edu
* Correspondence: fer@usal.es; Tel.: +34-677-522-678

Citation: Corchado, J.M.; Larriba-Pey, J.L.; Chamoso-Santos, P.; De la Prieta Pintado, F. Advances in Public Transport Platform for the Development of Sustainability Cities. *Electronics* **2021**, *10*, 2771. https://doi.org/10.3390/electronics10222771

Received: 4 November 2021
Accepted: 11 November 2021
Published: 12 November 2021

1. Introduction

There is high and varied mobility in modern societies which requires a complex transport system that adapts to social needs and guarantees the movement of people and goods in an economically efficient and safe way. All this designed from the new perspective of environmental wellness and of the sustainability paradigm. From this viewpoint, an efficient and flexible transport system that provides intelligent and sustainable mobility patterns is essential to our economy and quality of life. The current transport system poses growing and significant challenges for the environment, human health, and sustainability. Existent mobility schemes focus excessively on the use of private vehicles which have conditioned the lifestyle of citizens in cities, as well as urban and territorial sustainability.

Transport is an important element of the sustainable development framework due to the growing environmental strain, the associated social and economic effects, and its interconnection with other sectors. The continuous growth that this sector has experienced over the last few years and its foreseeable future growth, even considering the change of trend caused by the current situation of generalized crisis, make the challenge of sustainable transport a strategic priority at local, national, European, and global levels.

2. The Present Issue

This special issue consists of sixteen papers covering important topics in the field of public transportation under the framework of smart cities.

The research community is now turning its attention to different areas such as optimization and prediction [1–5]. As evidenced in references [2,3,5], which have analyzed travel time data to evaluate the performance of a public transport system. Others have focused on the demand for different modes of transportation and interaction among them, including a proposal for minimizing the passengers' waiting times and maximizing the vehicles' occupancy ratios. The use of unmanned aerial vehicles for emergency situations is extensively described in [1] for search and rescue operations, surveillance, disaster monitoring, response to terrorist attacks. Finally, ref. [4] studied the influence of the economy on transportation systems.

Recommender Systems are also commonly used within the framework of transportation for sustainable cities. Hence, references [6,7] focused on offering improved usability and services based on multi-modal door-to-door passenger experiences to increase engagement. Other examples can be found in reference [8], where recommendation systems are designed to improve the passengers' experience and the drivers' profit. Finally, other approaches focused on educating the general public about this topic [9].

Other topics included in this special issue are energy consumption forecasting in sustainable cities [10] as well as the analysis of energy trading and the development of a trust model [11]. Security is also an important issue within public transpormation, in reference [12] the secure management of railway transportation systems has been analyzed.

Finally, analytical models using Machine Learning and Deep Learning have been explored as part of this special issue [13,14]. Also, two case studies, carried out in the city of Barcelona, Spain [15] and Taipei, Taiwan [15], have been described.

3. Conclusions

This special issue has paid attention to all the research approaches that focus on the relationship between the evolution of transportation and the new perspective of achieving environmental wellness and efficiency, which has become one of the cornerstones of sustainability. It revolves around producing, consuming, and transporting people and goods better, while using up fewer resources and having lower environmental impact.

Author Contributions: J.M.C., J.L.L.-P., P.C.-S. and F.D.l.P.P. worked together in the whole editorial process of the special issue, "Advances in Public Transport Platform for the Development of Sustainability Cities", published by journal Electronics. F.D.l.P.P. drafted this editorial summary. J.M.C., J.L.L.-P. and P.C.-S. reviewed, edited, and finalized the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by "Ministerio de Ciencia, Innovación y Universidades. Proyectos I + D + i «RETOS INVESTIGACIÓN» del Programa Estatal de I+D+i orientada a los retos de la sociedad", grant number RTI2018-095390-B-C32. This research was also funded by the Shift2Rail Joint Undertaking under the Europeans Union's Horizon 2020 Research and Innovation Programme, grant number 777640.

Acknowledgments: First of all, we would like to thank all researchers who submitted articles to this special issue for their excellent contributions. We are also grateful to all the reviewers who helped in the evaluation of the manuscripts and made very valuable suggestions to improve the quality of the contributions. We would like to acknowledge the editorial board of Electronics, who invited us to guest edit this special issue. We are also grateful to the Electronics Editorial Office staff who worked thoroughly to maintain the rigorous peer-review schedule and timely publication.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Saha, S.; Vasegaard, A.E.; Nielsen, I.; Hapka, A.; Budzisz, H. UAVs Path Planning under a Bi-Objective Optimization Framework for Smart Cities. *Electronics* **2021**, *10*, 1193. [CrossRef]
2. Hartmann Tolić, I.; Nyarko, E.K.; Ceder, A. Optimization of Public Transport Services to Minimize Passengers' Waiting Times and Maximize Vehicles' Occupancy Ratios. *Electronics* **2020**, *9*, 360. [CrossRef]
3. Martin, L.; Wittmann, M.; Li, X. The Influence of Public Transport Delays on Mobility on Demand Services. *Electronics* **2021**, *10*, 379. [CrossRef]
4. Sousa, R.; Lima, T.; Abelha, A.; Machado, J. Hierarchical Temporal Memory Theory Approach to Stock Market Time Series Forecasting. *Electronics* **2021**, *10*, 1630. [CrossRef]
5. Yuan, Y.; Shao, C.; Cao, Z.; He, Z.; Zhu, C.; Wang, Y.; Jang, V. Bus Dynamic Travel Time Prediction: Using a Deep Feature Extraction Framework Based on RNN and DNN. *Electronics* **2020**, *9*, 1876. [CrossRef]
6. García-Retuerta, D.; Rivas, A.; Guisado-Gámez, J.; Antoniou, E.; Chamoso, P. Reputation System for Increased Engagement in Public Transport Oriented-Applications. *Electronics* **2021**, *10*, 1070. [CrossRef]
7. Rivas, A.; González-Briones, A.; Cea-Morán, J.J.; Prat-Pérez, A.; Corchado, J.M. My-Trac: System for Recommendation of Points of Interest on the Basis of Twitter Profiles. *Electronics* **2021**, *10*, 1263. [CrossRef]
8. Wan, X.; Ghazzai, H.; Massoud, Y. A Generic Data-Driven Recommendation System for Large-Scale Regular and Ride-Hailing Taxi Services. *Electronics* **2020**, *9*, 648. [CrossRef]
9. Jordán, J.; Valero, S.; Turró, C.; Botti, V. Using a Hybrid Recommending System for Learning Videos in Flipped Classrooms and MOOCs. *Electronics* **2021**, *10*, 1226. [CrossRef]
10. Oliveira, P.; Fernandes, B.; Analide, C.; Novais, P. Forecasting Energy Consumption of Wastewater Treatment Plants with a Transfer Learning Approach for Sustainable Cities. *Electronics* **2021**, *10*, 1149. [CrossRef]
11. Andrade, R.; Wannous, S.; Pinto, T.; Praça, I. Extending a Trust model for Energy Trading with Cyber-Attack Detection. *Electronics* **2021**, *10*, 1975. [CrossRef]
12. Hatzivasilis, G.; Fysarakis, K.; Ioannidis, S.; Hatzakis, I.; Vardakis, G.; Papadakis, N.; Spanoudakis, G. SPD-Safe: Secure Administration of Railway Intelligent Transportation Systems. *Electronics* **2021**, *10*, 92. [CrossRef]
13. Taboada, G.L.; Han, L. Exploratory Data Analysis and Data Envelopment Analysis of Urban Rail Transit. *Electronics* **2020**, *9*, 1270. [CrossRef]

14. Hernández-Nieves, E.; Parra-Domínguez, J.; Chamoso, P.; Rodríguez-González, S.; Corchado, J.M. A Data Mining and Analysis Platform for Investment Recommendations. *Electronics* **2021**, *10*, 859. [CrossRef]
15. Mariñas-Collado, I.; Frutos Bernal, E.; Santos Martin, M.T.; Martín del Rey, A.; Casado Vara, R.; Gil-González, A.B. A Mathematical Study of Barcelona Metro Network. *Electronics* **2021**, *10*, 557. [CrossRef]

electronics

MDPI

Article

UAVs Path Planning under a Bi-Objective Optimization Framework for Smart Cities

Subrata Saha [1], Alex Elkjær Vasegaard [1], Izabela Nielsen [1,*], Aneta Hapka [2], Henryk Budzisz [2]

[1] Department of Materials and Production, Aalborg University, Fibigerstræde 16, DK 9220 Aalborg, Denmark; saha@m-tech.aau.dk (S.S.); aev@mp.aau.dk (A.E.V.)
[2] Faculty of Electronics and Computer Science, Koszalin University of Technology, 75-343 Koszalin, Poland; aneta.hapka@tu.koszalin.pl (A.H.); henryk.budzisz@tu.koszalin.pl (H.B.)
* Correspondence: izabela@mp.aau.dk

Abstract: Unmanned aerial vehicles (UAVs) have been used extensively for search and rescue operations, surveillance, disaster monitoring, attacking terrorists, etc. due to their growing advantages of low-cost, high maneuverability, and easy deployability. This study proposes a mixed-integer programming model under a multi-objective optimization framework to design trajectories that enable a set of UAVs to execute surveillance tasks. The first objective maximizes the cumulative probability of target detection to aim for mission planning success. The second objective ensures minimization of cumulative path length to provide a higher resource utilization goal. A two-step variable neighborhood search (VNS) algorithm is offered, which addresses the combinatorial optimization issue for determining the near-optimal sequence for cell visiting to reach the target. Numerical experiments and simulation results are evaluated in numerous benchmark instances. Results demonstrate that the proposed approach can favorably support practical deployability purposes.

Keywords: unmanned aerial vehicles (UAVs); multi-objective optimization; integer programming; GLPK; variable neighborhood search; search and rescue

Citation: Saha, S.; Vasegaard, A.I.; Nielsen, I.; Hapka, A.; Budzisz, H. UAVs Path Planning under a Bi-Objective Optimization Framework for Smart Cities. *Electronics* **2021**, *10*, 1193. https://doi.org/10.3390/electronics10101193

Academic Editors: Juan M. Corchado, Josep L. Larriba-Pey, Pablo Chamoso and Fernando De la Prieta

Received: 16 March 2021
Accepted: 13 May 2021
Published: 17 May 2021

1. Introduction

The path planning problem for a set of Unmanned Aerial Vehicles (UAVs) has gained unprecedented interest from researchers and practitioners to develop intelligent systems and execute various tasks with minimum human intervention. With upgraded components such as cameras, sensors, or telemetry systems, UAV application is becoming an integral strategic part for emergency management; aerial photography; mountain rescue; smart farming; maritime search and rescue; information collection, post-disaster relief; homeland security, crowd management, etc. [1–3]. UAVs, in practice, has many significant advantages such as human workload reduction, high mobility, saving of valuable resources, etc. In the literature, the path planning problem is categorized in several ways according to problem characteristics. For example, according to the targets' reaction, one can classify the problem into two categories: one-sided vs. two-sided path planning problems. On the other hand, based on targets' motion, one can classify the situation as static vs. moving target search or open vs. closed-loop decision models based on the decision-making context [4–10].

In recent years, the utilization of UAVs has been becoming increasingly attractive in the context of Smart City Management solutions. Several key technologies are continuously integrated into smart cities operations, such as data collection and protection and intrusion detection technologies. In this regard, the application of UAVs to collect data or images is an economical and effective solution. UAVs operations can lead to a new paradigm for developing smart cities with a high-quality life and sustainable economic growth. For example, Felemban et al. [11] noted that UAVs could be used to detect the earlier signs of a stampede, congestion, and other crowd problems. The authors proposed a Priority-Based Routing Framework to increase the delivery speed of images during Hajj in Saudi

Arabia. Researchers found that UAVs can be helpful in policing systems to fight against crime [12]. It was reported that such UAV policing systems work well for extensive crime deterrence [13]. However, there are many challenges, and we highlight one of those where UAVs are deployed in search and rescue problems.

Due to the sequential decision-making nature, the fundamental search and rescue path planning problem is a non-deterministic polynomial-time problem (NP-hard) [14]. Therefore, researchers employ both exact algorithm and heuristic approaches alternatively to solve such complex decision-making problems. One can argue that the modern search theory originated from the pioneering works by the group of researchers, Stewart [8], Brown [15] and Benkoski et al. [16]. Researchers mainly focus on the allocation decision instead of the optimal sequential path generation. By assuming an exponential detection function, Stewart [8] formulated a network flow model to characterize a moving target detection problem and used the branch and bound method to find a near-optimal solution. Later, Eagle [4] formulated the model in a dynamic programming framework and utilized the Markov process to replicate target motion as state transitions. Washburn [17] made an effort to determine the best upper bound for a generalized path planning problem. After that, researchers progressively shifted their attention toward the evaluation of algorithm performance in more complex enshrinement [18]. However, the travel time in the earlier model was assumed as uniform. Lau et al. [19] relaxed this assumption and formulated a model where travel time among regions are non-uniform. Rogge and Aeyels [20] introduced the concept of a collaborative path planning problem where the search area consists of multiple moving targets with an arbitrary number of obstacles. Li et al. [21] studied energy-efficient rechargeable UAV deployment strategy to provide seamless coverage in urban areas and employed the two-stage particle swarm optimization (PSO) algorithm to solve the problem. Regarding other variants, Berger and Lo [22] introduced a mixed-integer programming model under a directed acyclic graph framework and used CPLEX software to find an optimal path. To overcome computational effort, Perez-Carabaza et al. [23] proposed a modified ant colony optimization (ACO) algorithm to investigate the nature of trajectories for a set of heterogeneous UAVs. Ye et al. [24] used an adaptive genetic algorithm (GA) to find the solution for a collaborative multiple task assignment problem with fixed-wing UAVs. The authors employed a robust encoding strategy to generate feasible chromosomes. Lu et al. [25] use the wolf pack algorithm (WPA) to solve the task assignment problem for UAVs. The authors found that WPA can outperform PSO and GA in terms of convergence speed and solution accuracy. Lou et al. [26] proposed a multi-swarm fruit fly optimization algorithm to find a solution for multi-UAV cooperative mission planning problem. However, Alhaqbani et al. [27] stated that a common problem in most of the metaheuristics is that those can perform poorly in regards to run time. More recently, Xiong et al. [28] introduced Voronoi-based Ant colony optimization algorithm combined with the Dijkstra's algorithm to investigate optimal trajectories. In recent years, various types of machine learning algorithms have been employed to obtain optimal deployment strategy, and we refer to the recent review works by [29] and [30] for detailed discussion in this aspect. In addition, we refer the following works for more discussion on path planning from various perspectives [31–37].

In this study, we use a modified Variable Neighborhood Search (VNS) meta-heuristic [38]. Since its inception, the algorithm has been employed in numerous fields such as network design problems in communication [39], facility location problem [40], data mining [41], timetabling and related manpower organization problems [42], single- and multi-objective job shop scheduling [43,44], vehicle routing problem [45] and bioinformatics [46] due to its user-friendliness, higher precision and robustness. The VNS systematically exploits the idea of neighborhood change iteratively to improve the initial solution inside the shaking and local search procedures [47,48]. Unlike other meta-heuristic approaches, parameter tuning is always an issue; the fundamental VNS algorithm and its extension version require few or, occasionally, no parameters. One significant advantage to the VNS-based approach for path planning is that it accommodates the path maneuverability through the path

constructor (see Algorithm 1) operator. At the same time, the inherent shaking procedure seeks to overcome the possible local optima. The algorithm then attempts to improve the randomly changed path to catch a more rewarded path than the incumbent solution.

The cited literature's main disadvantage is that most authors only studied the problem as a single-objective optimization problem, e.g., maximizing the probability of finding targets, minimizing the path length, equal utilization of resources, etc. However, in a time-constrained decision-making context, only considering one objective may not lead to an acceptable outcome [49,50]. From a practical point of view, it is essential to handle several objectives simultaneously to obtain a pragmatic solution. Explicitly, the two most fundamental goals that need to be considered are maximization of finding the targets and minimizing the path length objective that can ensure minimum utilization of resources and implicitly ensure less operational time and energy consumption. It is challenging to find the ideal solution due to the conflicting nature of objective functions; therefore, researchers have proposed different approaches such as weighted sum [51], global criterion [52], goal programming [53], multi-choice goal programming [54], non-dominated sorting genetic algorithm II [55], fuzzy-two phase approach [56], etc., and the issue of a specific method largely depends on the decision-makers. Note that UAV path planning is itself an NP-hard problem [57]; thus, we use a simple weighted sum approach in this study. This study formulated the model as binary linear programming (BLP) formulation under a bi-objective optimization environment and proposed a modified VNS algorithm to find the solution. Numerical experiments were conducted to validate the overall framework. The key contributions of the study are as follows: First, a bi-objective optimization problem is proposed to obtain paths for multiple UAVs in a time-constrained environment. Second, a modified VNS algorithm is proposed, which is highly parallelizable and straightforward to understand. Moreover, the simulation study reveals that it can provide a solution within a reasonable time when the exact solver fails to provide a solution, and the performance for the algorithm is always higher compared to Dijkstra's algorithm, which is extensively used by several researchers [58,59]. Finally, a sensitivity analysis on the weight-space provide an overview regarding the importance of multi-objective formulation in the practical implementation of UAVs.

The paper is organized as follows. The mathematical model and corresponding assumption and notation are presented in Section 2. In Section 3, an overview is presented for the data generation. The solution procedure for the model is described in Section 4. A detailed overview of the VNS algorithm is also presented in this section. Extensive numerical experiments and validation of the proposed solution framework's effectiveness are presented in Section 5. Finally, Section 6 concludes by highlighting findings, limitations and future research directions.

2. Mathematical Model

Path planning and trajectory mapping for UAV is an important topic because of the incredible versatility and flexibility of UAVs that allow them to be employed in different operations. Although path planning goes before trajectory mapping, fundamentally, their characteristics are not entirely distinct. If point-to-point trajectories are measured, the two problem needs to be solved simultaneously if the initial and final positions are specified. One can define the path planning problem as finding a collision-free motion within a specified environment where initial and final locations are pre-defined. In this study, we use the cell decomposition method. In this method, the entire search space is subdivided into several regions (equal/unequal), called cells. The corresponding path will represent a connected graph and describe the adjacent relations between cells. Simultaneously, the trajectory planning problem is based on the input generated by the path planner. To plan a trajectory, commonly, a sequence of waypoints needs to be extracted. A kinematic inversion needs to be performed based on some decision-maker criteria such as minimizing total execution time, energy, distance, jerk, etc. In the present formulation, we ignore the effect of the kinematics of the UAV. We assume that a team of homogeneous UAVs is searching stationary targets in a pre-defined search region [60].

The search area is divided into an $N \times N$ grid describing possible target locations. The time duration for each cell visit, with equal size, is assumed as constant. The cell occupancy probabilities are generated initially, and, as we assume the targets to be stationary and non-moving, we omit the dynamics of a changing probability map. To maneuver its neighboring cells, any UAV can move in eight different directions{E, W, N, S, SE, SW, NE, NW}. However, at the cell where the UAVs start maneuvering is located, the UAVs are also allowed to hover. This mimics the possibility of early landing or later departure for some UAVs. A graph theory-based directed acyclic network representation is employed to streamline the setup. The entire graph is defined as $G_t = (V_t, E_t)$ for all t in a given time horizon T, V_t, the set of vertices, represent all possible locations $n \in N^* = \{1, \ldots, N^2 - 1, N^2\}$ at time $t \in T$. E_t, the set of edges, represents all the possible state transition related to each UAV between episodes t and $t + 1$. An adjacency matrix A defines the connectivity of G, $A_{tn'n} = 1$ if $v_{tn'} \in V_t$ and $v_{t'n} \in V_{t+1}$ are connected, else $A_{tn'n} = 0$. Consequently, a binary decision variable x_{ntr} is introduced to represent the cells n traversed at the respective time period t for the respective rth UAV.

The following notations are used to formulate the mathematical model:

N	the entire search region is divided into $N \times N$ number of cells with equal area in the grid, $n \in N^* = \{1, .., N^2 - 1, N^2\}$
T	set of time intervals with equal length defining the time horizon to explore a grid, $t \in \{0, 1, \ldots, \lvert T\rvert - 1\}$
R	number of UAVs, $r \in \{1, \ldots, R\}$
p_n	probability of actual target occupancy on cell n
x_{ntr}	state transition binary variable; $x_{ntr} = 1$, if the path of rth UAV investigates the nth cell in time period t, while $x_{ntr} = 0$, if that the corresponding cell is not visited
$F_{n'ntr}$	a binary matrix representation of the infeasible maneuvers. That is, $F_{n'ntr} = 1$ whenever $A_{tn'n} = 0$
Z_{ntr}	a binary binary matrix representation of all cells through the time horizon representing the same location
B_{ntr}	a binary matrix representation of the cells that can only be visited once
H_{ntr}	a binary matrix representation of all maneuvers performed in the time period t
S_{ntr}	a binary matrix representation of start and ending positions for rth UAV

Based on the above notation, the following mathematical model is proposed, where the first objective represents the cumulative probability of success for the total number of UAVs to be deployed and the second objective minimizes the total spent time performing the mission:

$$\max \quad f_1 = \sum_{r \in R} \sum_{t \in T} \sum_{n \in N^*} p_n x_{ntr} \tag{1}$$

$$\min \quad f_2 = \sum_{r \in R} \sum_{t \in T} \sum_{n \in N^*} \frac{x_{ntr}}{RT} \tag{2}$$

s.t.

$$\sum_{t \in T} \sum_{n \in N^*} F_{n'ntr} x_{ntr} \leq 1 \quad \forall r \in R \quad \forall n' \in N^* \tag{3}$$

Constraint (3) ensures that infeasible maneuvers cannot be performed between two consecutive time periods. The binary matrix F showcases each pair between consecutive cells n and n' that are infeasible for a given time period t. That is, if $F_{n'ntr} = 1$, then the two cells n and n' in time period t and $t + 1$, respectively, are not feasible in the same path for any r.

$$\sum_{r \in R} Z_{ntr} x_{ntr} \leq 1 \quad \forall n \in N^* \quad t \in T \tag{4}$$

Constraint (4) enforces a safety zone around each path, that is, a single agent r can only traverse a cell in a given time period. Note that the binary matrix Z showcases the decision variable's index that represents the same time period.

$$\sum_{r \in R} \sum_{t \in T} B_{ntr} x_{ntr} \leq 1 \quad \forall n \in N^* \tag{5}$$

Here, constraint (5) considers gathering images of a cell over multiple different time periods, where the binary B matrix showcase each index that represents the same cell. In

this paper, we neglect the dynamics of changing probability, and we are not interested in obtaining a search path that acquires multiple images of the same cell. Note that we do not have to consider a conditional probability map that is dependent on the chosen paths because of this constraint, as the cumulative probability will be in the range of $[0, 1]$.

$$\sum_{n \in N^2} H_{ntr} x_{ntr} = 1 \quad \forall n \in N^* \quad r \in R \tag{6}$$

In constraint (6), the binary H matrix ensures that the paths only allow a single maneuver to be performed per time period per UAV.

$$\sum_{r \in R} \sum_{t \in T} S_{ntr} x_{ntr} = 1 \quad \forall r \in R \tag{7}$$

Constraint (7) ensures that the complete path starts and ends in the designated time zones in the designated time periods.

$$x_{ntr} \in \{0, 1\} \quad \forall n \in N^*, \forall t \in T, r \in \{1, \ldots, R\} \tag{8}$$

Finally, the above constraint (8) represent the decision and auxiliary variables.

3. Scenario Generation

The UAV-assisted SAR mission generally consists of multiple different phases, with the common goal of deploying as soon as possible when sufficient information about the mission is gathered. The UAV aspect is to either aid or collect information as fast as possible for the rescue team's job. In this research, the UAVs are only gathering information through images. Therefore, when generating the problem scenarios, we have to assume some information that later can be modified to accommodate real-world scenario. In general, the overall map is divided into an $N \times N$ grid where each cell is assumed to have the same area. Then, a probability map is generated where each cell is given a certain probability of containing the missing target. The probability map is generated randomly based on a given number of hotspots and corresponding spread (see Figure 1). To accommodate the problem scenario, the number of deployed UAVs also affects the size of the problem scenario. These are assumed to be taking off and landing in a specific grid cell. There is also denoted a time horizon with a given number of equidistant points in time, and the UAVs are then able to search an entire grid cell for each time period, and then go to one of their neighboring grid cells in the following time period. As mentioned in the Mathematical Modelling Section, the UAVs can move in all directions, but they can only hover (land) in the grid cell containing the UAV station. Note that this cell, therefore, should not have any gain or loss in terms of the objectives, e.g., probability of locating the target. Due to the problem complexity, we assume there to only be two hot spots with a spread of three and the UAV station to be located in grid cell $[0, 0]$. The parameters assumed to affect the size of the problem scenario are the grid size, N, time horizon, T, and number of UAVs, R.

Note that the proposed division of the search area is analogous to the raster model, which is a data storage method used extensively in geographic information systems.

4. Solution Procedure

In this section, we explain the solution procedure and the selection of search parameters for the employed search method. The exact approach is often not applicable in large-scale scenarios, as it can even fail to deliver a feasible solution. In a time-restricted environment such as UAV-assisted search and rescue, this is not applicable. On the other side of the spectrum, a greedy approach does deliver a feasible solution, but it often lacks in performance. This is what we try to investigate with the deployed VNS approach. We evaluate the performance of the algorithm with Dijkstra's algorithm and exact solvers such as GNU Linear Programming Kit (GLPK) to establish its efficiency. However, before doing so, the following definitions should be presented.

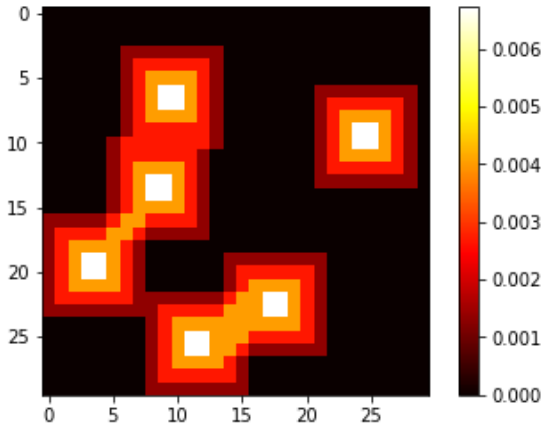

Figure 1. Probability map on a 30×30 grid given six hotspots with a spread of 3. The start and end cell (UAV station) is located in cell $[0, 0]$ in the upper left corner.

Definition 1. *Multiple objective optimization problems can be represented as follows:*

$$\begin{cases} max & (f_1(x), f_2(x), \ldots, f_k(x)) \\ min & (g_1(x), g_2(x), \ldots, g_r(x)) \\ s.t. & x \in X = \{x \mid h_t(x) \leq 0, t = 1, \ldots, m\} \end{cases}$$

where $x = (x_1, x_2, \ldots, x_n)$ *are the decision variables;* $f_i(x), (i = 1, \ldots, k)$ *are maximization type objective functions;* $g_j(x), (j = 1, \ldots, r)$ *are minimization type objective function;* $h_t(x)$, $(t = 1, \ldots, m)$ *are set of constraints [60].*

Definition 2. *A decision plan* $x^0 \in X$ *is said to be a Pareto optimal solution to the multiple objective optimization problems if there does not exist another* $y \in X$*, such that* $f_k(y) \leq f_k(x^0)$ *for all k and* $f_s(y) < f_s(x^0)$ *for at least one s Wu et al. [61].*

From the perspective of the search and rescue problem, it is difficult to define the strict upper or lower bounds for the multi-objective setting problem. This is first because of the fuzzy nature of the multi-objective setting but also because of the complexity of obtaining a solution. Therefore, we incorporate both exact and inexact solution approachs to illustrate these issues.

4.1. Transforming Multi-Objective Framework into a Single-Objective One

When dealing with a multi-objective framework, several types of solution approaches can be applied, such as transforming the problem into a single-objective one, incorporating them through a lexicographic method, identifying the entire Pareto front to determine the trade-off among objective weightings, etc. Therefore, it generally comes down to whether the decision maker's preference is incorporated before, under or after exploring the solution space.

In a time-restricted environment such as search and rescue mission planning, it is of absolute necessity that a solution can be obtained in real-time. Therefore, we utilize the approach to transform the multi-objective framework into a single objective. For the bi-objective framework, the objectives do not have a fitting cost transform due to the respective units of the objectives. However, there is a range similarity in terms of the sum of them being between 0 and 1; a simple weighted average is, therefore, fitting to do this. Here, α represents the trade-off between the objectives [62].

$$f_{combined} = \alpha f_1 + (1 - \alpha) f_2 \tag{9}$$

Note that the naive weighted average can be controversial, and we therefore elaborate the use of this in Section 5 (for more information see, Wang [29]).

4.2. GLPK

We utilized the freely available GNU Linear Programming Kit (GLPK) package for the exact solution procedure. The GLPK package is used for large-scale mixed-integer linear programming problems [63]. It utilizes the branch-and-cut method for integer restriction of the decision variables, extending to the branch-and-bound and cutting plane method. The package is implemented in Python, where a maximum solution time is set to 12 min. In a general real-world setting, the ultimately allowed solution time in practice is likely to be lower, and this limit is therefore only set for illustrative purposes.

4.3. Dijkstra's Algorithm

A useful path can be established by implementing graph searching algorithms. In this direction, we utilize Dijkstra's Algorithm, which is extensively used in single-source shortest path problems with non-negative weights for each edge. In implementing the Dijkstra's algorithm for the path finding problem, it is imperative to introduce the constraint on revisiting nodes that represent the same location in different time periods. A way to incorporate this is when visiting the node (i.e., that node being the lowest distance in the queue), then not allowing it to go back after a defined safety period has passed. The set of nodes is then removed in the same way as the visiting node is removed from the queue. Here, the distance that is sought to be minimized is the cumulative score, while the graph traversed is the directed graph G, not allowing it to go backward in time. We refer to the works of Yuan et al. [58] and Sathyara et al. [59] for the detail overview of the algorithm.

4.4. Variable Neighborhood Search

The inexact solution procedure developed in this research is a two-step VNS method that incorporates the general approaches of the VNS but couples that with the known information of directed acyclic graph of feasible paths through a path construction algorithm. The general VNS is proposed by Mladenovic and Hansen [38] in 1997, and it represents a flexible framework for building heuristics to approximately solve combinatorial and non-linear optimization problems. The VNS search heuristic systematically changes its neighborhood structures to obtain a solution. It does so based on the following key observations [64]:

- A local optimum relative to one neighborhood structure is not necessarily a local optimum for another neighborhood structure.
- A global optimum is a local optimum concerning all neighborhood structures.
- Empirical evidence shows that all or a large majority of the local optima are relatively close to each other for many problems.

The ingredients of a variable neighborhood search heuristic include an improvement phase used to improve a given solution and a so-called shaking phase used to resolve local minima traps. The improvement phase, the shaking procedure and the neighborhood change step are executed alternately until a predefined stopping criterion. This research combined it with a path construct algorithm to obtain feasible solutions more quickly and ensure that it follows the stated constraints. The path construct algorithm can be found in the pseudo-code of Algorithm 1. This approach linearly goes through the available time horizon and selects the next maneuver through a weighted probability based on each alternative's respective score. It accompanies the constraint by removing feasible maneuvers and steers it back to the end position by narrowing the feasible maneuvers based on the Chebyshev and Manhattan distances to the end position. Note that this feature of steering the path back to the selected end position is necessary as the two-step VNS randomly selects new neighborhoods to investigate. The grid representation is, therefore, not enough to steer it back. The integrated VNS approach selects a random neighborhood to improve upon the path. It stops selecting new neighborhoods when a designated number of iteration have been investigated. The pseudocode of the algorithm is presented in Algorithms 1 and 2.

Algorithm 1: The path constructing algorithm: **path_constructor(x_original, t1, t2, rs, score)**

```
Result: feasible path in x
x_original := the path that should be updated
N, R, T := the dimensions of the problem (grid size, number of UAVs, size of time horizon)
t1, t2 := the start and end time where a new path between should be located
rs := the set of UAVs
x_new := x_original
x_new[:,:,[t1:t2]] := 0
for r in rs do
    start = starting position
    end = end position
    if start, end is empty then
        start := global start
        end := global end
    end
    t := t1 + 1
    infeasible_neighbor := []
    infeasible_neighbor_T := []
    while t < t2 do
        prior := cell of t-1 maneuver
        feasible_maneuver := all maneuvers inside grid
        /* remove infeasible maneuvers                                          */
        feasible_maneuvers := remove infeasible manuevers as indicated by
          infeasible_neighbor if t is in infeasible_neighbor_T
        if Manhatten distance from prior to end >= t2 - (t-1) then
            maneuver_distance = manhatten distance from possible neighbors to end + 1
            if maneuver_distance <= t2-t then
                remove maneuver from feasible_maneuvers
            end
        end
        if chebyshev distance from prior to end >= t2 - (t-1) then
            maneuver_distance = chebyshev distance from possible neighbors to end + 1
            if maneuver_distance <= t2-t then
                remove maneuver from feasible_maneuvers
            end
        end
        if maneuver in feasible_maneuvers has already been visited by other UAVs then
            remove maneuver from feasible_maneuvers
        end
        if feasible_maneuvers is empty then
            if t-1 is equal to t1 then
                RETURN(x_original)
            end
            else
                Add prior manuever to infeasible_neighbor
                Add t-1 to infeasible_neighbor_T
                remove prior from x_original
                t := t-1
                break
            end
        end
        /* select feasible maneuver based on weighted probability              */
        ranked_maneuvers := ARGSORT(score[feasible_manuevers])
        weighted_maneuvers := ranked_maneuvers / SUM(ranked_maneuvers)
        chosen := CHOOSE(feasible_manuevers, weighted_maneuvers, 1)
        x_new[chosen] = 1
    end
end
RETURN(x_new)
```

Algorithm 2: Pseudocode representing **VNS(score, N, R, T, neighborhood_size, nmax, kmax, tmax)**

```
h! Result: best path P
score := score for each cell
N, R, T := the dimensions of the problem (grid size, number of UAVs, size of time horizon)
neighborhood_size := size of searched neighborhood
nmax := maximum number of neighborhood changes
kmax := maximum searches per neighborhood
tmax := total maximum runtime in seconds
n := 0
x_best := path_constructor(ZEROS(N,R,T), t1=0, t2=end, rs=[0,1], score)
score_best := SUM(x_best * score)
while n<nmax do
    k := 0
    while k<kmax do
        n1 := RANDOM(0,T)
        n2 := min(T,n1+neighborhood_size)
        nr := RANDOM(0,R)
        x_temp := path_constructor(x_best, t1=n1, t2=n2, rs=nr, score)
        score_temp := SUM(x_best * score)
        if score_temp > score_best then
            x_best := x_temp
            score_best := score_temp
            k := 0
        end
        k := k+1
    end
    n := n+1
end
RETURN(x_best)
```

5. Experiments

All numerical experiments were executed with Intel Core i5-8250 CPU with 1.60 GHz processors and 8.00 GB RAM for performance evaluation. For numerical verification, we model the probability map through two hotspots with a spread of two cells.

5.1. Sensitivity of VNS Parameters

The VNS algorithm has three different parameters indicating the search depth, i.e., neighborhood, $nmax$ and $kmax$, defining the size of the neighborhood each search considers; the maximum number of searched neighborhoods; and the number of searches per neighborhood. The results are shown in Table 1.

Table 1. Average performance, standard deviation and average runtime for 100 different runs with different neighborhood parameter.

Neighborhood Parameter	Relative Performance ($f_{combined}$)	Standard Deviation ($f_{combined}$)	Runtime (s)
0.250	0.630	0.086	13.239
0.333	0.820	0.033	20.094
0.417	0.837	0.036	27.870
0.500	0.860	0.025	35.917
0.583	0.908	0.027	39.621
0.667	0.921	0.027	42.898
0.750	0.893	0.043	47.320
0.833	0.862	0.035	49.739

The performance in Table 2 illustrates the change in deviation and runtime when modifying the *nmax* and *kmax* parameter, but it should be noted that the computation of these could easily be parallelized. In the parameter indicating the neighborhood's size, we can see that there is not a unified result showing which size of a neighborhood to chose. Therefore, we choose to further extend the algorithm by randomly selecting a length within the range of 0.3 to 0.9 for each neighborhood change. This furthers the shake and improvement steps of the VNS, as both local and global solutions will be investigated.

Table 2. Average relative performance of the Variable Neighborhood Search (VNS) method compared to the exact GNU Linear Programming Kit (GLPK) approach for 100 different runs with different nmax and kmax settings.

	Avg. Performance (Relative to GLPK)						**Avg. Runtime (In Seconds)**					
nmax\kmax	**5**	**15**	**25**	**35**	**45**	**55**	**5**	**15**	**25**	**35**	**45**	**55**
50	0.778	0.844	0.847	0.865	0.907	0.889	1.279	3.634	6.006	8.895	10.885	13.773
100	0.777	0.865	0.889	0.885	0.885	0.847	2.335	7.114	11.527	16.675	22.014	26.318
150	0.931	0.890	0.870	0.931	0.950	0.933	3.662	11.177	16.975	24.241	32.656	37.795
200	0.926	0.931	0.843	0.823	0.864	0.912	5.356	14.136	23.159	32.099	42.245	54.182
250	0.867	0.779	0.911	0.933	0.891	0.865	6.185	16.359	28.851	39.850	55.093	62.888
500	0.932	0.867	0.913	0.911	0.869	0.975	12.262	33.660	59.775	87.177	102.260	126.429
1000	0.934	0.912	0.910	0.868	0.871	0.932	23.976	70.143	111.092	156.020	211.473	269.177
1500	0.886	0.846	0.928	0.928	0.867	0.913	32.441	103.338	171.001	248.853	323.948	360.003
2000	0.927	0.849	0.911	0.912	0.956	0.912	45.564	133.198	229.218	338.928	360.003	360.007
2500	0.869	0.976	0.911	0.912	0.974	0.912	61.237	175.410	298.774	360.004	360.002	360.002

5.2. Performance and Runtime for VNS, Dijkstra, and GLPK

GLPK is an exact approach and is therefore significantly slower, but it also yields the optimal solution. However, the GLPK is not able to solve any of the larger problem scenarios. The performance and runtime for the three approaches on different scenario sizes relative to grid size N, time horizon T and the number of UAVs R can be seen in Figure 2. Note that, when a solution approach reaches the time limit, the time is noted, while its performance is not.

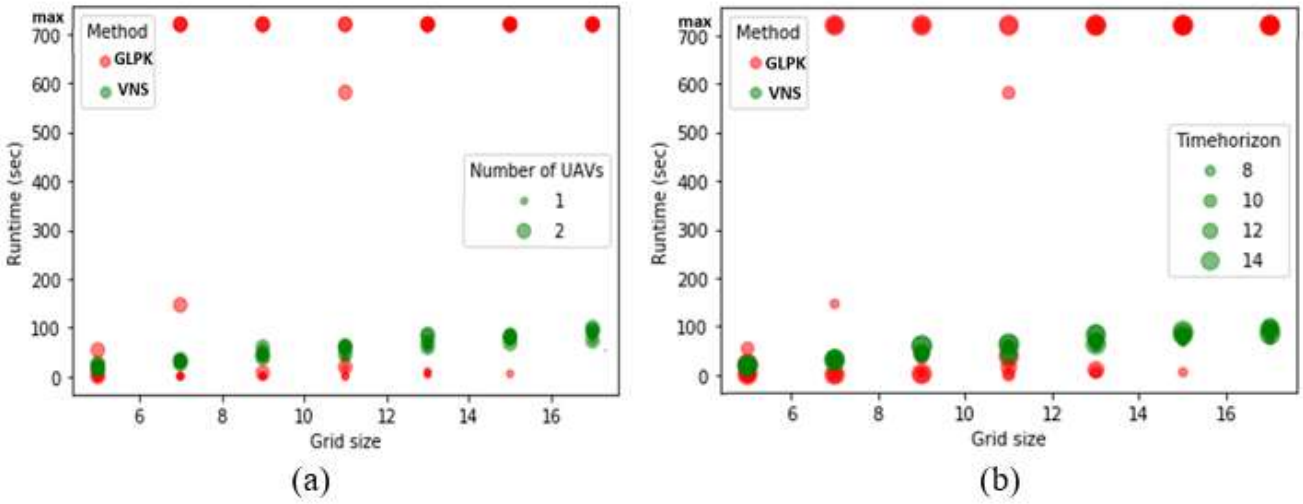

Figure 2. (**a**) The relative performance of the Variable Neighborhood Search (VNS) and Dijkstra algorithm compared to the optimal solution found by the GNU Linear Programming Kit (GLPK) approach is shown. Note that many experiments do not yield a relative performance as GLPK could not obtain a solution. (**b**) The runtime for the three approaches is presented, demonstrating relation to different grid sizes and time horizons. The exact value of performance measures is presented in Table 3.

Table 3. The performance of the respective solution approaches on different scenarios. Note that GLPK could not obtain a solution on some of the scenarios. This is illustrated by (-), while its runtime reached the limit of 720 s.

Grid Size	Time Horizon	No. of UAVs	Performance ($f_{combined}$)			Relative Performance	
N	T	R	VNS	Dijkstra	GLPK	$\frac{f_{\|VNS}}{f_{\|GLPK}}$	$\frac{f_{\|Dijkstra}}{f_{\|GLPK}}$
5	10	1	0.225	0.018	0.242	0.928	0.074
5	10	2	0.137	0.324	0.416	0.329	0.778
5	14	1	0.310	0.206	0.357	0.868	0.576
5	14	2	0.454	0.124	0.454	1.000	0.273
5	18	1	0.332	0.199	0.484	0.686	0.412
5	18	2	0.371	0.172	0.428	0.865	0.406
5	22	1	0.460	0.128	0.460	1.000	0.280
5	22	2	0.400	0.185	0.481	0.830	0.384
7	10	1	0.075	0.046	0.102	0.730	0.450
7	10	2	0.082	0.019	0.135	0.612	0.143
7	14	1	0.163	0.037	0.178	0.912	0.208
7	14	2	0.119	0.045	0.143	0.831	0.316
7	18	1	0.366	0.140	0.385	0.951	0.364
7	18	2	0.214	0.061	-	-	-
7	22	1	0.554	0.005	0.554	1.000	0.009
7	22	2	0.344	0.016	-	-	-
9	10	1	0.184	0.057	0.222	0.832	0.258
9	10	2	0.319	0.188	0.344	0.927	0.546
9	14	1	0.096	0.035	0.101	0.947	0.351
9	14	2	0.026	0.011	-	-	-
9	18	1	0.346	0.244	0.371	0.932	0.656
9	18	2	0.431	0.003	-	-	-
9	22	1	0.446	0.237	0.496	0.899	0.479
9	22	2	0.458	0.300	-	-	-
11	10	1	0.185	0.007	0.185	1.000	0.042
11	10	2	0.283	0.099	0.296	0.958	0.336
11	14	1	0.273	0.042	0.297	0.916	0.141
11	14	2	0.408	0.162	0.427	0.955	0.380
11	18	1	0.242	0.029	0.297	0.812	0.098
11	18	2	0.272	0.003	-	-	-
11	22	1	0.447	0.191	0.447	0.999	0.428
11	22	2	0.545	0.002	-	-	-
13	10	1	0.097	0.056	0.111	0.876	0.504
13	10	2	0.110	0.027	-	-	-
13	14	1	0.223	0.068	0.248	0.899	0.273
13	14	2	0.347	0.274	-	-	-
13	18	1	0.348	0.002	0.348	1.000	0.008
13	18	2	0.522	0.005	-	-	-
13	22	1	0.373	0.089	-	-	-
13	22	2	0.547	0.047	-	-	-
15	10	1	0.111	0.042	0.111	0.999	0.384
15	10	2	0.085	0.028	-	-	-
15	14	1	0.095	0.009			
15	14	2	0.039	0.028	-	-	-
15	18	1	0.072	0.043	-	-	-
15	18	2	0.112	0.039	-	-	-
15	22	1	0.092	0.004	-	-	-
15	22	2	0.107	0.031	-	-	-
17	10	1	0.010	0.008			
17	10	2	0.024	0.081	-	-	-
17	14	1	0.492	0.068	-	-	-
17	14	2	0.573	0.355	-	-	-
17	18	1	0.372	0.163	-	-	- -
17	18	2	0.448	0.003	-	-	-
17	22	1	0.249	0.001	-	-	-
17	22	2	0.141	0.033	-	-	-

The performance clearly indicates that the GLPK is generally faster for small problem scenarios with a single UAV. However, it cannot even obtain a solution whenever there are two UAVs to consider or the grid size or time horizon is larger. The relative performance of VNS indicates that, for larger problem scenarios, it will perform within 20% of the optimal

solution, while, for smaller problem scenarios, it performs within 50% of the optimal. The latter is perhaps because VNS searches with a neighborhood size that is too small relative to the grid size, so it will never get out of the local optima. However, it does not seem to be an issue for the larger problem scenarios. Similarly, the Dijkstra approach seems to decrease in performance relative to the exact approach when the scenario size increases. This is probably due to the greedy nature of the method, as it does not want to investigate areas that require it to cross a section of cells without any probability of success. The results also showcase the complexity of the large-scale problem scenarios in UAV-assisted search and rescue missions. Overall, Figure 2 demonstrates that the VNS outperforms GLPK and Dijkstra's algorithm in the perspective of relative performance measures for most of the instances.

5.3. Sensitivity of Objective Weighting for the GLPK

Figure 3 illustrates the sensitivity to changes in the trade-off between objectives represented by modifying α. The sensitivity analysis sheds light on the change in the optimal path for different trade-offs. Figure 3 shows that the UAVs for alpha equal to 0 and 0.1 clearly stay in take-off and landing zone for the entire time horizon for both UAVs or just for one UAV. This is because the score for each grid cell outside the take-off zone is too high to consider. Finally, Figure 4 shows that the optimal path changes for almost all different alpha settings. However, the pattern of each path seems to follow the same structure because the path is sensitive to the parameter α, which also justifies the multi-objective formulation of the problem.

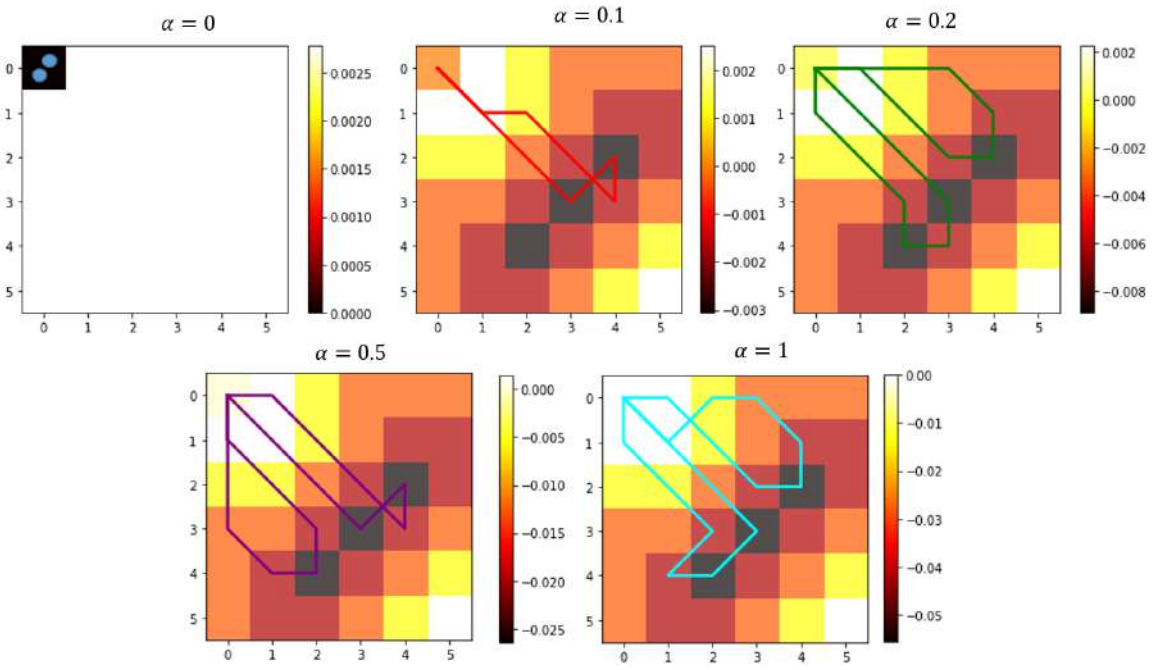

Figure 3. The corresponding route in 2D generated by GLPK for different weightings of alpha on the corresponding scoring map. Note that the illustrated paths is for two UAVs on a 6×6 grid with a time horizon of 10 and start and end in grid cell $[0, 0]$. In addition, for alpha = 0.1, the second UAV stays in the start cell.

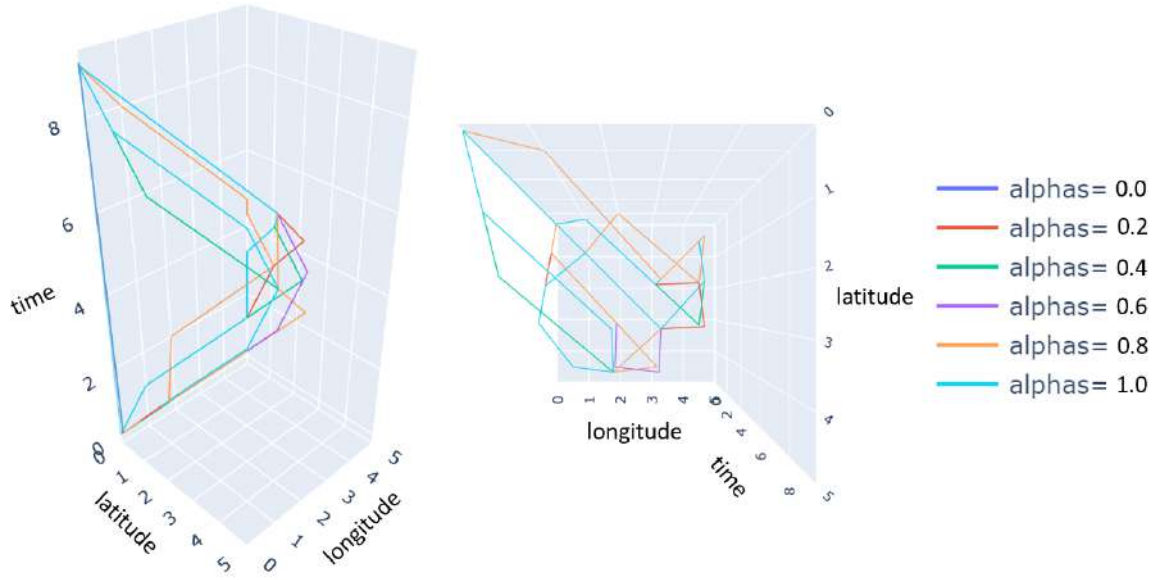

Figure 4. The corresponding route in three dimension generated by GLPK for different weights of alpha. The longitude and latitude axes represent the possible maneuvers on the grid, while time illustrates the time dimension.

5.4. Benefits and Adverse Circumstances Associated with Multi-Objective Framework

The results on the sensitivity clearly showcase some of the dangers when incorporating the bi-objective framework on the UAV pathfinding. It is very difficult to see which alpha enforces that all equipment will be employed and not spending too much time in the landing zone. Clearly, the solution procedure should allow UAVs to return before time, but it is very difficult to identify when it is too early to specify through the alpha parameter.

There is similarly a robustness issue when introducing the multi-objective framework as objectives can be conflicting, and a solution can satisfy an objective that is not of our interest. In the case of this paper, we are clearly interested in searching as many high probability cells as possible in as little time as possible. However, indicating how little time is too much is very difficult in the presented setting. The last thing one wants to introduce is nervousness in the scheduling, so some rules about searching different areas could be of advantage.

Nevertheless, introducing these additional objectives clearly brings us closer to the optimal goal. For these search and rescue Missions, we are interested in accumulating the highest probability of locating the missing target. We are, however, also interested in doing it as quickly as possible by obtaining the best quality images possible. Similarly, there could be a chance that the missing target has a higher probability of survival in some regions than others, which is why we also are interested in locating the target alive. Therefore, additional objectives other than the ones considered in this research could be introduced.

6. Conclusions

The smart city concept is almost around last couple of decades, and one of the critical concepts is to integrate cutting-edge technology without raising costs in improving environmental sustainability and life expectancy. In this direction, we proposed a multi-objective path planning and trajectory mapping problem under the mixed integer programming problem framework for a set of homogeneous UAVs deployed to search for static targets. A graph theory-based directed acyclic network representation is employed to reduce complexities and track the inward and outward movement of each UAV from its respective present cell location by ensuring flow conservation. A modification of the basic VNS algorithm is proposed and implemented in two phases to find the solution. In the first phase, a path is generated and in the second phase, trajectory mapping is done sequentially by considering constraints associated with the problem environment. Numerical simulation on synthetic experimental settings demonstrates that the proposed approach can reduce computational

complexity and provide a solution within reasonable amount of time compared to the exact solver. Moreover, it is found that the exact solver is unable to provide a solution within a time threshold. When we compare the relative performance of VNS with GLPK or Dijkstra's algorithm, it was found that Dijkstra's algorithm's performance is relatively lower as the grid size increases, which justifies the efficiency of the proposed algorithm. To our best knowledge, this is the first work to explore the path for multiple UAVs by using a bi-objective VNS algorithm. Considering the numerical evaluation, one can conclude that the approach presented in this study is a better alternative than the exact solver, and methodology can contribute to intelligent systems.

For future work, we intend to extend the proposed approach to calculate paths for finding moving targets. We assumed altitude differentiation from the perspective of collision avoidance. We ignored constraints such as fuel, sensor capacity, search pattern, etc., those need to be integrated to formulate a robust path planning model. We compared the outcome of proposed solution approach with exact solver, therefore one can employ other algorithms such as particle swarm optimization [65], bat algorithm [66], A^* algorithm [59], machine learning (ML) algorithms [29] etc. to compare the performance of the proposed VNS algorithm. Finally, one can use a multi-criterion decision-making algorithm [67] to incorporate customizable preferences of decision-makers robustly to take advantage of the inherent flexibility while setting weights.

Author Contributions: Conceptualization, S.S. and A.E.V.; methodology, S.S. and A.E.V.; software, A.E.V.; validation, I.N., A.H. and H.B.; formal analysis, I.N.; investigation, A.E.V.; resources, A.H. and H.B.; writing—review and editing, S.S., A.E.V. and I.N.; and project administration, S.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data used in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Nielsen, I.E.; Dang, V.Q.; Nielsen, P.; Pawlewski, P. Scheduling of Mobile Robots with Preemptive Tasks, Advances in Intelligent Systems and Computing. In *Distributed Computing and Artificial Intelligence, 11th International Conference*; Springer: Berlin, Germany, 2014; pp. 19–29.
2. Sung, I., Nielsen, P. Zoning a service area of unmanned aerial vehicles for package delivery services. *J. Intell. Robot. Syst.* **2020**, *97*, 719–731. [CrossRef]
3. Thibbotuwawa, A.; Bocewicz, G.; Radzki, G.; Nielsen, P.; Banaszak, Z. UAV Mission planning resistant to weather uncertainty. *Sensors* **2020**, *20*, 515. [CrossRef]
4. Eagle, J.N. 1984. The optimal search for a moving target when the search path is constrained. *Oper. Res.* **1984**, *32*, 1107–1115. [CrossRef]
5. Pugliese, L.D.P.; Guerriero, F.; Zorbas, D.; Razafindralambo, T. Modelling the mobile target covering problem using flying drones. *Optim. Lett.* **2016**, *10*, 1021–1052. [CrossRef]
6. Samaniego, F.; Sanchis, J.; García-Nieto, S.; Simarro, R. Recursive Rewarding Modified Adaptive Cell Decomposition (RR-MACD): A Dynamic Path Planning Algorithm for UAVs. *Electronics* **2019**, *8*, 306. [CrossRef]
7. San, Juan, V.; Santos, M.; Andújar, J.M. Intelligent UAV map generation and discrete path planning for search and rescue operations. *Complexity* **2018**. [CrossRef]
8. Stewart, T.J. Search for a moving target when searcher motion is restricted. *Comput. Oper. Res.* **1979**, *6*, 129–140. [CrossRef]
9. Wang, J.; Jiang, C.; Han, Z.; Ren, Y.; Maunder, R.G.; Hanzo, L. Taking drones to the next level: Cooperative distributed unmanned-aerial-vehicular networks for small and mini drones. *IEEE Veh. Technol. Mag.* **2017**, *12*, 73–82. [CrossRef]
10. Yuan, H.; Xiao, C.; Zhan, W.; Wang, Y.; Shi, C.; Ye, H.; Jiang, K.; Ye, Z.; Zhou, C.; Wen, Y. Target detection, positioning and tracking using new UAV gas sensor systems: Simulation and analysis. *J. Intell. Robot. Syst.* **2019**, *94*, 871–882. [CrossRef]
11. Felemban, E.; Sheikh, A.A.; Naseer, A. Improving response time for crowd management in Hajj. *Computers* **2021**, *4*, 46. [CrossRef]
12. Miyano, K.; Shinkuma, R.; Shiode, N.; Shiode, S.; Sato, T.; Oki, E. Multi-UAV allocation framework for predictive crime deterrence and data acquisition. *Internet Things* **2020**, *11*, 100205. [CrossRef]
13. Huang, S.; Gui, J.; Wang, T.; Li, X. Joint Mobile Vehicle–UAV Scheme for Secure Data Collection in a Smart City. *Ann. Telecommun.* **2020**, 1–22. Available online: https://www.researchgate.net/publication/344005230_Joint_mobile_vehicle-UAV_scheme_for_secure_data_collection_in_a_smart_city (accessed on 10 March 2021). [CrossRef]

14. Trummel, K.E.; Weisinger, J.R. The complexity of the optimal searcher path problem. *Oper. Res.* **1986**, *34*, 324–327. [CrossRef]
15. Brown, S.S. Optimal search for a moving target in discrete time and space. *Oper. Res.* **1980**, *28*, 1275–1289. [CrossRef]
16. Benkoski, S.J.; Monticino, M.G.; Weisinger, J.R. A survey of the search theory literature. *Nav. Res. Logist.* **1991**, *38*, 469–494. [CrossRef]
17. Washburn, A.R. Branch and bound methods for a search problem. *Nav. Res. Logist.* **1998**, *45*, 243–257. [CrossRef]
18. Lau, H.; Huang, S.; Dissanayake, G. Optimal search for multiple targets in a built environment. In Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, Edmonton, AB, Canada, 2–6 August 2005; pp. 3740–3745.
19. Lau, H.; Huang, S.; Dissanayake, G. Discounted mean bound for the optimal searcher path problem with non-uniform travel times. *Eur. J. Oper. Res.* **2008**, *190*, 383–397. [CrossRef]
20. Rogge, J.A.; Aeyels, D. Multi-robot coverage to locate fixed and moving targets. In Proceedings of the 2009 IEEE Control Applications (CCA) & Intelligent Control (ISIC), St. Petersburg, Russia, 8–10 July 2009; pp. 902–907.
21. Li, X.; Yao, H.; Wang, J.; Xu, X.; Jiang, C.; Hanzo, L. A near-optimal UAV-aided radio coverage strategy for dense urban areas. *IEEE Trans. Veh. Technol.* **2019**, *68*, 9098–9109. [CrossRef]
22. Berger, J.; Lo N. An innovative multi-agent search-and-rescue path planning approach. *Comput. Oper. Res.* **2015**, *53*, 4–31. [CrossRef]
23. Perez-Carabaza, S.; Besada-Portas, E.; Lopez-Orozco, J.A.; Jesus, M. Ant colony optimization for multi-UAV minimum time search in uncertain domains. *Appl. Soft Comput.* **2018**, *62*, 789–806. [CrossRef]
24. Ye, F.; Chen, J.; Tian, Y.; Jiang, T. Cooperative task assignment of a heterogeneous multi-UAV system using an adaptive genetic algorithm. *Electronics* **2020**, *4*, 687. [CrossRef]
25. Lu, Y.; Ma, Y.; Wang, J.; Han, L. Task assignment of UAV swarm based on Wolf Pack algorithm. *Appl. Sci.* **2020**, *23*, 8335. [CrossRef]
26. Luo, R.; Zheng, H.; Guo, J. Solving the multi-functional heterogeneous UAV cooperative mission planning problem using multi-swarm fruit fly optimization algorithm. *Sensors* **2020**, *18*, 5026. [CrossRef]
27. Alhaqbani, A.; Kurdi, H.; Youcef-Toumi, K. Fish-inspired task allocation algorithm for multiple unmanned aerial vehicles in search and rescue missions. *Remote Sens.* **2021**, *1*, 27.
28. Xiong, C.; Chen, D.; Lu, D.; Zeng, Z.; Lian, L. Path planning of multiple autonomous marine vehicles for adaptive sampling using Voronoi-based ant colony optimization. *Robot. Auton. Syst.* **2019**, *115*, 90–103. [CrossRef]
29. Wang, J.; Jiang, C.; Zhang, H.; Ren, Y.; Chen, K.C.; Hanzo, L. Thirty years of machine learning: The road to Pareto-optimal wireless networks. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 1472–1514. [CrossRef]
30. Kundid Vasić, M.; Papić, V. Multimodel Deep Learning for Person Detection in Aerial Images. *Electronics* **2020**, *9*, 1459. [CrossRef]
31. Li, Y.; Yuan, X.; Zhu, J.; Huang, H.; Wu, M. Multiobjective Scheduling of Logistics UAVs Based on Variable Neighborhood Search. *Appl. Sci.* **2020**, *10*, 3575. [CrossRef]
32. Lo N.; Berger, J.; Noel, M. Toward optimizing static target search path planning. In Proceedings of the 2012 IEEE Symposium on Computational Intelligence for Security and Defence Applications, Ottawa, ON, Canada, 11–13 July 2012; pp. 1–7.
33. Nielsen, L.D.; Sung, I.; Nielsen, P. Convex decomposition for a coverage path planning for autonomous vehicles: Interior extension of edges. *Sensors* **2019**, *19*, 4165. [CrossRef]
34. Perez-Carabaza, S.; Scherer, J.; Rinner, B.; López-Orozco, J.A.; Besada-Portas, E. UAV trajectory optimization for Minimum Time Search with communication constraints and collision avoidance. *Eng. Appl. Artif. Intell.* **2019**, *85*, 357–371. [CrossRef]
35. Raap, M.; Meyer-Nieberg, S.; Pickl, S.; Zsifkovits, M. Aerial vehicle search-path optimization: A novel method for emergency operations. *J. Optim. Theory Appl.* **2017**, *172*, 965–983. [CrossRef]
36. Sitek, P.; Wikarek, J.; Rutczyńska-Wdowiak, K.; Bocewicz, G.; Banaszak, Z. Optimization of capacitated vehicle routing problem with alternative delivery, pick-up and time windows: A modified hybrid approach. *Neurocomputing* **2021**, *423*, 670–678. [CrossRef]
37. Thibbotuwawa, A.; Bocewicz, G.; Zbigniew, B.; Nielsen, P. A solution approach for UAV fleet mission planning in changing weather conditions. *Appl. Sci.* **2019**, *9*, 3972. [CrossRef]
38. Mladenovic, N.; Hansen, P. Variable neighborhood search. *Comput. Oper. Res.* **1997**, *24*, 1097–1100. [CrossRef]
39. Loudni, S.; Boizumault, P.; David, P. On-line resources allocation for ATM networks with rerouting. *Comput. Oper. Res.* **2006**, *33*, 2891–2917. [CrossRef]
40. Geiger, M.J.; Wenger, W. On the assignment of students to topics: A Variable Neighborhood Search approach. *Socio-Econ. Plan. Sci.* **2010**, *44*, 25–34. [CrossRef]
41. Brusco, M.J.; Singh, R.; Steinley, D. Variable neighborhood search heuristics for selecting a subset of variables in principal component analysis. *Psychometrics* **2009**, *74*, 705. [CrossRef]
42. Schilde, M.; Doerner, K.F.; Hartl, R.F.; Kiechle, G. Metaheuristics for the bi-objective orientation problem. *Swarm Intell.* **2009**, *3*, 179–201. [CrossRef]
43. Anghinolfi, D.; Paolucci, M. Parallel machine total tardiness scheduling with a new hybrid metaheuristic approach. *Comput. Oper. Res.* **2007**, *34*, 3471–3490. [CrossRef]
44. Qian, B.; Wang, L.; Huang, D.X.; Wang, X. Multi-objective flow shop scheduling using differential evolution. In *Intelligent Computing in Signal Processing and Pattern Recognition*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 1125–1136.
45. Fleszar, K.; Osman, I.H.; Hindi, K.S. A variable neighborhood search algorithm for the open vehicle routing problem. *Eur. J. Oper. Res.* **2009**, *195*, 803–809. [CrossRef]

46. Montemanni, R.; Smith, D.H. Construction of constant GC-content DNA codes via a variable neighborhood search algorithm. *J. Math. Model. Algorithms* **2008**, *7*, 311. [CrossRef]
47. Hansen, P.; Mladenović, N.; Pérez, J.A.M. Variable neighborhood search: Methods and applications. *Ann. Oper. Res.* **2010**, *175*, 367–407. [CrossRef]
48. Ribeiro, C.C.; Aloise, D.; Noronha, T.F.; Rocha, C.; Urrutia, S. An efficient implementation of a VNS/ILS heuristic for a real-life car sequencing problem. *Eur. J. Oper. Res.* **2008**, *191*, 596–611. [CrossRef]
49. Gosiewski, Z.; Kwaśniewski, K. Time Minimization of Rescue Action Realized by an Autonomous Vehicle. *Electronics* **2020**, *9*, 2099. [CrossRef]
50. Lanillos, P.; Yañez-Zuluaga, J.; Ruz, J.J.; Besada-Portas, E. A bayesian approach for constrained multi-agent minimum time search in uncertain dynamic domains. In Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation, Amsterdam, The Netherlands, 6–10 July 2013; pp. 391–398.
51. Peng, Z.H.; Wu, J.P.; Chen, J. Three-dimensional multi-constraint route planning of unmanned aerial vehicle low-altitude penetration based on coevolutionary multi-agent genetic algorithm. *J. Cent. South Univ. Technol.* **2011**, *18*, 1502. [CrossRef]
52. Sakawa, M.; Yano, H.; Nishizaki, I.; Nishizaki, I. *Linear and Multiobjective Programming with Fuzzy Stochastic Extensions*; Springer US: New York, NY, USA, 2013.
53. Chen, L.; Peng, J.; Zhang, B. Uncertain goal programming models for bicriteria solid transportation problem. *Appl. Soft Comput.* **2017**, *51*, 49–59. [CrossRef]
54. Chung, C.K.; Chen, H.M.; Chang, C.T.; Huang, H.L. On fuzzy multiple objective linear programming problems. *Expert Syst. Appl.* **2018**, *114*, 552–562. [CrossRef]
55. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T.A.M.T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **2002**, *6*, 182–197. [CrossRef]
56. Moon, I.; Jeong, Y.J.; Saha, S. Fuzzy bi-objective production-distribution planning problem under the carbon emission constraint. *Sustainability* **2016**, *8*, 798. [CrossRef]
57. Davoodi, M.; Panahi, F.; Mohades, A.; Hashemi, S.N. Multi-objective path planning in discrete space. *Appl. Soft Comput.* **2013**, *13*, 709–720. [CrossRef]
58. Yuan, Z.; Yang, Z.; Lv, L.; Shi, Y. A Bi-Level Path Planning Algorithm for Multi-AGV Routing Problem. *Electronics* **2020**, *9*, 1351. [CrossRef]
59. Sathyaraj, B.M.; Jain, L.C.; Finn, A.; Drake, S. Multiple UAVs path planning algorithms: A comparative study. *Fuzzy Optim. Decis. Mak.* **2008**, *7*, 257. [CrossRef]
60. Nielsen, I.E.; Bocewicz, G.; Saha, S. Multi-Agent Path Planning Problem Under a Multi-objective Optimization Framework. In *DCAI 2020: Distributed Computing and Artificial Intelligence, Special Sessions, 17th International Conference, Proceedings of the International Symposium on Distributed Computing and Artificial Intelligence, L'Aquila, Italy, 17–19 June 2020*; Springer: Cham, Switzerland, 2020; pp. 5–14.
61. Wu, Y.K.; Liu, C.C.; Lur, Y.Y. Pareto-optimal solution for multiple objective linear programming problems with fuzzy goals. *Fuzzy Optim. Decis. Mak.* **2015**, *14*, 43–55. [CrossRef]
62. Marler, R.T.; Arora, J.S. Survey of multi-objective optimization methods for engineering. *Struct. Multidiscip. Optim.* **2004**, *26*, 369–395. [CrossRef]
63. Makhorin, A. GLPK (GNU Linear Programming Kit). 2008. Available online: http://www.gnu.org/s/glpk/glpk.html (accessed on 12 March 2019).
64. Hidalgo-Paniagua, A.; Vega-Rodríguez, M.A.; Ferruz, J. Applying the MOVNS (multi-objective variable neighborhood search) algorithm to solve the path planning problem in mobile robotics. *Expert Syst. Appl.* **2016**, *58*, 20–35. [CrossRef]
65. Jeong, Y.; Saha, S.; Chatterjee, D.; Moon, I. Direct shipping service routes with an empty container management strategy. *Transp. Res. Part E Logist. Transp. Rev.* **2018**, *118*, 123–142. [CrossRef]
66. Saha, S.; Chatterjee, D.; Sarkar, B. The ramification of dynamic investment on the promotion and preservation technology for inventory management through a modified flower pollination algorithm. *J. Retail. Consum. Serv.* **2021**, *58*, 102326. [CrossRef]
67. Vasegaard, A.E.; Picard, M.; Hennart, F.; Nielsen, P.; Saha, S. Multi criteria decision making for the multi-satellite image acquisition scheduling problem. *Sensors* **2020**, *20*, 1242. [CrossRef]

electronics

MDPI

Article

Optimization of Public Transport Services to Minimize Passengers' Waiting Times and Maximize Vehicles' Occupancy Ratios

Ivana Hartmann Tolić [1,*], Emmanuel Karlo Nyarko [2] and Avishai (Avi) Ceder [3,4,5]

[1] Department of Software Engineering, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek, J.J. Strossmayer University of Osijek, 31000 Osijek, Croatia
[2] Department of Computer Engineering and Automation, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek, J.J. Strossmayer University of Osijek, 31000 Osijek, Croatia; karlo.nyarko@ferit.hr
[3] Transportation Research Institute, Technion—Israel Institute of Technology, Faculty of Civil and Environmental Engineering, Haifa 32000, Israel; a.ceder@auckland.ac.nz
[4] Department of Civil and Environmental Engineering, University of Auckland, Auckland 1142, New Zealand
[5] School of Traffic and Transportation, Beijing Jiaotong University, Beijing 100044, China
* Correspondence: ivana.hartmann@ferit.hr

Received: 21 January 2020; Accepted: 14 February 2020; Published: 20 February 2020

Abstract: Determining the best timetable for vehicles in a public transportation (PT) network is a complex problem, especially because it is just necessary to consider the requirements and satisfaction of passengers as the requirements of transportation companies. In this paper, a model of the PT timetabling problem which takes into consideration the passenger waiting time (PWT) at a station and the vehicle occupancy ratio (VOR) is proposed. The solution aims to minimize PWT and maximize VOR. Due to the large search space of the problem, we use a multiobjective particle swarm optimization (MOPSO) algorithm to arrive at the solution of the problem. The results of the proposed method are compared with similar results from the existing literature.

Keywords: optimization models; timetable; passenger waiting time; vehicle occupancy ratio

1. Introduction

In modern transportation systems, the greatest challenge is to minimize, in general terms, energy consumption, and maximize economic, technological and social goals. The problem of optimizing and finding the best timetable for public transportation (PT) vehicles has been known for years [1–3]. Recent research has provided more efficient algorithms which have achieved better results by modifying known mathematical models and modifying or combining various known algorithms [4–6]. Planning of PT is a highly complex task which is usually analyzed via two different aspects: minimization of the passenger waiting time (PWT) at the station and optimization of the number and/or sizes of vehicles. The train timetabling problem (TTP) has recently been studied, and the main problem in this field is to determine a periodic or non-periodic timetable, which satisfies the capacities of vehicles and limits of operations [7–10]. Some of the greatest challenges in waiting time (WT) minimization models are to find the optimal number of vehicles and to find the optimal route and minimize travel time [11]. The goal of every PT service should be to attract more people to use it by reducing the use of private cars, which is directly related to reducing traffic congestion, decreasing the number of car accidents and reducing pollution. The use of PT services by passengers depends on three elements; namely, travel time (walking, waiting and riding times), fare (ticket and other related services' costs) and convenience (a comfortable walk, waiting under a shelter, having a seat on the

vehicle, air-conditioning on the vehicle, etc.). However, for the PT operator to maximize profits, operational costs need to be minimized. The PT operator's requirements can be achieved by designing an efficient network (the more transfers the network has the more efficient it is), adopting a quality timetable, efficiently schedule the vehicles and maximize the vehicle occupancy ratio (VOR). This paper proposes a model of the PT timetabling problem whose solution improves PT operations planning in terms of timetabling and vehicle scheduling; that is, changes to the departure times and assignment of the PT vehicles, so as to reduce total PWT and increase VOR (thereby minimizing operational costs of the vehicles for the PT operator). Due to the complexity of the problem, a multiobjective particle swarm optimization (MOPSO) algorithm is used in order to minimize PWT while maximizing VOR. The paper is structured as follows: Section 2 extensively describes the state-of-the-art in the timetabling problem. Section 3 elaborates on the problem statement. Section 4 presents the proposed method, and Section 5 contains two numerical examples. Section 6 presents the analysis and discussion of the results. Section 7 contains the concluding remarks.

2. State-of-the-Art

In order to increase the productivity and efficiency of transport services on the one hand, and customer satisfaction on the other , four main analytical methods for determining the number of vehicles needed during the relevant period are presented in [1,3,12]. The first two methods in these papers ensure the average maximum daily occupancy of the vehicles during a given period. The other two methods elaborate on the capacities of the vehicles, which will never be exceeded with an additional constraint on the part of the route which is loaded more than the required availability. Numerical calculation of the average PWT at a station with a limited capacity of the intercity transport vehicles is described in [13]. It is concluded that for a more accurate representation of PWT at the station, the reliability of supply services, passenger behavior and characteristics of the transportation system should be considered. The passenger transport problem is always observed from the viewpoint of costs and improvement of business transport companies. On the contrary, passenger satisfaction is rarely taken into account, especially when creating the timetable [14]. Passenger satisfaction is directly related to the total travel time; the shorter the travel time the more satisfied the passenger is and vice-versa. Excluding the travel speed of the vehicle, the total travel time can be reduced by reducing PWT and the number of passenger transfers. Hence, in this paper, we use the term passenger satisfaction to refer primarily to PWT. A multicriteria approach to timetabling problems, with an emphasis on minimizing PWT at the station, is proposed in [15,16]. The authors in these papers analyze two criteria; namely, empty seat penalty (empty seat kilometers or empty seat hours) and approximate PWT at the station. The problem is solved using a multiobjective label-correcting algorithm that results in 43% saving of PWT with an acceptable load of the vehicle. In order to ensure fast and energy efficient urban rail transport, a nonlinear problem of minimizing the total travel time and energy consumption is presented in [17]. The model reduces the cost (number of trains and energy consumption) and improves passenger satisfaction (reduced PWT and the number of transfers, thereby reducing the total travel time). Optimization of energy consumption and PWT can also be found in [18]. The authors in this paper propose a bi-objective timetable optimization model to minimize PWT and energy consumption. The genetic algorithm (GA) is used for generating a solution with reduced total energy consumption and total passenger waiting time in comparison to the real timetable. The timetable synchronization problem (TSP) refers to the problem of waiting time during a transfer, which is usually solved using a branch and bound (B&B) method. In order to speed up the execution time, the optimization based heuristic method (OHM) is developed and compared with B&B in [19]. It is concluded that OHM is much faster and optimizes the problem more efficiently. The scheduling problem is usually based on maximization of the number of synchronized vehicles arriving at the transfer station or minimization of the total waiting time at the station. For solving the latter problem, a genetic algorithm with local search is used in [20]. The model is applied to a small bus network and the cost of the waiting time is reduced by 9.5%. An optimization model for synchronizing a timetable

is proposed in [21]; i.e., for minimizing the maximum PWT during the transfer and reducing the worst time of the transfer. Mathematical models with PWT as the objective function at the transfer station include a set of mixed integer programming (MIP) models [22]. The model can be solved using a conventional MIP solver such as CPLEX solver (B&B) if there are fewer than 50 lines and by using genetic algorithms for a greater network. Additionally, a solution of the timetable synchronization problem is proposed in [23]. The authors develop a multicriteria optimization model which takes the vehicle scheduling and passenger demand assignment into account. In order to solve the problem to obtain a set of Pareto-efficient solutions, a novel deficit function (DF)-based sequential search method is proposed. Minimization of the average transfer time in the periodic scheduling of trains (PRTS—periodic railway timetable scheduling problem) is solved using an improved differential evolution (DE) algorithm with dual population [24]. A comparison of the presented model against the B&B method and greedy-based heuristic algorithm for using the PRTS simulation algorithm to solve the problem of schedules shows that the given model provides a better indicator for PRTS problem and numerical indicators of optimization functions. The problem of minimizing PWT at the station and the cost of vehicle occupancy is solved using the genetic algorithm in [25]. The models developed for solving the problem of minimizing PWT at the station are defined with preserving the flow of passengers waiting at the station, as shown in [26,27]. The problem of PWT at the station occurs when there is a delay in the timetable. One possible solution is to include additional time delays according to the probability theory; i.e., the objective function includes an exponential distribution of delay, $Exp()$ with the expected delay, as shown in [28,29]. In [30], the problem of minimizing train delays while maximizing the total satisfaction during a traffic jam (for example, a vehicular crash or similar) is solved using a heuristic algorithm. Timetable optimization is based on the optimal departure time of vehicles from the station for each line of vehicles in order to reduce PWT. A model which minimizes PWT and distinguishes a direct vehicle transfer from walking from one station to another is solved using a heuristic algorithm, the same as in [31]. If the headway is reduced, the average PWT can be reduced as well. Minimization of the sum of headways results in minimization of the average PWT. The average PWT is equal to the ratio of the sum of headway squares and the sum of headways during which a traveler arrives. A numerical method for solving this issue is described in [32].

In order to rationalize departure intervals as much as possible, make the bus journey quicker and minimize PWT, an optimization bus schedule model is proposed in [33]. The authors in this paper consider vehicle overtaking, the limit of the vehicle's capacity and the uncertainty of passenger choice for a bus type (traditional bus or rapid bus). They propose two methods for the solution: a hybrid method of traditional PSO (HPSO) and a combination of GA and PSO named GAPSO. Table 1 summarizes a literature review and shows the details of the studies presented in this section.

Table 1. Literature review.

Paper	Year	Objective	Constraints	Solution Method	Case
[28]	2006	Minimize the waiting time	Running time, departure time, buffer time, spacing time	Linear programming	real
[19]	2008	Minimize the interchange waiting times	Running time, dwell times, trip times, headways, turnaround	B&B, heuristic	real
[22]	2010	Minimize the waiting time	Headway bounds, extra dwell times	Genetic algorithm	real
[15]	2012	Minimize the expected passenger waiting time and Minimize the discrepancy from a desired occupancy level on the vehicles	Headway bounds, vehicle capacity	A multi-objective label-correcting algorithm	real

Table 1. *Cont.*

Paper	Year	Objective	Constraints	Solution Method	Case
[25]	2012	Minimize the waiting time	Headway, in-train passengers, passenger demands, fleet-size, number of boarded passengers	Genetic algorithm	real
[24]	2013	Minimize the waiting time	Traveling time	Evolutionary algorithm	real
[31]	2014	Minimize the waiting time	Headway, departure time	Tabu Search algorithm	test
[17]	2015	Minimize the total travel time of all passengers and the energy consumption of the trains using a weighted sum strategy	Headway, train capacity	Evolutionary algorithms	test
[21]	2015	Minimize the maximal passenger waiting time	Headway, departure time, running time	Genetic algorithm	real
[32]	2015	Minimize the waiting time	Headway	Analytical	test
[33]	2017	Minimize the waiting time	Headway	Improvements of the Genetic Algorithm and Particle Swarm Optimization	test
[23]	2018	Vehicle scheduling problem with the transit assignment	Headway, the number of vehicle departures, fleet size	Heuristic	test
[18]	2019	Minimize the PWT and energy consumption	Headway	Genetic algorithm	real
This paper		Minimize the waiting time and maximize vehicles' occupancy	Headway, passenger demands, number of boarded passengers	Particle swarm optimization	test

3. Problem Statement

In order to further elaborate on the effects of departure times and the assignments of PT vehicles on PWT and VOR, a representative network with stations A, B, C, D and E is considered [23] (see Figure 1). The PT network has two terminals (a and b), two routes ($r_{a\rightarrow b}$, $r_{b\rightarrow a}$) and one transfer stop (node C). An estimated origin-destination (OD) demand matrix is shown in Table 2.

Table 2. Simple example origin destination matrix [23].

From \To	A	B	C	D	E
A	0	100	50	70	80
B	0	0	0	0	50
C	50	20	0	80	60
D	60	0	0	0	0
E	80	100	20	60	0

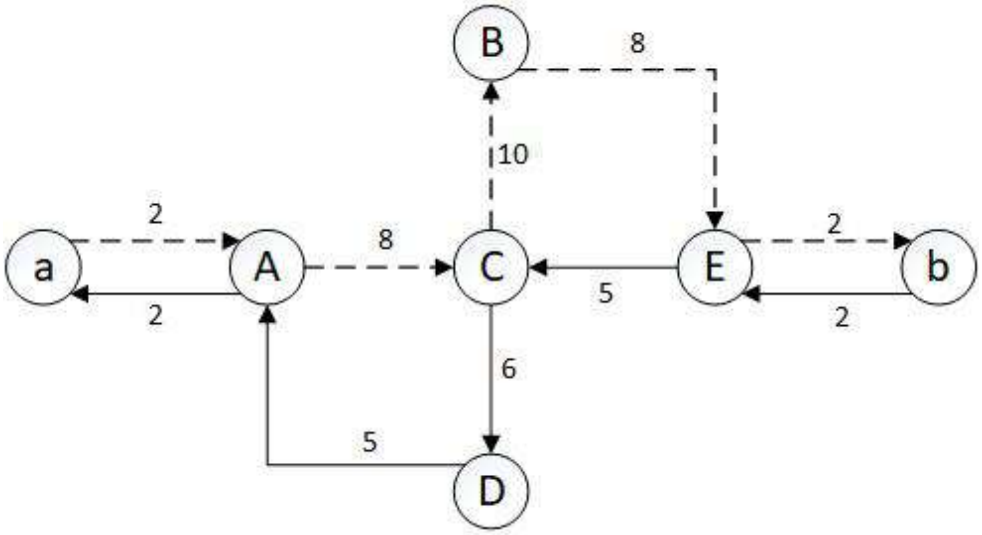

Figure 1. Example of a small network with running times (based on [23]).

The maximum load on route $r_{a \rightarrow b}$ is the load on route segment C-B (360 passengers), and the maximum load on route $r_{b \rightarrow a}$ is the load on route segment C-D (340 passengers). Assuming that a vehicle with a suitable capacity is used for service ACBE1, all 360 passengers on route segment C-B will be satisfied; i.e., transferred from their respective origins to destinations. For a case in which the vehicle does not have a suitable capacity, some passengers will be left at one of the stations, depending on the vehicle capacity. These passengers may choose to wait for the next service or use another type of transport (e.g., taxi). This decision is based upon the time needed to wait for the next service. In order to perform some example calculations, let us assume that the desired occupancy (d_o) for both routes is set to 70 passengers and that we have a given set of the number of departures per hour (four for route $r_{a \rightarrow b}$ and four for route $r_{b \rightarrow a}$). After service ACBE1 leaves Station A, 230 passengers are left behind and have to wait for the next service. Assuming that all 50 passengers for Station C get onboard at Station A, after service ACBE1 leaves Station C, 60 passengers are left behind. Considering the number of passengers left in Station A by the previous service (ACBE1), after service ACBE2 leaves Station A, 460 passengers are left behind. Assuming that all 50 passengers for Station C get onboard at Station A, after service ACBE2 leaves Station C, 120 passengers are then left behind (this number includes those passengers left behind from the previous service). The vehicle occupancy ratio for services ACBE1 and ACBE2 is the same: [1 1 1]. Each element of the occupancy ratio array is given by the ratio of the number of passengers in the vehicle to the vehicle capacity and is defined for each OD pair of the service—in this case AC, CB, BE. The amount of PWT for a given station is defined as the product of the number of passengers left behind by the previous service and the time needed for the current service or vehicle to arrive (plus an additional constant term which takes into consideration the number of passengers arriving at the station in the meantime—see Equation (11)). Hence, if the time between consecutive services is 15 min, i.e., service ACBE2 leaves 15 min after ACBE1, the amount of PWT at Station A for service ACBE2 is 3450 $passengers \cdot min$, while that of Station C is 900 $passengers \cdot min$. Assuming that the headway between services is too long, it can be assumed that the remaining passengers will find other means of transportation; this is suitable neither for the service operator nor for the passengers. A cost effective solution for the service operator would be if route ACBE were to be such that the same vehicle and crew can return and perform both services within an hour. Otherwise, if the operator sends more vehicles per time period, additional costs are incurred (more vehicles and crew members are needed). Supposing the operator has passenger cars or coaches that can be connected (assuming this is a tram or rail line), then, in this example, if the capacity is 350 ($5 \cdot 70$), no passengers are left behind at Station A, which is good. However, the vehicle occupancy ratio for both ACBE1 and ACBE2 is now $[0.8571 \quad 1 \quad 0.54286]$. The numbers of passengers

left at the stations are $[0 \quad 10 \quad 0]$ and $[0 \quad 20 \quad 0]$ respectively for the two consecutive services. For these reasons, it is necessary to make optimal decisions in order to satisfy the operators business interests and passenger needs at the same time.

4. Proposed Model

In order to describe the proposed model, an overview of the notation and variables used is provided in Table 3.

Table 3. Notation.

GENERAL	
OD	Origin-destination
OD_s	Set of OD pairs for a given service s
S_{OD}	Set of services for a given OD pair
$M = \{1, 2, \ldots, m\}$	Set of vehicles
$N = \{1, 2, \ldots, n\}$	Set of stations
L	Set of lines
$v_{s,c}$	Vehicle v with capacity c, for service s
INDEXES	
$s \in S_{OD}$	Service s from the set of services
$i \in N$	Station i from the set of stations
$v \in M$	Vehicle v from the set of vehicles
$l \in L$	Line l from the set of lines
VARIABLES	
$t^d_{s,i}$	Departure time of the vehicle at station i for service s
$t^a_{s,i}$	Arrival time of the vehicle at station i for service s
d_o	Desired occupancy
$dwell_{s,i}$	Dwelling time at station i for service s
$run_{s,i}$	Running time – traveling time between adjacent stations i and $i+1$ for service s
run_s	Sum of running time between all adjacent stations for service s
$H_{i,s}$	Headway – difference between departure times at station i for consecutive services s and $s+1$
$T_{s,i}$	Difference between departure times of vehicle v between adjacent stations i and $i+1$ for a given service s
T_i	Time horizon for i-th station
$P(v_{s,c}, i)$	Total number of passengers in vehicle v in station i
P_{ij}	Number of passengers entering the vehicle in station i heading for destination j (ij OD pair)
$P(v_{s,c}, kj)$	Number of passengers who entered the vehicle in station k with traveling to j
$P^{out}(v_{s,c}, i)$	Number of passengers exiting vehicle v of service s at station i
$P^{in}(v_{s,c}, i)$	Number of passengers entering vehicle v of service s at station i
$P^{curr}(v_{s,c}, i)$	Number of passengers in vehicle v of service s arriving at station i
$P^{stay}(v_{s,c}, i)$	Number of passengers remaining at station i after vehicle v of service s leaves the station
w_i	Average waiting time at station i
k_i	Average number of passengers per time
$Z_{i,s}$	Amount of PWT at station i for service s
η	Total vehicles' occupancy ratio for all services for a given line
$\tau_{v,s}$	Average vehicle occupancy ratio for service s

The OD demand formulation of the number of passengers is given as input data in order to calculate PWT for a given station. According to Figure 1, the OD pairs are AB, AC, AD, AE, CD, CA, CB, CE, BE, EC, EB, ED, EA and DA; while the possible lines are AC, ACB, ACBE, ACD, etc. Each line (l) has multiple services per day; for example, line ACBE has services ACBE1, ACBE2, ACBE3, etc.

Hence, a set of services is defined for each OD pair; i.e., $S_{ACBE} = ACBE1, ACBE2, ACBE3, \ldots$ Each service s depends on the:

- Departure time of the vehicle from station i $(t^d_{s,i})$;
- Vehicle assignment—$v_{s,c}$ = the vehicle v of service s with given capacity c.

The set of OD pairs for each service s is denoted as OD_s, and the set of services for each OD pair is denoted as S_{OD}.

Let us suppose that the parameters are as follows. There are six OD pairs for one line ACBE: AC, AB, AE, CB, CE and BE (i.e., $OD_{\{ACBE\}} = \{AC, AB, AE, CB, CE, BE\}$). The number of passengers entering service s at station i $(P^{in}(v_{s,c}, i))$ is lower than or equal to the sum of the number of passengers for all OD pairs in station i at departure time t in service s:

$$P^{in}(v_{s,c}, i) \leq \sum_{j=i+1}^{n} P_{ij} \tag{1}$$

where P_{ij} is the number of passengers who arrive at station i and are traveling to station j (input data from OD matrix). An inequality sign in Equation (1) signifies that is not necessary that all passengers at station i board vehicle v.

4.1. Assumptions, Variables, Parameters Which Are Time Dependent

First, a set of vehicles with different capacities is considered. The set of vehicles is marked with $M = 1, 2, \ldots, m$ (v-th vehicle is marked with index v). This is important in order to track the number of used vehicles in the network at a certain time. The time between two consecutive services in station i with the same OD pair is defined as:

$$H_{i,s} = t^d_{s,i} - t^d_{(s-1),i}, \quad \forall s \in S_{OD} \tag{2}$$

which is the objective of timetable generation and is a dependent variable of the model.

When the vehicle arrives at the final station of the network for one service, this vehicle is free to be used for another service. The traveling time between adjacent stations is defined as running time $(run_{s,i})$ for each vehicle v:

$$run_{s,i} = t^a_{s,(i+1)} - t^d_{s,i}. \tag{3}$$

For each vehicle v at each station i of the service s, the dwelling time $(dwell_{s,i})$ is defined:

$$dwell_{s,i} \leq t^d_{s,i} - t^a_{s,i}. \tag{4}$$

The time for boarding or alighting is considered as a constant for now. The difference between departure times of vehicle v between consecutive stations (running time + dwelling time) for a given service s (see Figure 2) is defined by:

$$T_{s,i} = t^d_{s,i+1} - t^d_{s,i}, \quad \forall s \in S_{OD}. \tag{5}$$

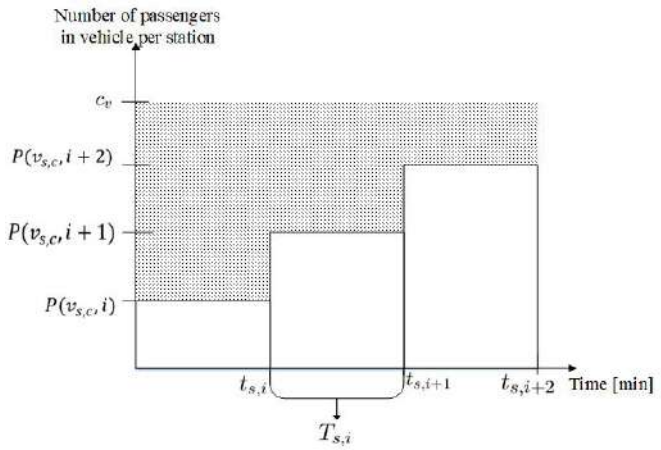

Figure 2. Vehicle occupancy depending on time, between consecutive stations for a given service s.

4.2. Assumptions, Variables, Parameters and Sets—Passengers, PWT and VOR

For each OD pair, we have passenger variables that depend on departure times for stations $(t^d_{s,i})$. The number of passengers in i-th station for j-th OD-pair (e.g., AB) for all stations for one line (at departure time $t^d_{s,i}$), for a given vehicle $v_{s,c}$ is shown in Figure 2. The total number of passengers in vehicle v in station i is defined by:

$$P(v_{s,c}, i) = P^{curr}(v_{s,c}, i) + P^{in}(v_{s,c}, i) - P^{out}(v_{s,c}, i) \tag{6}$$

where $P^{out}(v_{s,c}, i)$ is the number of passengers leaving service s at station i and $P^{in}(v_{s,c}, i)$ is the number of passengers entering service s at station i. The following assumptions are made for the number of passengers entering service s at station i: (a) The maximum number of passengers that can board the service depends on the currently available capacity of the service; i.e., the current number of free seats. (b) Passengers board the vehicle at station i according to the direct proportional distribution determined by the current number of passengers for each OD pair of station i (station i being the origin).

Those assumptions are necessary in order to simplify the calculation of Equation (6); i.e., P^{out} for a given station will always be the maximum possible for that station.

The number of passengers in vehicle v before station i (entered in station k) and with destination j (curr = current) is:

$$P^{curr}(v_{s,c}, i) = \sum_{k=1}^{i-1} \sum_{j=i+1}^{n} P(v_{s,c}, kj) \tag{7}$$

$$P(v_{s,c}, i) = \sum_{k=1}^{i-1} \sum_{j=i+1}^{n} P(v_{s,c}, kj) + P^{in}(v_{s,c}, i) - P^{out}(v_{s,c}, i); \tag{8}$$

i.e., passengers entering before station i and exiting after station i (station i excluded).

The average number of passengers at a station between consecutive services.

$$k_i = \frac{\sum_{s \in S_{OD}} P(v_{s,c}, i)}{T_i} \tag{9}$$

where T_i is the time horizon for station i. The average waiting time at station i is given by [23]

$$w_i = \frac{E[H_i]}{2} \left[1 + \frac{Var[H_i]}{E^2[H_i]} \right] \tag{10}$$

where $E[H_i]$ and $Var[H_i]$ are, respectively, the mean and variance of headway time between vehicles at station i. The amount of PWT in station i for every service (s) is given by

$$Z_{i,s} = P^{stay}(v_{s,c}, i) \cdot H_{i,s} + k_i H_{i,s} w_i, \tag{11}$$

where $P^{stay}(v_{s,c}, i)$ is the number of passengers left at station i because they could not enter service $s-1$. The amount of PWT (Equation (11)) at station i is the sum of the amount of waiting time at station i until the next service arrives at the station $(P^{stay}(v_{s,c}, i) \cdot H_{i,s}))$ and the average amount of waiting time at station i given by $k_i H_{i,s} w_i$ (for all OD pairs). It should be noted that this constant term is a measure of passengers arriving at the station between services with the average waiting time (w_i) for these passengers. This is based on the assumptions that (a) passengers randomly arrive at the station and (b) the arrivals of vehicles are uneven but occur in predetermined time intervals.

In the multiobjective optimization algorithm implemented in this paper, average normalized values of PWT and VOR are used. The average normalized amount of PWT is defined by:

$$Z_{av,norm} = \frac{\sum_{i,s} Z_{i,s}^{norm}}{n \cdot m} \tag{12}$$

where the normalized amount of PWT is defined by $Z_{i,s}^{norm} = \dfrac{Z_{i,s} - Z_{min}}{Z_{max} - Z_{min}}$ for a given station i and service s, with Z_{max} being the amount of PWT for the maximal difference between departure times of consecutive services for a given station using vehicles of minimal capacity, and Z_{min} is the amount of PWT for the minimal differences between departure times of consecutive services for a given station using vehicles of maximal capacity. n is the number of stations and m is the total number of vehicles used (i.e., number of services). Equation (12) represents the objective function for minimizing the amount of PWT.

The vehicle occupancy ratio for service s is defined by:

$$\tau_{v,s} = \frac{\sum_{i=1}^{n} P^{curr}(v_{s,c}, i) \cdot run_{s,i}}{n \cdot c_v \cdot run_s}. \tag{13}$$

where $run_s = \sum_{i=1}^{n} r_{s,i}$ is the sum of running time between all adjacent stations for service s.

The total vehicle occupancy ratio (VOR) for all services across all stations is given by:

$$\eta = \sum_{s \in S} \tau_{v,s}. \tag{14}$$

The average normalized value of η, $\eta_{av,norm}$ is defined by:

$$\eta_{av,norm} = \frac{\eta}{n \cdot m} \tag{15}$$

Equation (15) represents the objective function for maximizing VOR. The profit margin of a transport company increases as the vehicles' occupancy ratio increases.

4.3. Objective Function

The optimization problem aims at minimizing the amount of PWT while maximizing VOR. The variable PWT is used in the optimization model in order to satisfy users of PT. On the another hand, VOR (Equation (15)), used in the optimization model, is based on vehicles' occupancy for all services in time horizon. The variables included in the model are discrete so the model can be solved using combinatorial optimization methods.

The objective function of the model is defined as:

$$\min_x F(x) = (f_1(x), f_2(x)) \tag{16}$$

where x is decision variable in the solution space of dimension $2|S_{OD}|$. $|S_{OD}|$ represents the number of services in S_{OD}. The first $|S_{OD}|$ elements in x represent the departure times of the services from the first station on the line, while the second $|S_{OD}|$ elements in x represent the corresponding vehicle capacities of the services. $f_1(x)$ corresponds to the inverse average normalized value of η, i.e., $(1 - \eta_{av,norm})$ given by (15), and $f_2(x)$ corresponds to the average normalized amount of PWT, $Z_{av,norm}$, given by (12). The objective function (16) minimizes two goals and it is solved using multiobjective optimization. It minimizes $f_1(x)$ and $f_2(x)$, which correspond to maximizing VOR and minimizing the amount of PWT. The objective function of the model has the following constraints:

$$P(v_{s,c}, i) \leq c_v, \qquad \forall v \in \mathcal{M}, \forall i \in N \tag{17a}$$

$$t^d_{s-1,i} < t^d_{s,i}, \qquad \forall s \in S_{OD} \tag{17b}$$

$$T_{s,i} = t^d_{s+1,i} - t^d_{s,i}, \qquad \forall i \in N, \forall s \in S_{OD} \tag{17c}$$

$$H_{i,s} = t^d_{s,i} - t^d_{s-1,i}, \qquad \forall i \in N, \forall s \in S_{OD} \tag{17d}$$

$$k_i \leq \frac{c_v}{\sum\limits_{s \in S} H_{i,s}} \qquad \forall i \in N, \forall s \in S_{OD} \tag{17e}$$

In order to simplify the model and future calculations, it is assumed that the number of passengers in j-th vehicle cannot be more than c_v, $\forall v \in M$, as shown in constraint (17a). Constraint (17b) implies that the departure time of vehicle v in service s has to be after the previous departure time of the same OD pair served. Equation (17c) expresses the difference between departure times of vehicle v between consecutive stations in one service s. Equation (17d) expresses the difference of the departure time between two consecutive services in station i with the same OD pair. Constraint (17e) expresses that the average arrival rate for the OD pair at station i is less than or equal to the average maximum capacity rate.

5. Results

Due to the complexity of the objective function (16) and the large search space, we propose to use a heuristic optimization algorithm to determine a suitable solution. Such algorithms have shown to be suitable in such optimization problems, especially those involving large search space [34–37]. The efficiency and suitability of the various heuristic optimization algorithms in solving a whole range of complex problems have been receiving a lot of attention from academia for many years. Most of the available heuristic optimization algorithms mainly fall into two categories—swarm intelligence algorithms and evolutionary algorithms. The main representatives of these categories are particle swarm optimization (PSO) and the genetic algorithm (GA), respectively. In this paper, the multiobjective particle swarm optimization (MOPSO) algorithm, proposed in [38], was used. The algorithm in [38] extends the standard PSO algorithm to solve multiobjective optimization problems by utilizing an additional repository of particles to help the main set of particles in their search. The exploratory capabilities of the algorithm are also enhanced by a specific mutation operator that is incorporated. A preliminary comparison by the authors of this paper, of an implementation of the aforementioned algorithm, was made with a brute force search for the Pareto-optimal set solution of a model of a much simpler PT timetabling problem, and the same results were obtained. As a result, the implemented version was deemed suitable enough. A comparison of the MOPSO algorithm implemented in this paper with other PSO algorithms for multiobjective optimization, such as that proposed in [39], will be considered in future research, especially with respect to the accuracy and convergence rate in solving the proposed objective function. Additionally, a comparison of other heuristic optimization algorithms regarding their suitability in solving the proposed objective function will also be considered.

The proposed optimal solution from the Pareto-optimal set, obtained using the implemented MOPSO algorithm, was determined based on the multicriteria decision making method. The technique for order

of preference by similarity to ideal solution (TOPSIS), as proposed in [40], was used for solving traffic problems. In this section, two experimental problems based on input data and assumptions given in [23] are provided and analyzed. In both experiments, the results obtained are compared using the proposed method and assumptions and the results from [23].

5.1. Experiment 1

The first experiment consists of a simple passenger transportation line (Figure 1) involving three sets of a number of departures (q = 4, q = 5 and q = 6) for route $r_{a \rightarrow b}$ with four stations and two sets of a number departures (q = 4 and q = 5) for route $r_{b \rightarrow a}$. The input data during the given time period (7 a.m.–8 a.m.) consist of the average travel times, and the constant time needed for alighting and boarding is set to 0.5 minutes. An estimated OD demand matrix is presented in Table 2. The running times between each station for each service are presented in Figure 1. Other details of the experimental setup, which define the search space, are as follows:

- Desired occupancy of each vehicle: 70;
- Possible departure time (in minutes) at the first terminal is defined by the time intervals for each service respectively:
  - $[0, 15], [16, 30], [31, 45], [46, 60]$
  - $[0, 12], [13, 24], [25, 36], [37, 48], [49, 60]$
  - $[0, 10], [11, 20], [21, 30], [31, 40], [41, 50], [51, 60]$.

Table 4 presents the combined results of the proposed method using MOPSO algorithm and the results from the literature. The results are displayed using the following parameters: the number of passengers left at a station, waiting time and amount of PWT. Each of these are displayed as a matrix with the number of columns representing the number of stops and the number of rows representing the number of services. Based on the input data and results in [23], the departure times from the terminals a and b, needed for comparison, are presented in Table 4, in three sets for the number of departures for route $r_{a \rightarrow b}$ and two sets for the number of departures for route $r_{b \rightarrow a}$.

With respect to the MOPSO algorithm, all experiments were performed using 200 particles (population size) and 500 generations. The detailed results are presented in Table 4.

In order to compare the results, the waiting time and amount of PWT for each solution are presented. As shown, the waiting time for the next service is shorter with the MOPSO algorithm although it is an uneven timetable. Hence, the amount of PWT based on (11) is better when using MOPSO. Although VOR is maximized (it is not displayed in Table 4, but it can be deduced by looking at the number of passengers left at the station and taking into consideration the fixed desired occupancy), the amount of PWT is not acceptable because of the high number of passengers left at the station and the long waiting time for consecutive service. From the obtained results, it can be concluded that passengers will choose another type of PT. In order to provide an example with a more user-oriented PT service, Experiment 1 is expanded as presented in Experiment 2.

5.2. Experiment 2

The second experiment is a bit more complex. All the parameters and conditions are the same as in Experiment 1 with two changes: the PT line is a train or tram line and the vehicles are passenger cars or coaches each having a capacity of 70. It is also assumed that a maximum of seven passenger cars can be connected. The aim of this experiment is to reduce the amount of PWT and maximize VOR at the same time. MOPSO was used to determine the optimal parameters. With respect to the MOPSO algorithm, all experiments were performed using 200 particles (population size) and 500 generations. The proposed optimal solution was found by the TOPSIS method. The obtained Pareto-optimal set and the proposed optimal solution is displayed in Figure 3, The detailed results are presented in Table 5.

Comparing the results in Experiment 2 with those in Experiment 1, the waiting time for an uneven headway is acceptable when more than one passenger car or coach is used per service. The number of passengers left at the station is reduced, and it is assumed that passengers will wait for the next service.

Table 4. Comparison of the results obtained in Experiment 1 using the proposed method and those obtained in the literature [23].

	$d_o = 70$		Dep. Time	$Z_{av,norm}$	Number of Passengers Left at the Station			Waiting Time			Amount of PWT (Equation (11))		
$r_{a\to b}$	q = 4	Proposed method	[7:15, 7:28, 7:38, 7:46]	0.142	230 460 690 920	152 304 456 608	8 16 24 32	13 10 8 90	12 9 7 89	12 9 7 89	4386.45 5674.19 6379.35 92,467.74	2813.87 3684.52 4163.61 60,520.65	336.74 339.03 335.23 4491.29
		Results according to [23]	[7:15, 7:30, 7:45, 8:00]	0.161	230 460 690 920	152 304 456 608	8 16 24 32	15 15 15 90	14 14 14 89	14 14 14 89	5700 9150 12,600 96,300	3630 5910 8190 62,820	495 615 735 5130
	q = 5	Proposed method	[7:12, 7:24, 7:35, 7:43, 7:49]	0.126	230 460 690 920 1150	152 304 456 608 760	8 16 24 32 40	12 11 8 6 72	11 10 7 5 71	11 10 7 5 71	4239.73 6416.42 6506.49 6259.86 91,678.38	2711.84 4157.85 4239.89 4091.92 60,047.03	342.62 402.07 356.41 315.31 4359.73
		Results according to [23]	[7:12, 7:24, 7:36, 7:48, 8:00]	0.141	230 460 690 920 1150	152 304 456 608 760	8 16 24 32 40	12 12 12 12 72	11 11 11 11 71	11 11 11 11 71	4560 7320 10,080 12,840 93,600	2904 4728 6552 8376 61,200	396 492 588 684 4680
	q = 6	Proposed method	[7:10, 7:20, 7:30, 7:40, 7:47, 7:51]	0.114	230 460 690 920 1150 1380	152 304 456 608 760 912	8 16 24 32 40 48	10 10 10 7 4 60	9 9 9 6 3 59	9 9 9 6 3 59	3635.37 5935.37 8235.37 7374.76 5134.15 90,812.20	2321.22 3841.22 5361.22 4816.85 3360.49 59,527.32	302.56 382.56 462.56 379.79 249.02 4215.37
		Results according to [23]	[7:10, 7:20, 7:30, 7:40, 7:50, 8:00]	0.127	230 460 690 920 1150 1380	152 304 456 608 760 912	8 16 24 32 40 48	10 10 10 10 10 60	9 9 9 9 9 59	9 9 9 9 9 59	3800 6100 8400 10,700 13,000 91,800	2420 3940 5460 6980 8500 60,120	330 410 490 570 650 4380
$r_{b\to a}$	q = 4	Proposed method	[7:15, 7:28, 7:38, 7:46]	0.169	190 380 570 760	178 355 533 711	21 41 62 83	13 10 8 90	12 9 7 89	12 9 7 89	3680.26 4730.97 5304.77 76,778.71	3291.52 4301.94 4865.55 70,757.42	552.29 624.84 667.87 9403.55
		Results according to [23]	[7:15, 7:30, 7:45, 8:00]	0.192	190 380 570 760	178 355 533 711	21 41 62 83	15 15 15 90	14 14 14 89	14 14 14 89	4800 7650 10,500 80,100	4245 6900 9570 73,440	765 1065 1380 10,170
	q = 5	Proposed method	[7:12, 7:24, 7:35, 7:43, 7:49]	0.150	190 380 570 760 950	178 355 533 711 888	21 41 62 83 103	12 11 8 6 72	11 10 7 5 71	11 10 7 5 71	3562.43 5355.56 5414.95 5201.22 76,094.59	3171.81 4854.49 4954.54 4783.91 70,150.86	547.95 722.28 693.30 645.97 9191.68
		Results according to [23]	[7:12, 7:24, 7:36, 7:48, 8:00]	0.168	190 380 570 760 950	178 355 533 711 888	21 41 62 83 103	12 12 12 12 72	11 11 11 11 71	11 11 11 11 71	3840 6120 8400 10,680 77,760	3396 5520 7656 9792 71,496	612 852 1104 1356 9576

Table 5. Detailed results obtained in Experiment 2 using the proposed method.

				$1-\eta_{av,norm}$	Vehicle Occupancy	Number of Free Seats	Number of Passengers Left at the Station	$Z_{av,norm}$	Waiting Time	Amount of PWT (Equation (11))
$r_{a\to b}$	q = 4	Dep. time	[7:15, 7:25, 7:37, 7:46]	0.080	1 1 1 1 1 0.52 1 1 1 1 1 0.52	0 0 0 0 0 236 0 0 0 0 0 236	230 152 8 40 136 0 270 288 8 80 272 0	0.498	10 9 9 12 11 0 9 8 8 90 89 0	3348.39 2149.03 254.73 1738.06 2386.84 209.68 3373.55 3158.13 229.26 16635.48 30141.29 1572.58
		d_o	[70, 490, 70, 490]							
	q = 5	Dep. time	[7:12, 7:21, 7:32, 7:40, 7:49]	0.064	1 1 1 1 1 0.52 1 1 1 1 1 0.52 1 1 1	0 0 0 0 0 236 0 0 0 0 0 236 0 0 0	230 152 8 40 136 0 270 288 8 80 272 0 310 424 8	0.057	9 8 8 11 10 0 8 7 7 9 8 0 72 71 71	3125.07 2001.04 247.84 1729.53 2269.72 214.92 3097.84 2866.7 220.31 1775.07 3081.04 175.84 30760.54 35592.32 1982.76
		d_o	[70, 490, 70, 490, 350]							
	q = 6	Dep. time	[7:10, 7:19, 7:29, 7:36, 7:45, 7:51]	0.080	1 1 1 1 1 0.52 1 1 1 1 1 0.52 1 1 1 1 1 0.52	0 0 0 0 0 236 0 0 0 0 0 236 0 0 0 0 0 236	230 152 8 40 136 0 270 288 8 80 272 0 310 424 8 120 408 0	0.040	9 8 8 10 9 0 7 6 6 9 8 0 6 5 5 60 59 0	3245.49 2073.29 267.91 1669.51 2121.71 211.59 2778.66 2549.20 204.11 1862.56 3133.54 190.43 2621.71 3001.02 174.95 14817.07 29050.24 1269.51
		d_o	[70, 490, 70, 490, 70, 490]							
$r_{b\to a}$	q = 4	Dep. time	[7:15, 7:25, 7:37, 7:46]	0.066	1 1 1 1 1 0.63 1 1 1 0.98 1 0.6	0 0 0 0 0 154 0 0 0 10 0 196	190 178 21 30 194 0 220 371 20 0 350 0	0.060	10 9 9 12 11 0 9 8 8 0 89 0	2808.60 2513.87 419.68 1450.32 3208.65 251.61 2797.74 3999.48 368.71 8177.42 38104.84 1887.10
		d_o	[70, 420, 70, 490]							
	q = 5	Dep. time	[7:12, 7:22, 7:31, 7:41, 7:49]	0.055	1 1 1 1 1 1 1 1 0.57 1 1 1 1 1 0.60	0 0 0 0 0 0 0 0 212 0 0 0 0 0 194	190 178 21 310 323 3 80 306 0 270 484 21 40 469 0	0.061	10 9 9 9 8 8 10 9 0 8 7 7 72 71 0	2910.14 2595.88 443.11 3699.12 3641.29 236.80 1810.14 3875.88 233.11 2968.11 4524.70 354.49 10152.97 39642.32 1678.38
		d_o	[70, 140, 490, 70, 490]							

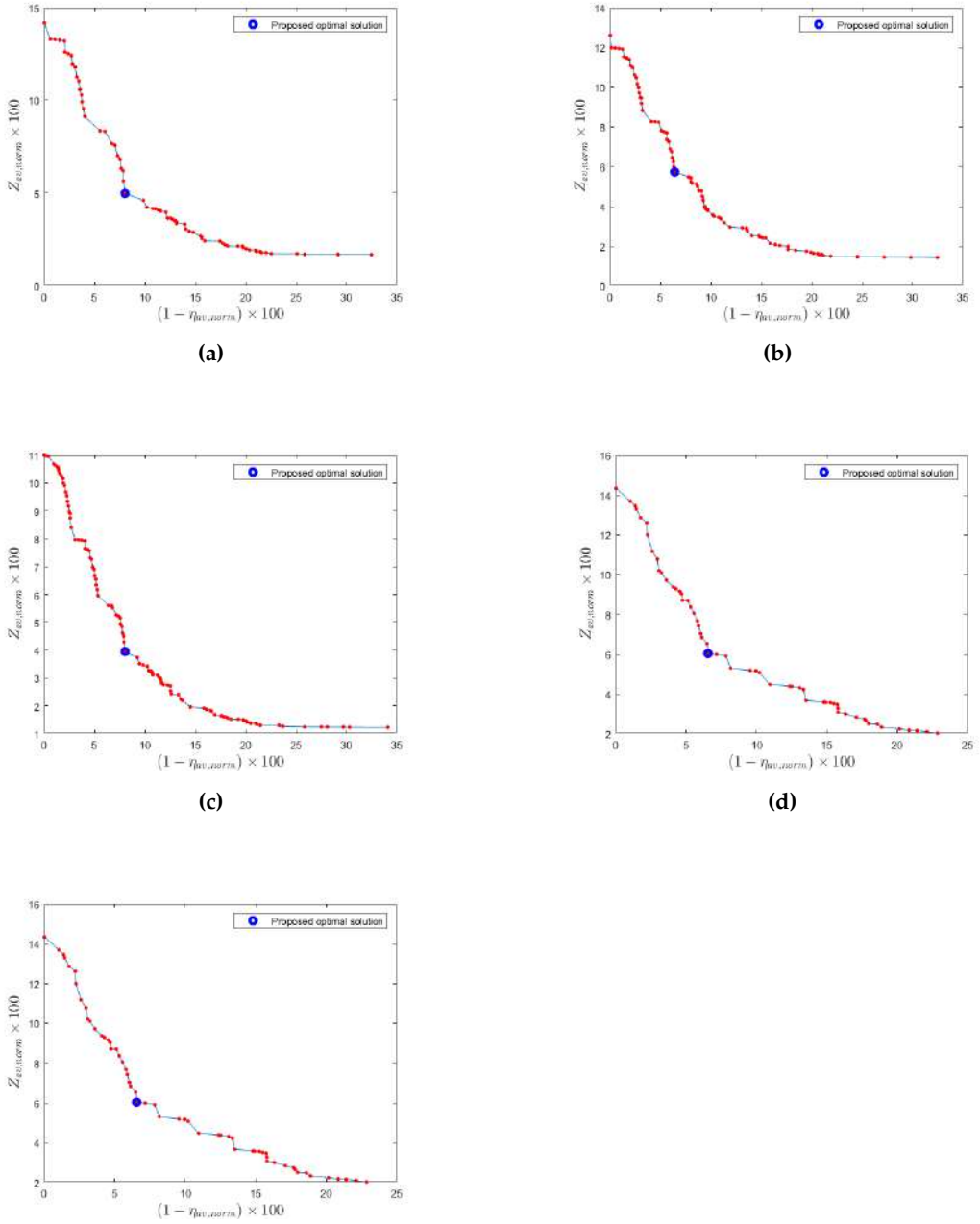

Figure 3. Obtained Pareto-optimal set and the proposed optimal solutions. (**a**–**c**) The solutions for $r_{a \rightarrow b}$ and $q = 4$, $q = 5$, $q = 6$, respectively. (**d**,**e**) Solutions for $r_{b \rightarrow a}$ and $q = 4$, $q = 5$, respectively.

6. Analysis and Discussion

In Experiment 1 (Table 4), the results indicate that by optimizing the timetable using the proposed method, an uneven timetable is obtained compared to the solution obtained by [23]. However, the amount of PWT is lower . In Experiment 2, it is assumed that the PT line is a train or tram line and that a maximum of seven coaches, each of capacity 70, can be used. The results obtained using the proposed method show a drastic improvement of the amount of PWT (i.e., a decrease in the value of $Z_{av,norm}$).

Table 6 shows the results that are obtained when the desired vehicle occupancy obtained in Experiment 2 is combined with the departure time obtained in Experiment 1. As was expected, both $(1 - \eta_{av,norm})$ and $Z_{av,norm}$ values are decreased (implying an increase in VOR and decrease in the amount of PWT) compared to the solutions obtained by [23] in Experiment 1.

Table 6. Results obtained when vehicle occupancies of Experiment 2 are combined with corresponding departure times of Experiment 1.

		$r_{a\longrightarrow b}$				$r_{b\longrightarrow a}$			
		Dep.Time	d_o	$1-\eta_{av,norm}$	$Z_{av,norm}$	**Dep.Time**	d_o	$1-\eta_{av,norm}$	$Z_{av,norm}$
q = 4	Proposed method	[7:15, 7:28, 7:38, 7:46]	70 490 70 490	0.080	0.050	[7:15, 7:28, 7:38, 7:46]	70 420 70 490	0.066	0.061
	Results according to [23]	[7:15, 7:30, 7:45, 8:00]	70 490 70 490	0.080	0.064	[7:15, 7:30, 7:45, 8:00]	70 420 70 490	0.066	0.078
q = 5	Proposed method	[7:12, 7:24, 7:35, 7:43, 7:49]	70 490 70 490 70	0.095	0.040	[7:12, 7:24, 7:35 , 7:43 , 7:49]	70 140 490 70 490	0.055	0.061
	Results according to [23]	[7:12, 7:24, 7:36, 7:48, 8:00]	70 490 70 490 70	0.095	0.048	[7:12, 7:24, 7:36, 7:48, 8:00]	70 140 490 70 490	0.055	0.073
q = 6	Proposed method	[7:10, 7:20, 7:30, 7:40, 7:47, 7:51]	70 490 70 490 70 490	0.080	0.040				
	Results according to [23]	[7:10, 7:20, 7:30, 7:40, 7:50, 8:00]	70 490 70 490 70 490	0.080	0.046				

For example, for route $r_{a\longrightarrow b}$, when four departures and the possibility of using more passenger cars or coaches are considered, the value indicating the amount of PWT (0.050) is lower than that obtained by [23] (0.064), while the value representing the vehicles' occupancy ratio (0.080) is the same for both algorithms (Table 6). If the desired vehicle occupancy is fixed, i.e., $d_o = 70$, the amount of PWT when using the proposed method is 0.142, while it is 0.161 when using the input given in [23] (Table 4). From these results, it can be concluded that it is more appropriate to use more passenger cars or coaches and an uneven timetable for the input data during the particular time period.

A qualitative analysis of the proposed timetable, with respect to the headway, is performed using the following indicators—the average headway (AH) and the expected waiting time (EWT), as recommended in [12]. AH and EWT of randomly arriving passengers, for all sets of departures and routes, are presented in Table 7. For example, for route $r_{a\longrightarrow b}$ and six departures during the given time period, the expected PWTs, when using the proposed method, are 4.45 and 4.23 for Experiments 1 and 2 respectively, while the expected PWT is 5 when using the procedure in [23]. The presented

qualitative analysis confirms that the timetable obtained using the proposed model and optimized using MOPSO is more appropriate compared to the timetable obtained in literature [23].

Table 7. Average headway and expected waiting time.

			Average Headway	Expected Waiting Time
$r_{a\rightarrow b}$	q = 4	Proposed method – exp 1 / exp 2	10.33	5.37/5.24
		[23] – exp 1	15	7.5
	q = 5	Proposed method – exp. 1 / exp. 2	9.25	4.93/4.68
		[23] – exp 1	12	6
	q = 6	Proposed method – exp 1 / exp 2	8.2	4.45/4.23
		[23] – exp 1	10	5
$r_{b\rightarrow a}$	q = 4	Proposed method – exp 1 / exp 2	10.33	5.37/5.24
		[23] – exp 1	15	7.5
	q = 5	Proposed method – exp 1 / exp 2	9.25	4.93/4.66
		[23] – exp 1	12	6

7. Conclusions

This paper presents a new model for determination of PT timetable. The model is formulated using a multiobjective optimization model to optimize VOR and PWT. Thus, this model takes into account the satisfaction of transport companies and passengers. The MOPSO algorithm is used in optimizing the model. The best solution in the Pareto-optimal set was found by the TOPSIS method. Practical implementation of the proposed model is presented using two numerical examples. PWT and VOR indices are used to present the performances of the proposed model in comparison to the similar results in the existing literature. Experiment 1 uses a simple passenger transportation line involving three sets of a number of departures (q = 4, q = 5 and q = 6) for route $r_{a\rightarrow b}$ with four stations and two sets of number departures (q = 4 and q = 5) for route $r_{b\rightarrow a}$. Experiment 2 has the same parameters and conditions as Experiment 1, and the additional assumptions that the PT line is a train or tram line and the vehicles are passenger cars or coaches. In both experiments, the proposed model using MOPSO algorithm shows better performances; i.e., shorter PWT and greater VOR. The case study based on the operation data from the existing literature shows that the proposed approach can reduce the average PWT by 10.54% for all sets with differing numbers of departures for two routes during the given time period. Based on the presented results, it can be concluded that the presented model, which uses the MOPSO algorithm to determine the optimal timetable, has an advantage in comparison to the existing models in scientific literature, which makes it suitable for scientists and practitioners in the field of PT. Further research will initially involve verification of the model using data from a real network scenario. We plan to modify the proposed model by including an extra constraint that ensures that the operation duration remains unchanged, especially if the number of vehicles or services is kept constant. We also plan to take into consideration boarding and alighting times, transfer stations and scheduling several lines.

Author Contributions: Conceptualization, I.H.T., E.K.N. and A.C.; methodology, I.H.T., E.K.N. and A.C.; software, I.H.T. and E.K.N.; formal analysis, I.H.T., E.K.N. and A.C.; investigation, I.H.T.; data curation, I.H.T. and E.K.N.; visualization, Ivana Hartamnn Tolić; writing—original draft preparation, I.H.T.; writing—review and editing, E.K.N. and A.C.; supervision, E.K.N. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

PT	Public transportation
PSO	Particle swarm optimization
PWT	Passenger waiting time
VOR	Vehicles' occupancy ratio
TTP	Train timetabling problem
WT	Waiting time
MOPSO	Multiobjective particle swarm optimization
GA	Genetic algorithm
TSP	Timetable synchronization problem
B&B	Branch and bound
OHM	Optimization based heuristic method
MIP	Mixed integer programming
PRTS	Periodic railway timetable scheduling problem
DE	Differential evolution
HPSO	Hybrid method of traditional PSO
OD	Origin-destination
TOPSIS	Technique for order of preference by similarity to ideal solution
AH	Average headway
EWT	Expected waiting time

References

1. Ceder, A. Bus frequency determination using passenger count data. *Transp. Res. Part A General* **1984**, *18*, 439–453. [CrossRef]
2. Ceder, A.; Wilson, N.H. Bus network design. *Transp. Res. Part B Methodol.* **1986**, *20*, 331–344. [CrossRef]
3. Ceder, A. Methods for creating bus timetables. *Transp. Res. Part A General* **1987**, *21*, 59–83. [CrossRef]
4. Isaai, M.T.; Singh, M.G. Hybrid applications of constraint satisfaction and meta-heuristics to railway timetabling: A comparative study. *IEEE Trans. Syst. Man Cybern. Part C (Appl. Rev.)* **2001**, *31*, 87–95. [CrossRef]
5. ShangGuan, W.; Yan, X.H.; Cai, B.G.; Wang, J. Multiobjective optimization for train speed trajectory in CTCS high-speed railway with hybrid evolutionary algorithm. *IEEE Trans. Intell. Transp. Syst.* **2015**, *16*, 2215–2225. [CrossRef]
6. Park, Y.B. A hybrid genetic algorithm for the vehicle scheduling problem with due times and time deadlines. *Int. J. Prod. Econ.* **2001**, *73*, 175–188. [CrossRef]
7. Caprara, A.; Fischetti, M.; Toth, P. Modeling and solving the train timetabling problem. *Oper. Res.* **2002**, *50*, 851–861. [CrossRef]
8. Siebert, M.; Goerigk, M. An experimental comparison of periodic timetabling models. *Comput. Oper. Res.* **2013**, *40*, 2251–2259. [CrossRef]
9. Shafia, M.A.; Sadjadi, S.J.; Jamili, A.; Tavakkoli-Moghaddam, R.; Pourseyed-Aghaee, M. The periodicity and robustness in a single-track train scheduling problem. *Appl. Soft Comput.* **2012**, *12*, 440–452. [CrossRef]
10. Bussieck, M.R.; Kreuzer, P.; Zimmermann, U.T. Optimal lines for railway systems. *Eur. J. Oper. Res.* **1997**, *96*, 54–63. [CrossRef]
11. Yalcınkaya, Ö.; Bayhan, G.M. Modelling and optimization of average travel time for a metro line by simulation and response surface methodology. *Eur. J. Oper. Res.* **2009**, *196*, 225–233. [CrossRef]
12. Ceder, A. *Public Transit Planning and Operation: Modeling, Practice And Behavior*; CRC Press, Boca Raton, FL, USA, 2016.
13. Chang, S.J.; Hsu, S.C. Modeling of passenger waiting time in intermodal station with constrained capacity on intercity transit. *Int. J. Urban Sci.* **2004**, *8*, 51–60. [CrossRef]
14. Baita, F.; Pesenti, R.; Ukovich, W.; Favaretto, D. A comparison of different solution approaches to the vehicle scheduling problem in a practical case. *Comput. Oper. Res.* **2000**, *27*, 1249–1269. [CrossRef]

15. Hassold, S.; Ceder, A. Multiobjective approach to creating bus timetables with multiple vehicle types. *Transp. Res. Rec.* **2012**, *2276*, 56–62. [CrossRef]
16. Hassold, S.; Ceder, A. Public transport vehicle scheduling featuring multiple vehicle types. *Transp. Res. Part B Methodol.* **2014**, *67*, 129–143. [CrossRef]
17. Wang, Y.; Tang, T.; Ning, B.; van den Boom, T.J.; De Schutter, B. Passenger-demands-oriented train scheduling for an urban rail transit network. *Transp. Res. Part C Emerg. Technol.* **2015**, *60*, 1–23. [CrossRef]
18. Sun, H.; Wu, J.; Ma, H.; Yang, X.; Gao, Z. A Bi-Objective Timetable Optimization Model for Urban Rail Transit Based on the Time-Dependent Passenger Volume. *IEEE Trans. Intell. Transp. Syst.* **2019**, *20*, 604–615. [CrossRef]
19. Wong, R.C.W.; Yuen, T.W.Y.; Fung, K.W.; Leung, J.M.Y. Optimizing Timetable Synchronization for Rail Mass Transit. *Transp. Sci.* **2008**, *42*, 57–69. [CrossRef]
20. Wu, Y.; Tang, J.; Yu, Y.; Pan, Z. A stochastic optimization model for transit network timetable design to mitigate the randomness of traveling time by adding slack time. *Transp. Res. Part C Emerg. Technol.* **2015**, *52*, 15–31. [CrossRef]
21. Wu, J.; Liu, M.; Sun, H.; Li, T.; Gao, Z.; Wang, D.Z. Equity-based timetable synchronization optimization in urban subway network. *Transp. Res. Part C Emerg. Technol.* **2015**, *51*, 1–18. [CrossRef]
22. Shafahi, Y.; Khani, A. A practical model for transfer optimization in a transit network: Model formulations and solutions. *Transp. Res. Part A Policy Pract.* **2010**, *44*, 377–389. [CrossRef]
23. Liu, T.; Ceder, A.A. Integrated public transport timetable synchronization and vehicle scheduling with demand assignment: A bi-objective bi-level model using deficit function approach. *Transp. Res. Part B Methodol.* **2018**, *117*, 935–955. [CrossRef]
24. Zhong, J.H.; Shen, M.; Zhang, J.; Chung, H.S.H.; Shi, Y.H.; Li, Y. A differential evolution algorithm with dual populations for solving periodic railway timetable scheduling problem. *IEEE Trans. Evolut. Comput.* **2012**, *17*, 512–527. [CrossRef]
25. Niu, H.; Zhang, M. An optimization to schedule train operations with phase-regular framework for intercity rail lines. *Discret. Dyn. Nat. Soc.* **2012**, *2012*. [CrossRef]
26. Barrena, E.; Canca, D.; Coelho, L.C.; Laporte, G. Exact formulations and algorithm for the train timetabling problem with dynamic demand. *Comput. Oper. Res.* **2014**, *44*, 66–74. [CrossRef]
27. Barrena, E.; Canca, D.; Coelho, L.C.; Laporte, G. Single-line rail rapid transit timetabling under dynamic passenger demand. *Transp. Res. Part B Methodol.* **2014**, *70*, 134–150. [CrossRef]
28. Vansteenwegen, P.; Van Oudheusden, D. Developing railway timetables which guarantee a better service. *Eur. J. Oper. Res.* **2006**, *173*, 337–350. [CrossRef]
29. Vansteenwegen, P.; Van Oudheusden, D. Decreasing the passenger waiting time for an intercity rail network. *Transp. Res. Part B Methodol.* **2007**, *41*, 478–492. [CrossRef]
30. Corman, F.; D'Ariano, A.; Pacciarelli, D.; Pranzo, M. Bi-objective conflict detection and resolution in railway traffic management. *Transp. Res. Part C Emerg. Technol.* **2012**, *20*, 79–94. [CrossRef]
31. Parbo, J.; Nielsen, O.A.; Prato, C.G. User perspectives in public transport timetable optimisation. *Transp. Res. Part C Emerg. Technol.* **2014**, *48*, 269–284. [CrossRef]
32. Berrebi, S.J.; Watkins, K.E.; Laval, J.A. A real-time bus dispatching policy to minimize passenger wait on a high frequency route. *Transp. Res. Part B Methodol.* **2015**, *81*, 377–389. [CrossRef]
33. Li, J.; Hu, J.; Zhang, Y. Optimal combinations and variable departure intervals for micro bus system. *Tsinghua Sci. Technol.* **2017**, *22*, 282–292. [CrossRef]
34. Hussain, B.; Khan, A.; Javaid, N.; Hasan, Q.U.; A Malik, S.; Ahmad, O.; Dar, A.H.; Kazmi, A. A Weighted-Sum PSO Algorithm for HEMS: A New Approach for the Design and Diversified Performance Analysis. *Electronics* **2019**, *8*, 180. [CrossRef]
35. Zuo, X.; Li, B.; Huang, X.; Zhou, M.; Cheng, C.; Zhao, X.; Liu, Z. Optimizing hospital emergency department layout via multiobjective tabu search. *IEEE Trans. Autom. Sci. Eng.* **2019**, *16*, 1137–1147. [CrossRef]
36. Wu, Q.; Zhou, M.; Zhu, Q.; Xia, Y.; Wen, J. MOELS: Multiobjective Evolutionary List Scheduling for Cloud Workflows. *IEEE Trans. Autom. Sci. Eng.* **2019**, *17*, 166–176. [CrossRef]
37. Gao, K.; Cao, Z.; Zhang, L.; Chen, Z.; Han, Y.; Pan, Q. A review on swarm intelligence and evolutionary algorithms for solving flexible job shop scheduling problems. *IEEE/CAA J. Autom. Sin.* **2019**, *6*, 904–916. [CrossRef]

38. Coello, C.A.C.; Pulido, G.T.; Lechuga, M.S. Handling multiple objectives with particle swarm optimization. *IEEE Trans. Evol. Comput.* **2004**, *8*, 256–279. [CrossRef]
39. Lv, Z.; Wang, L.; Han, Z.; Zhao, J.; Wang, W. Surrogate-assisted particle swarm optimization algorithm with Pareto active learning for expensive multi-objective optimization. *IEEE/CAA J. Autom. Sin.* **2019**, *6*, 838–849. [CrossRef]
40. Wang, Z.; Rangaiah, G.P. Application and analysis of methods for selecting an optimal solution from the Pareto-optimal front obtained by multiobjective optimization. *Ind. Eng. Chem. Res.* **2017**, *56*, 560–574. [CrossRef]

electronics

MDPI

Article

The Influence of Public Transport Delays on Mobility on Demand Services

Layla Martin [1,*], Michael Wittmann [2,*] and Xinyu Li [3]

[1] Operations, Planning, Accounting and Control, School of Industrial Engineering, Eindhoven University of Technology, 5615 AP Eindhoven, The Netherlands
[2] Chair of Automotive Technology, TUM School of Engineering and Design, Technical University of Munich, Boltzmannstr. 15, 85748 Garching, Germany
[3] Faculty of Engineering, McGill University, Montreal, QC H3A 0G4, Canada; xinyu.li@mail.mcgill.ca
* Correspondence: l.martin@tue.nl (L.M.); michael.wittmann@tum.de (M.W.)

Abstract: Demand for different modes of transportation clearly interacts. If public transit is delayed or out of service, customers might use mobility on demand (MoD), including taxi and carsharing for their trip, or discard the trip altogether, including a first and last mile that might otherwise be covered by MoD. For operators of taxi and carsharing services, as well as dispatching agencies, understanding increasing demand, and changing demand patterns due to outages and delays is important, as a more precise demand prediction allows for them to more profitably operate. For public authorities, it is paramount to understand this interaction when regulating transportation services. We investigate the interaction between public transit delays and demand for carsharing and taxi, as measured by the fraction of demand variance that can be explained by delays and the changing OD-patterns. A descriptive analysis of the public transit data set yields that delays and MoD demand both highly depend on the weekday and time of day, as well as the location within the city, and that delays in the city and in consecutive time intervals are correlated. Thus, demand variations must by corrected for these external influences. We find that demand for taxi and carsharing increases if the delay of public transit increases and this effect is stronger for taxi. Delays can explain at least 4.1% (carsharing) and 18.8% (taxi) of the demand variance, which is a good result when considering that other influencing factors, such as time of day or weather exert stronger influences. Further, planned public transit outages significantly change OD-patterns of taxi and carsharing.

Keywords: carsharing; data analysis; delays; demand; public transit; taxi

Citation: Martin, L.; Wittmann, M.; Li, X. The Influence of Public Transport Delays on Mobility on Demand Services. *Electronics* **2021**, *10*, 379. https://doi.org/10.3390/electronics10040379

Academic Editor: Juan M. Corchado
Received: 15 January 2021
Accepted: 28 January 2021
Published: 4 February 2021

1. Introduction

Metropolitan areas suffer as a consequence of a car-centric city layout. Roads are frequently congested [1], air quality decreases [2], and valuable space for active mobility is restricted [3]. Thus, city planners aim to incentivize as many travelers as possible to use rail and public transit. As travelers minimize their own transportation cost—comprised of travel time and fare—[4], it is paramount to understand the impact of the different factors of the cost function. The travel time depends on the scheduled transit time and delays. While the scheduled transit time is usually comparatively low, delays can severely impact both the actual and the perceived travel time [5]. Nowadays, public transit competes with rising MoD services [6], such as ride-hailing and carsharing, since vehicle ownership is lower in metropolitan than in rural areas ([7], p. 35). While delays also occur in MoD systems, they do not propagate as severely as in public transit, due to missed connecting trains, and they are perceived less severe by travelers [8]. Thus, surveys indicate that users switch to road-based individual transportation if delays increase or public transit is unavailable [9], resulting in additional demand. This additional demand (i) increases road congestion and (ii) results in additional planning effort for MoD operators. They must react by moving vehicles to locations with increased demand (rebalancing, dispatching), and

they may have to increase their fleet size to accommodate those peak demands. Otherwise, the unserved demand increases, which results in lost sales on a short horizon [10], and they may also influence customer satisfaction and retention on a longer horizon [11]. The changing demand patterns during the COVID-19 pandemic pose an additional challenge, but they also permit us to study the system under a different demand profile.

This paper studies the influence of public transit delays on the demand of MoD services. As such, it helps MoD operators to increase the quality of service by improving their demand predictions and, consequentially, their operational strategies (e.g., [12]), and gives public transit authorities first insights into how their delays impact road traffic, and eventually congestion. A better understanding of the influence of public transit delays—and, thus, the quality of service of public transit—eventually helps to carry over a demand prediction tool that was developed in one location to another (as [13] attempt for bikesharing services).

This work is the first data-driven approach for establishing the correlation between public transit delays and demand for carsharing and taxi services. From the data, we establish a lower bound of how much taxi and carsharing substitute public transit by measuring the number of additional taxi and carsharing trips for increasing delays, both over a period of 10 months and exemplarily for closures of the main tracks, and both for individual stations and the entire city center.

This suggests that

- a lower delay and/or higher coverage of the public transit system can result in a smaller necessary fleet (by up to 2.2% and up to 0.5% of the total fleet size, respectively), and most likely result in fewer traffic; and,
- carsharing and taxi operators can improve their demand prediction performance by including information on planned public transit closures.

In the following, we first review related work in Section 2, and describe the data collection process and general statistics in the data presented in Section 3. Section 4 analyzes the data to establish how demand MoD-systems and public transit delays are connected. Section 5 discusses the results and concludes the paper.

2. Related Work

This work is related to two different streams of literature: demand prediction for carsharing and taxi services, as well as substitution and complementarity between different modes of shared and public transit.

[14] review different works focusing on demand prediction for carsharing. They conclude that research still lacks in-depth knowledge about the intricacies of demand processes. [15] investigate how taxi, Uber, and Lyft demand increases during severe rain in New York City, as well as the price elasticity of Uber and Lyft during those periods. They find that the total number of rides increases, but the number of taxi rides is only weakly correlated with rain, which suggests that the additional demand is due to Uber and Lyft's pricing strategy. The impact of weather on demand is clearer than the influence of public transit delays, since rain can be modeled as a Boolean variable, while the demand increase may depend on the extent of the delay. [16] study how events influence taxi demand in New York City using online information (web mining demand hot spots from social media). They find that the frequency that events were mentioned has a significant impact on the taxi demand. However, the influence of events differs from the influence of public transit delays, as events only affect the origin of a trip, not origin-destination pairs, and since events are known earlier than delays. [17] show that this event information can improve the performance of demand predictors. Thus, we cannot directly adapt models for measuring the influence of weather or events to the influence of delays. [18] show that socio-demographic features have an impact on carsharing demand in Munich and Berlin (Germany), [19] study the impact of these factors on New York City taxi demand, and [20] investigate the impact of socio-demographic factors on ride-hailing demand in California. All three studies find that socio-demographics can explain parts of the demand variances. However, socio-demographics usually do not change within the observation period, unlike

public transit delays. Therefore, we must develop a new methodology to measure the impact of a cardinal variable that varies during the observation period, but it has strong correlation with some of the previously studied variables.

Ref. [19] also explore the influence of demographic and socioeconomic factors on the taxi passenger demand in New York. The results clearly indicate that a relationship between public transit accessibility and taxi demand exists. Taxi trips in this study occur more often if public transit is more accessible. The authors note that a finding of whether this relationship is competitive or complementary could not be determined from their results. Additionally, they do not consider if and how taxi demand varies, depending on the temporal availability of public transit. Ref. [21] study the demand patterns for taxi and Uber on a coarse-grained level. They find that similar external factors impact the demand for taxi and Uber. Ref. [22] state that users of free-floating carsharing are more likely to have a public transit subscription than the control group which suggests complementarity. Ref. [23] consider the spatio-temporal availability of public transit, but do not extend their analysis beyond examples during an outage at a central location. Naturally, such a dependency implies that if the required time for one of the transit modes increases, the demand for this mode should decrease. Ref. [23] gives some examples that support that public transit outages increase the demand for alternative transportation modes in the city of Vancouver, but do not extend this to a city-wide experiment over a longer period of time. Ref. [24] observe that, for the most common carsharing trips (origin-destination pairs) in Madrid, traveling by car is only slightly faster than public transit, but significantly more expensive, which suggests a customer preference for traveling by car, but they do not investigate whether the number of trips increases even further if public transit becomes less available (due to outages or delays). For early carsharing adopters, ref. [25] find that, on average, over seven European and North American cities, 40% of carsharing trips could have been performed more quickly by public transit, which suggests that carsharing, in fact, substitutes public transit. However, they do not include delays or outages, as well as the waiting time for the next train. Several studies indicate that, on the customer side, there is a notable difference in the perceived and real waiting time—especially in the case of unestimated delays. Ref. [26] found that passengers who did not have knowledge about the actual schedule perceived waiting times significantly longer than passengers with knowledge. Ref. [27] confirm that users already more frequently decide against using public transit for an entire trip if only a single trip segment is sub-optimal (e.g., long waiting times, slow transit). Ref. [28] study the modal choice of travelers in Taiwan using a general estimation equation, and find that intermodal transportation accessibility has a positive influence of public transit ridership at those stations where available. However, ref. [28] do not study the influence of the public transit operator's availability and punctuality on ridership (and alternative choices).

This paper is the first to analyze the influence of public transit delays and outages on MoD demand in a data-driven fashion (as compared to simulation- and survey-based research), both on a city level and more granular per public transit station (when compared to single occasions). This provides substantial additional insights, but it also requires a new methodology.

3. Reference Datasets

This study uses three different datasets for the city of Munich in the period from May 2019 to March 2020: downtime and delays for public transit, vehicle movement data of a major carsharing provider (only until December 2019), and taxi customer trip data of a local taxi agency. The data are discretized and filtered.

3.1. Public Transit Data

We query current departures (scheduled and actual departure time) of all suburban railway (S-Bahn) and underground (U-Bahn) lines at all stops in 5 min. intervals to obtain an understanding about delays and outages in the public transit system. The delay δ_{it} can

then be calculated as the difference between the scheduled and actual departure time. Both of the times are only given in minutes, thus delays are only reported if they are at least 60 s, and delays are rounded down to the next integer. For our analysis, longer delays are more relevant, reducing the impact if rounding down. Throughout this paper, we indicate how we handle integrality. For each station i and time frame t, we report both the average delay δ_{it} and maximum delay $\max(\delta_{it})$ of all departures. Outages are derived from comparing the number of departures in a given interval to comparable intervals on other days (maximum over all weeks). Additionally, we collected information regarding track closures during the analysis period from public authorities.

Table 1 lists the basic statistics on the data set and the delays. Even in the short observation period of $\approx$ 10 months, the number of departures is in the order of 12 M (This number includes all departures of the same train and, thus, may seem to be excessively high at first). Thus, the impact of those instances with a very high delay is negligible. The mean delays (in minutes) are low (slightly above or below 1 min.), and the majority of departures is not delayed (contrary to what customers perceive). Figure 1 depicts the average of delays (suburban railway and underground) during an example week (week 42/2019, 14–20 October) at station Marienplatz. During rush hour, most of the lines are slightly delayed, and the morning rush hour incurs more severe delays than the evening rush hour. Major delays (>6 min.) occur infrequently, and on a more random pattern.

Table 1. Data Description Public Transit.

	Suburban Railway	**Underground**
# stations	125	98
# lines	7	8
# records	4,566,500	7,683,766
mean delay in min	1.168	0.837
max delay in min	354	1431
# records $1 >$ delay	2,131,051	3,890,972
# records $3 >$ delay ≥ 1	1,601,061	2,894,220
# records $6 >$ delay ≥ 3	618,356	759,421
# records delay ≥ 6	216,032	139,153

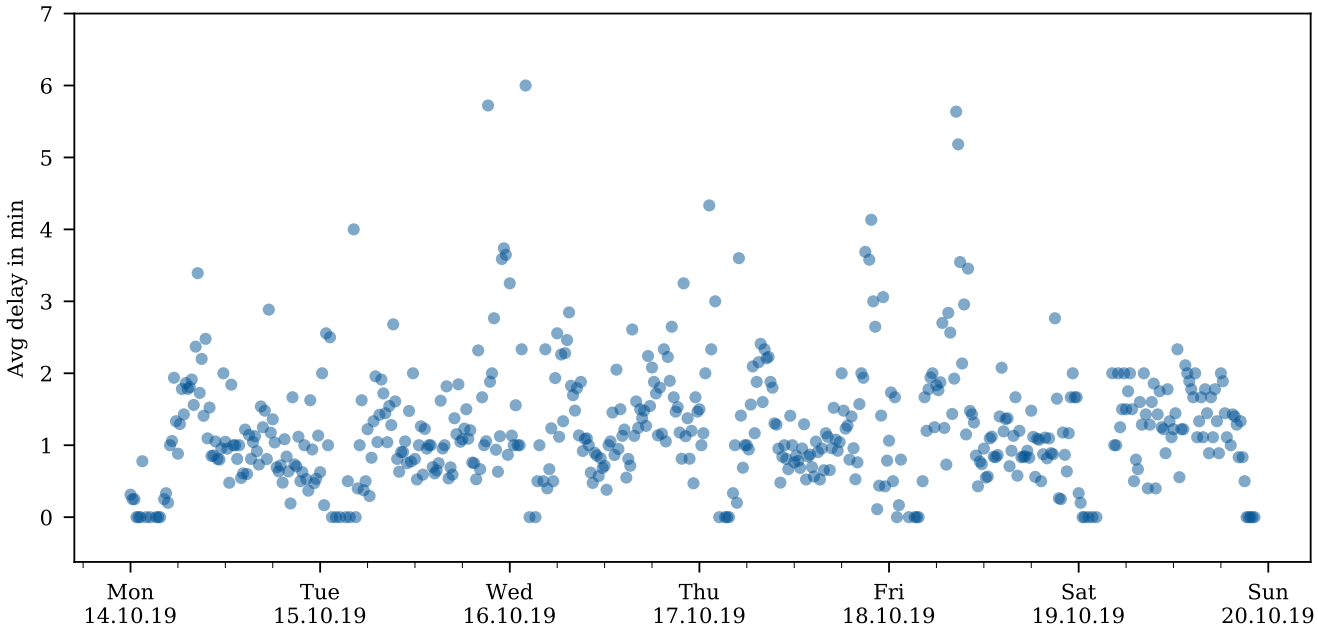

Figure 1. Average of Delays at Marienplatz during Example Week.

If an above average delay occurs at some point in time t, there is a high probability (>55%) that there will also be an above average delay at time $t + 15$ min.. If the delay at time t is in the 80th percentile ($\%^{\tau(t)}(\delta_{ti}) \geq 0.8$, where $\%^{\tau(\cdot)}(\cdot)$ is the percentile function that assigns the delay percentile among comparable time frames ($\tau(t)$) at station i), above average delays occur even more frequently, and delays persist longer. Figure 2 depicts the delay persistency, i.e., the probability that a delay occurs at the same station a given time interval after another delay.

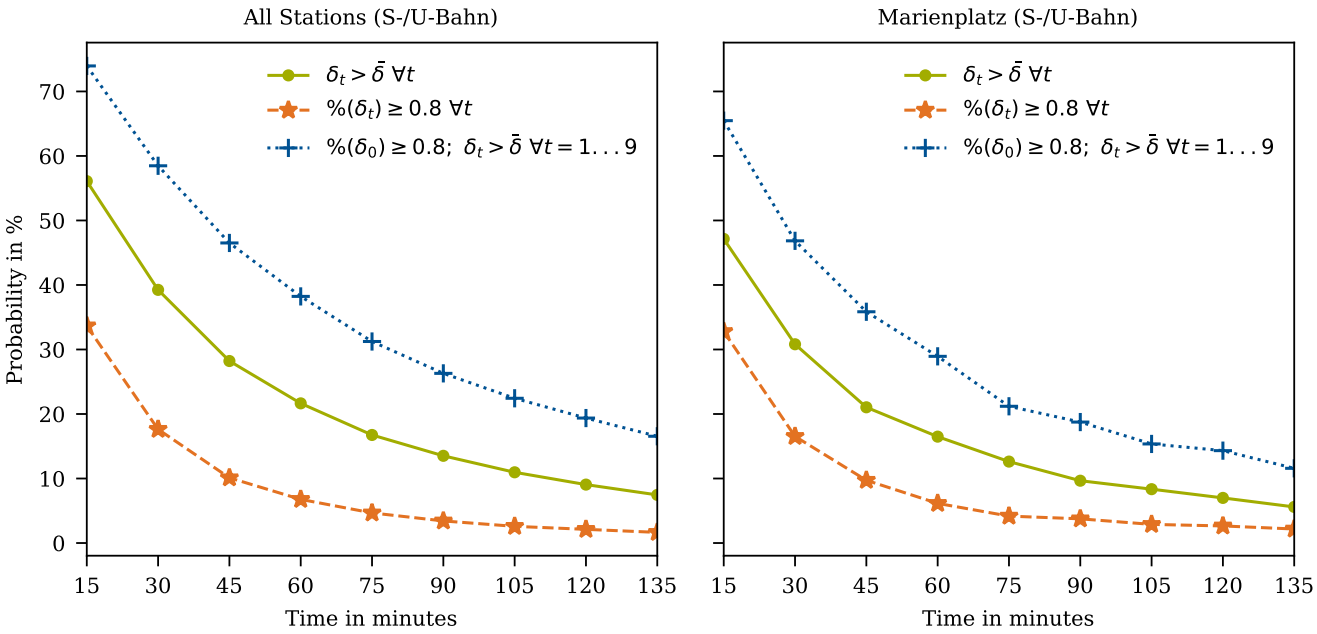

Figure 2. Delay Persistency in Munich and at Marienplatz.

When comparing the entire Munich city center to Marienplatz, it, at first, seems surprising that the central location with a very dense time table and high utilization has a lower delay persistency, but Marienplatz also has four tracks while most stations only have two tracks. Delays propagate almost independently on different tracks. This delay propagation is visible from Figure 3.

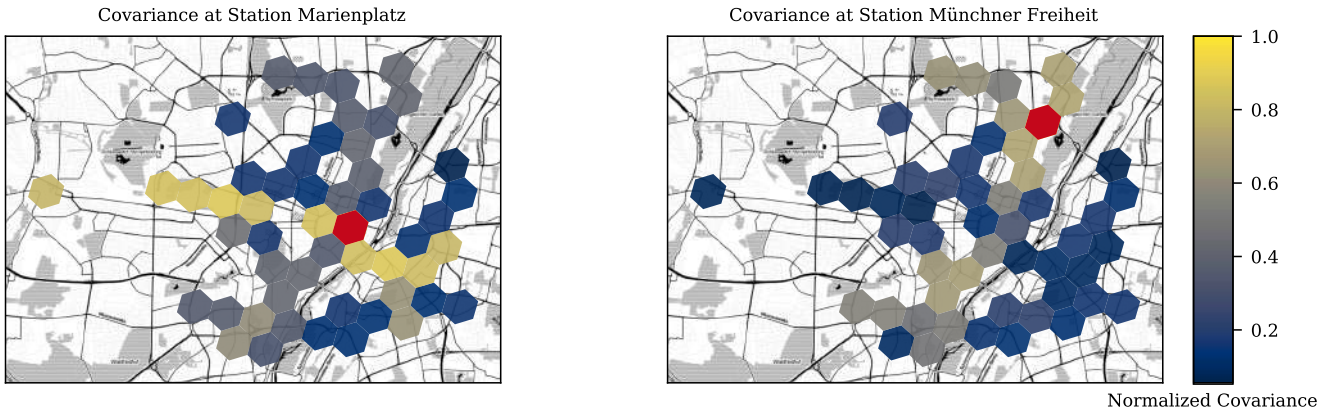

Figure 3. Delay Propagation in the Public Transit Network.

It shows the probability that an above average delay at Marienplatz (left) or Münchner Freiheit (right) correlates with an above average delays at other stations. Yellow shading refers to a high covariance, blue shading to a low covariance. Clearly, stations along the same line have a higher probability of delays and, at Marienplatz, this mainly affects

the East-West connection (S-Bahn), not so much the North-South connection (U-Bahn). The covariance between the selected station and stations along other lines is low, and the remaining covariance can be due to external influences, customers transferring lines, or intersecting lines. From the strong spatial and temporal differences in delays, we conclude that our spatial and temporal resolution is reasonable.

3.2. Carsharing Data

Carsharing data for one of the largest Munich free-floating carsharing operators has been collected while using webscraping techniques since April 2018. The scraping ended mid-January 2020 when the API was discontinued. Every 5 min., the current location of all available vehicles (not rented or reserved by customers, or blocked by the operator) was recorded. Because of the data collection method, outages appear, and the data are cleared accordingly.

Movements of vehicles are created by recording the last location of the vehicle before becoming "invisible" and the first location after re-appearing in the data stream. The data collection method does not allow for us to differentiate between customer trips and rebalancing operations, but there should be significantly less rebalancing operations than customer trips, and the literature reports that rebalancing mainly occurs during the night [29]. We remove time windows during the night, as described later.

The data set contains >1.5 M trips. Roughly 20% of all data points are missing due to outages in the data collection. The reasons for outages include power outages, network connection loss, or, otherwise, discontinued service on the collection server, unavailability of the API, and service downtime of the carsharing service provider. On average, 131 trips occurred per hour, with a maximum of 653 trips during an one-hour interval, and the number of trips highly depends on the time of day and weekday.

Figure 4 shows an example weekly pattern for week 42/2019 (14–20 October).

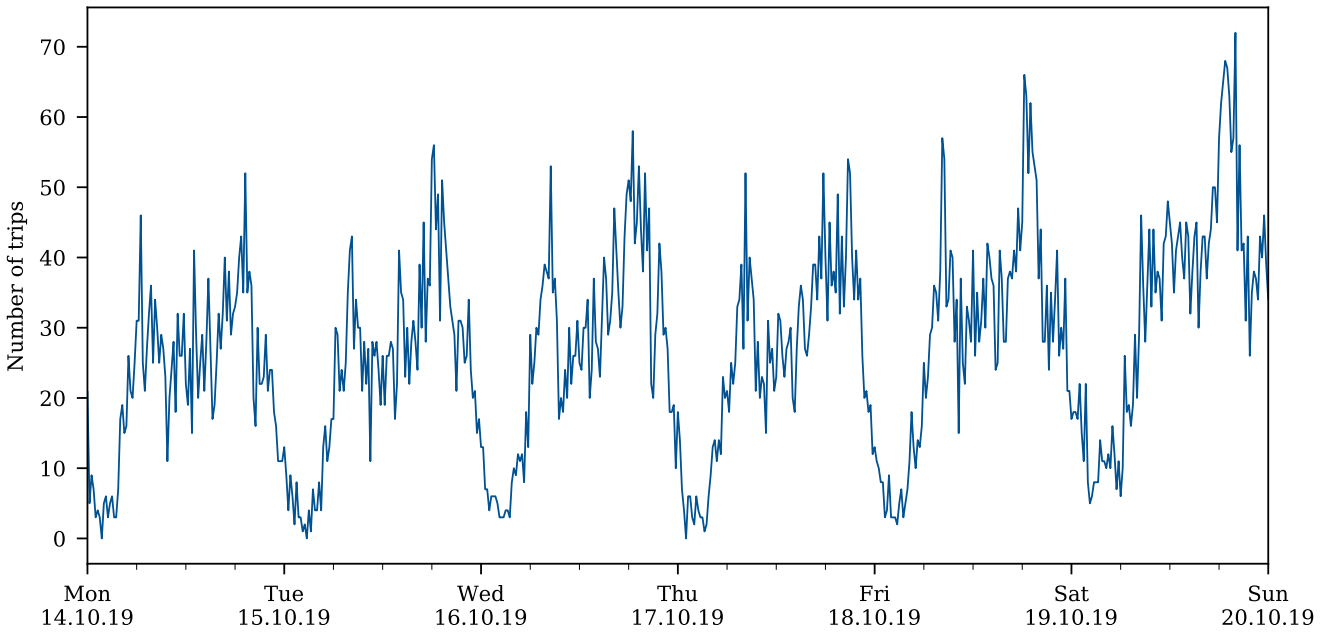

Figure 4. Number of Carsharing Trips during Example Week.

Demand is aggregated in one-hour intervals and it is reported at the beginning of the interval. The demand follows a daily pattern with more demand during the evening rush hour than the morning rush hour, and slightly decreasing demand during the course of the week.

3.3. Taxi Data

This study makes use of floating car data from a local taxi agency to derive the passenger demand for taxi services in Munich. A fleet of 550 taxis served ≈10 M customer trips between 2015 and 2020. The data are being continuously retrieved from the fleet management interface, which is usually used for dispatching by the local taxi agency [30]. The data are directly provided by the dispatching agency with full information about trip start, trip end, and driven route.

The data set contains >3.8 M trips in the observation period between April 2018 and March 2020. On average, 252 trips are recorded per hour (at most, 740 trips per hour).

Figure 5 shows an example weekly pattern for week 42/2019 (14–20 October).

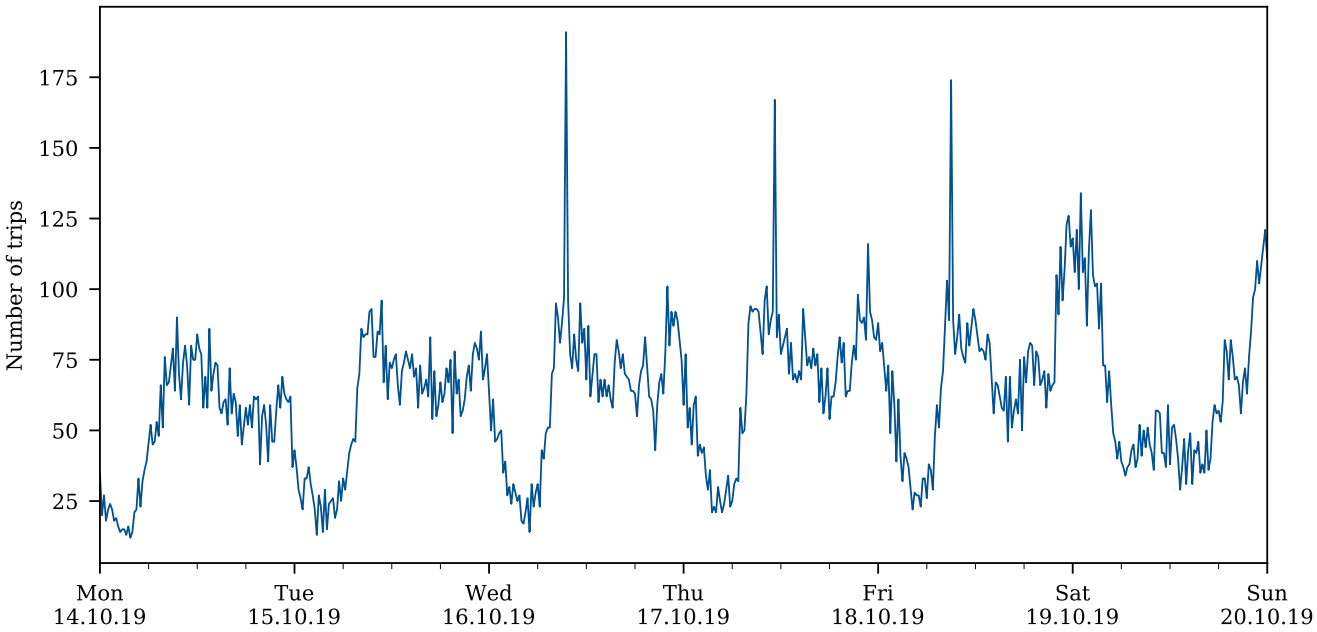

Figure 5. Number of Taxi Trips during Example Week.

The demand pattern follows the same high level trend as carsharing demand, but the highest demand peaks occur on the weekend rather than at the beginning of the week. The morning peak is more pronounced, and the daily afternoon demand peaks occur slightly later in the day than in the carsharing system.

3.4. Data Discretization and Filtering

For comparability, we discretize the area in hexagons with an edge length and radius of 461 m using Uber's Hierarchical Spatial Index H3 [31]. Imposing a maximum walking distance of 461 m is in alignment with literature [32]. Temporally, we discretize the carsharing and taxi data into one-hour intervals on a rolling scheme, creating data points every 15 min. Every data point then contains the total number of trips—the demand d_{it}—occurring in the 60 min. after a delay. One hour is a reasonable time frame for potential impacts and in line with delay persistency (Figure 2). This (i) increases the amount of available data points and, therefore, reduces random variances of our results compared to one-hour intervals without rolling time windows and (ii) smoothes the demand pattern as compared to 15-min. intervals, as one can see in Figure 6 for the carsharing and taxi demand.

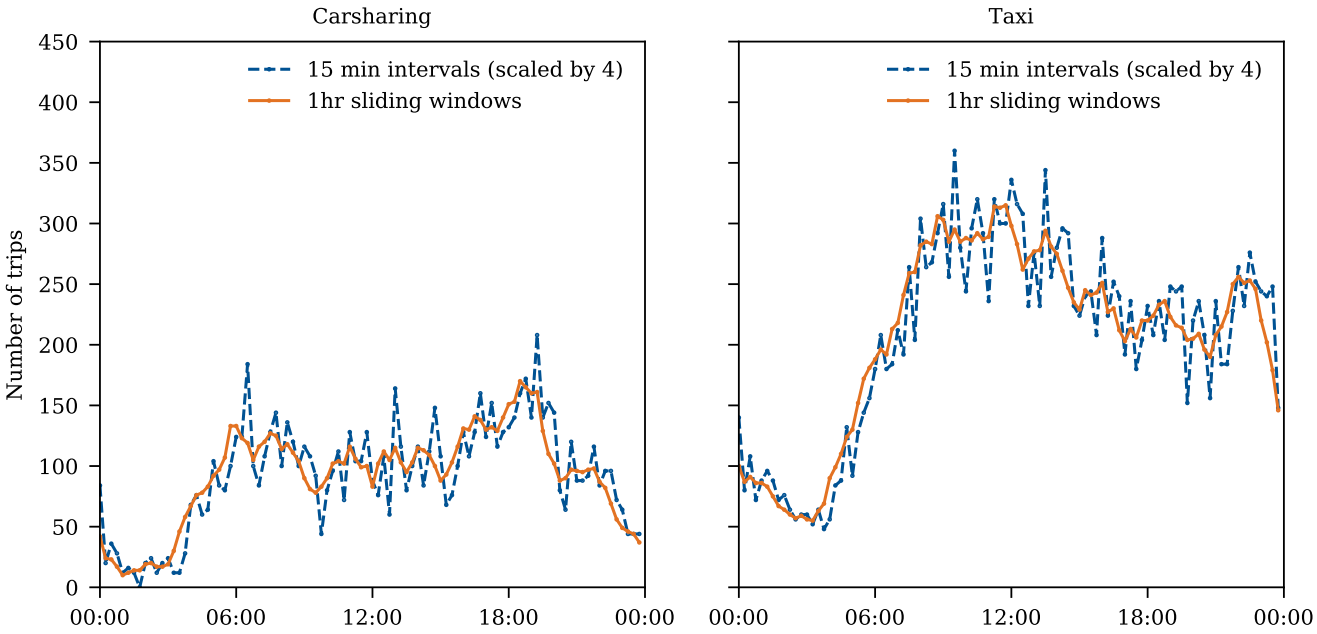

Figure 6. Number of Carsharing and Taxi Trips during Example Week with and without Smoothing.

Because public transit delays persist for some time and, as these delays take time to manifest in the taxi demand, using sliding time windows is also advantageous to be able to record longer-time impacts on the demand.

Obviously, carsharing and taxi demand can only be evaluated against public transit delays if the public transit service is scheduled to run. Further, an influence can only be measured to a statistically significant level if the average number of carsharing and taxi trips is sufficiently high. This does not exclude temporary outages, but it does exclude nights, as public transit is not operating between 1:30–4:30 AM, and the number of departures decreases substantially during the late evening. We exclude the time frame 10:00 PM–5:00 AM, to be safe against startup and end-of-horizon effects, and the low demand during the night. Omitting this longer period of time also makes it more probable that vehicle movements in carsharing are customer trips, rather than vehicle rebalancing (since vehicle rebalancing in carsharing systems mainly occurs during the night [29]). Subsequently, 13.0% of all public transit departures, 14.6% of all carsharing trips, and 31.6% of all taxi trips are omitted.

Additionally, to be able to measure the effect of public transit on carsharing and taxi demand, we exclude those hexagons without a public transit stop (suburban railway or underground) and those outside the Munich city highway "Mittlerer Ring" (except for Pasing station which is the west-most end of the suburban railway main tracks). Thus, we consider demand in the 53 hexagons that are depicted in Figure 7 with blue squares for U-Bahn stations, green dots for S-Bahn stations, and orange rhombuses for stations with both U-Bahn and S-Bahn connections.

Figure 7. Considered Hexagons (in grey shading) with Suburban Railway (green dots), Underground (blue squares) stations, or a Combination of both (orange rhombuses).

3.5. Censored Demand

Demand for MoD is subject to censoring [33,34]: if no vehicle is available, one cannot record demand, and a straight-forward model tends towards underestimating demand. Outages of the carsharing service might correlate with public transit disruptions. Thus, none of the approaches that have been suggested in literature can be applied, since we measure increased delay which also impacts the demand censoring. Instead, we split the data set in those data points with and without censored demand (assuming that censoring can apply only if supply is 0). This occurs more frequently in remote locations with low demand. For the high demand location Marienplatz, censoring might have occurred in up to 24% of all data points. Such censored demand only occurs for the carsharing service, but not the taxi service, as taxi street-hailing is less common in Munich [35], and as Munich has a significant oversupply in taxis [36].

4. Analysis

Using the previously described data sets, we analyze how public transit delays influence demand for MoD services. In particular, we give a high level relation, calculate the fraction of demand variation that can be explained by public transit delays, analyze the varying demand patterns during outages, the probability of having no vehicles available depending on delays and outages, and the demand changes during the COVID-19 pandemic. We use this analysis to estimate the additional demand, traffic, and necessary increase in the fleet size due to delays and outages.

All of the numerical analyses are implemented in Python 3 with (among others) Numpy, Scikit Learn, and Gurobi. The experiments are performed on an Ubuntu server.

4.1. High Level Relation

Station-timeframe tuples with high delays (mean delay $\delta_{it} \geq 3$ min.) more frequently result in high taxi and carsharing demand than tuples with lower maximum delays. Tuples are clustered by the observed demand relative to the mean for this station-timeframe tuple (in 2% bins), and the observed maximum delay (no delay, up to 3 min. delay, and higher delays).

Figure 8 reports the relative frequency for each tuple by means of a cumulative distribution function (CDF). It is obvious that the higher the delay, the more frequent high demand instances appear. While this indicates some dependency, it does not yet show how delays and demand correlate.

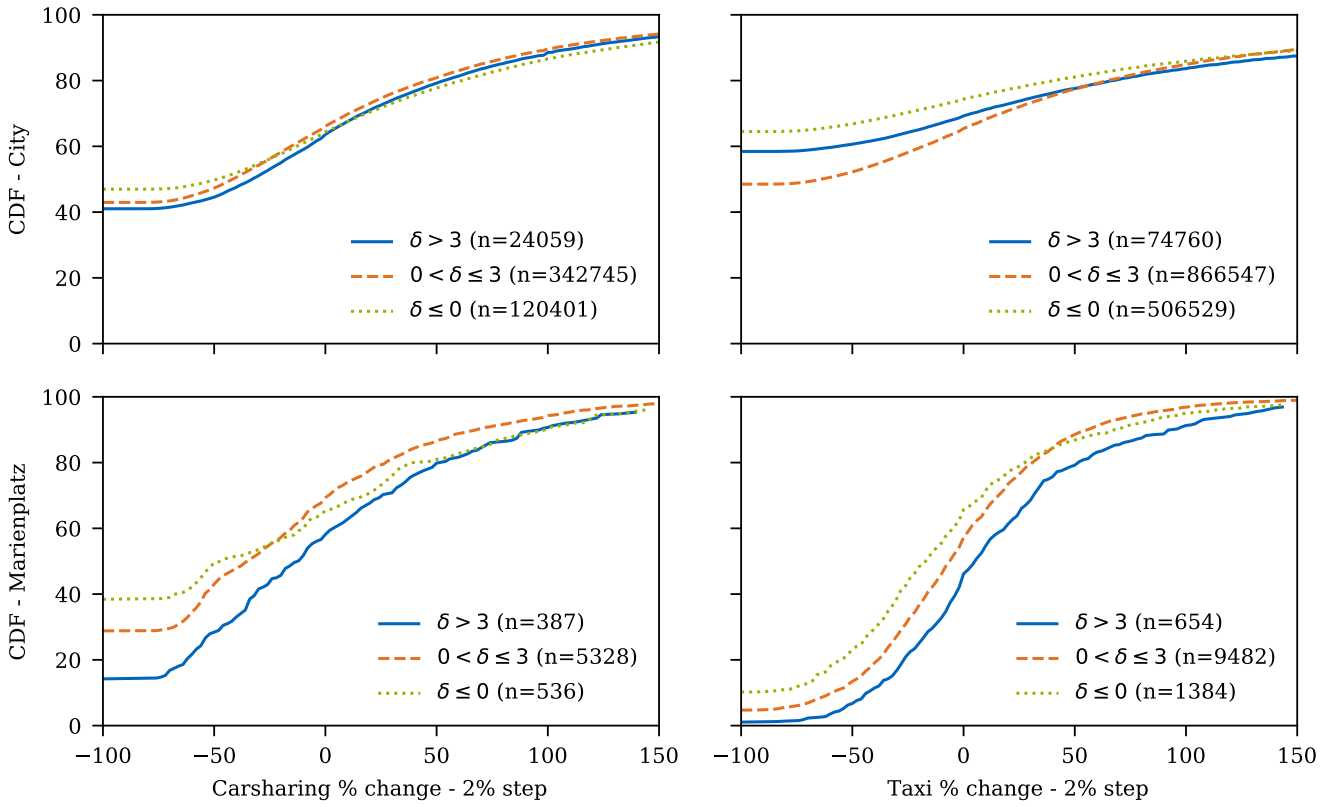

Figure 8. Maximum Delay in the Entire City and at Marienplatz.

To this end, we further observe that any increase in public transit delays entails an increase of the demand for taxi and carsharing services. We group the data points by delay (in intervals: [0, 1), [1, 3), [3, 6), [6, 10), [10, 20), [20, 60), [60, ∞), with the last two being aggregated for carsharing due to a low number of data points). The lower number of high delay data points is a consequence of filtering potentially censored demand points in the carsharing data. Figure 9 depicts the boxplots of trip deviations from mean for carsharing and taxi. For ease of exposition, the boxplots do not contain outliers. The trip deviation from mean increases from 0.0% to 13.5% for carsharing, and −2.1% to 50.0% for taxi.

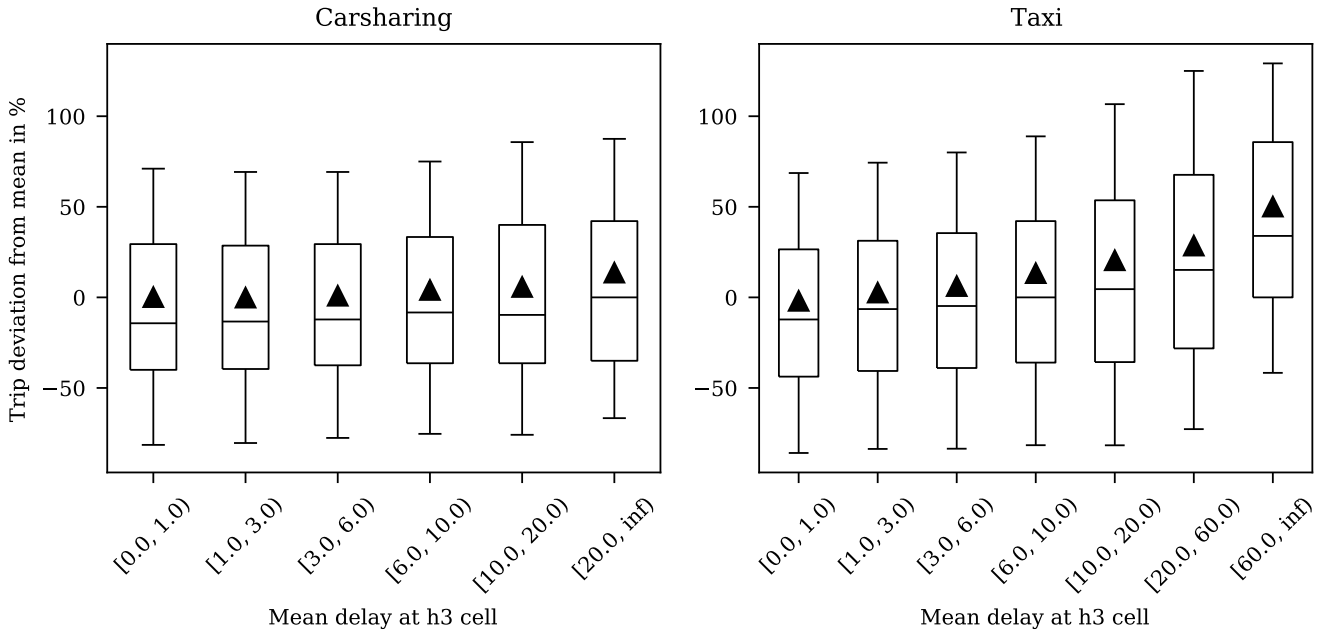

Figure 9. Increased Demand during Delays.

Thus, it is clear that MoD demand and public transit delay are correlated, even though other external factors (time of day, station, weather, events) also exert a strong influence.

4.2. Explained Demand Variance Due to Delays

We showed that the demand for MoD increases if delays occur. However, delays are not the only factor that can explain variances in the MoD demand, and some randomness is intrinsic to the system. In order to measure the explanatory power of delays on MoD demand, we assume that demand at a station during a given timeframe can be predicted using the mean value as a baseline, and measure how much the variance decreases when correcting the trip counts for the delay. Therefore, we filter the dataset for observations in which a mean delay $\geq$3 is observed. Thus, we define a lookup function $f(\delta_{it})$, which returns the mean percentage trip deviation for each delay bin.

The data points are transformed into the deviation dev_{it} from mean $\mu_{i\tau(t)}$ for location i and timeframe t.

$$\text{dev}_{it} = \frac{d_{it}}{\mu_{i\tau(t)}} - 1$$

resulting in the "basic" data set S, and potentially corrected by

$$\hat{\text{dev}}_{it} = \frac{d_{it} \cdot (1 - f(\delta_{it})) - \mu_{i\tau(t)}}{\mu_{i\tau(t)}}$$

resulting in the "corrected" data set $\hat{S}$.

Each set of data points S and $\hat{S}$ can be represented as a density function. The histograms for the density functions can be found in Figure 10.

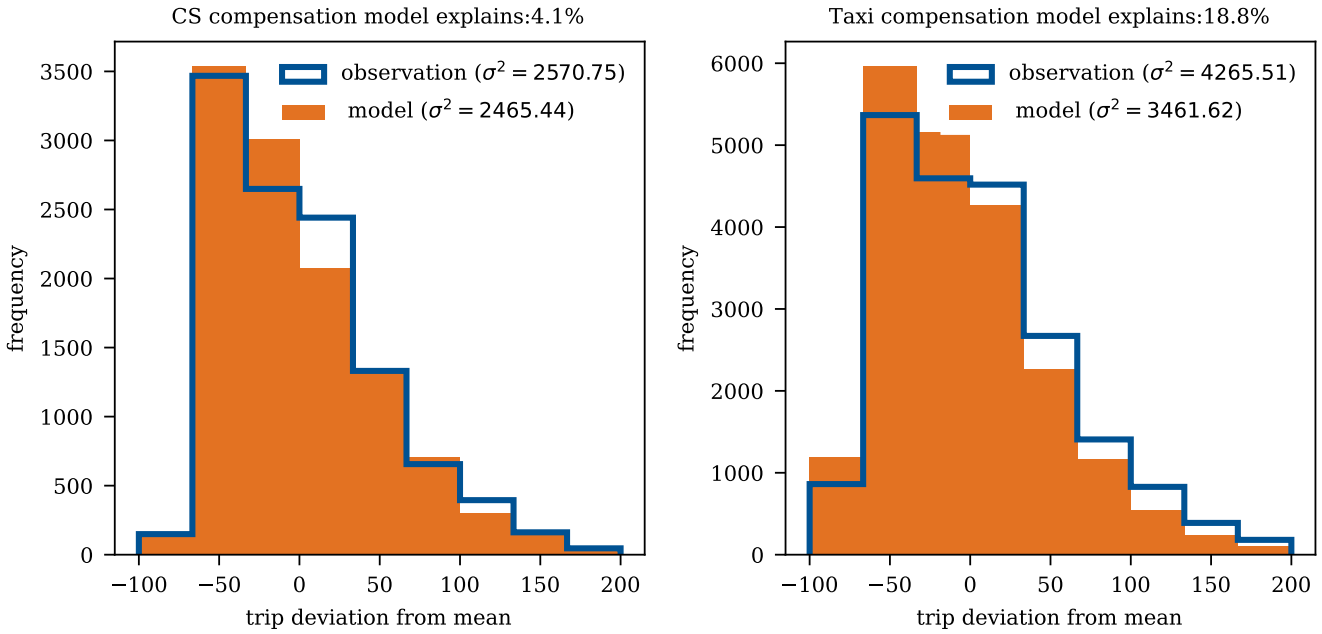

Figure 10. Explained Demand Variance due to Delays.

We individually compute the variance in each data set as

$$\sigma^2 = \sum_{it} (\mathrm{dev}_{it} - \mu_{\mathrm{dev}})^2$$
$$\hat{\sigma}^2 = \sum_{it} \left(\hat{\mathrm{dev}}_{it} - \mu_{\hat{\mathrm{dev}}}\right)^2,$$

where μ_{dev} and $\mu_{\hat{\mathrm{dev}}}$ refer to the average over all dev_{it} and $\hat{\mathrm{dev}}_{it}$, respectively. The relative decrease in variance from the basic to corrected data set (S to $\hat{S}$) corresponds to the fraction of demand variation that can be explained by delays in the public transit network.

$$1 - \frac{\hat{\sigma}^2}{\sigma^2}$$

Thus, we can explain 4.1% of the variance by delays for carsharing and 18.8% for taxi. Consequentially, demand prediction accuracy can be improved by considering the delays during periods of high delay. At first, explaining 4.1% of carsharing demand may not seem much, but, when considering the abundant number of influencing factors (weather, events, ...), uncertainties of human behavior, the fact that delays do not vary too much, and spatial differences in the explanatory power, explaining 4.1% is already significantly improving the situation of MoD operators in a business with low profit margins. The explanatory power of delays on taxi demand variation is significantly higher. Technically, this is a consequence of larger maximum correction factors returned by the function $f(\delta_{it})$, as readily visible from Figure 9. It appears as if users rather switch to taxi than to carsharing if delays occur. This suggests that carsharing customers, on average, have a lower valuation of time than taxi customers, which is backed by general intuition. It does not necessarily mean that the user groups of taxi and public transit are overlapping more than the user groups of carsharing and public transit, but might rather point towards a lower willingness to wait among taxi users.

4.3. Changing Demand Patterns in Presence of Known Outages

While one might assume that MoD demand increases during an outage (and such a correlation has been reported by [23]), we cannot support this assumption based on our data. In Munich, the total demand does not significantly change on days with outages.

This might be due to the fact that outages are commonly known before, and travelers, therefore, either forgo trips, or use private vehicles or bikes. This indicates that, while taxi and carsharing are an alternative to public transit if the delay was unexpected, customers tend to use a different mode of transportation or omit trips if the delay was known before.

Instead, demand patterns (given by trip origins) change. For every day in the observation period, we compare the relative taxi demand per origin-destination pair to the previous year by means of a Wasserstein distance. The Wasserstein distance (also known as the Earth mover's distance) refers to the "work necessary" to transform one distribution into another. We compare the difference between two vectors v_t and v'_t, where v_t refers to the relative demand distribution in the previous year for some timeframe t

$$v_{it} = \frac{d_{it}}{\sum\limits_{j} d_{jt}},$$

and analogous for v' (because outages occur for a longer period of time, and since random demand imbalances can occur within the day with Integer demands, we aggregate timeframes t to a daily level). To increase comparability, we compare any day in the current year to the closest day in the previous year, which is the same weekday (e.g., Monday 4 November 2019 to Monday 5 November 2018), and omit days that were a public holiday in either year. The Wasserstein distance per timeframe t is then calculated by solving the following linear program of a transportation problem

$$\begin{aligned} &\min \sum_{i,j} \Delta_{ij} x_{ij} \\ v_{it} + \sum_{j} x_{ij} &= v'_{it} \\ x_{ij} &\geq 0 \qquad\qquad \forall i, j \end{aligned}$$

where the decision variables x_{ij} refer to the amount of demand that is "shifted", and Δ_{ij} refers to the Euclidean distance between locations i and j (representing the "difference" between demand patterns, i.e., the transportation cost).

There are 21 days on which the main line of the S-Bahn was closed in one direction at one station, and 10 instances where the main line was closed in both directions. While taxi data are available for all days in the observation period, carsharing data are only available on 20 and six of these days, respectively.

We test whether the Wasserstein distances increase (or decrease) during outages when comparing an instance with an outage 2019 and no outage 2018 using a two-tailed Welch's t-test. Table 2 lists the results for carsharing and taxi. Carsharing demand patterns significantly ($\alpha = 5\%$) differ in presence of delays, given by a significantly increasing Wasserstein distance, for one-directional and two-directional closures, both independently and jointly. For taxi demand patterns, the null hypothesis (mean Wasserstein distances do not differ, travel patterns are similar) cannot be rejected for uni-directional closures (In this case, the mean Wasserstein distance even decreased insignificantly.) and all closures. Wasserstein distances for taxi demand patterns significantly increase (at a significance level of 10%) when the main track was closed in both directions. Thus, we conclude that demand patterns change in the presence of outages, and MoD operators should include this information in demand prediction at the local level.

Table 2. Results of the Welch's t-Test for Changing Demand Patterns during Outages.

	No Outage	**Outage**	**No Out.**	**1-Way**	**No Out.**	**2-Way**
Carsharing						
mean	25,498	30,878	25,498	29,496	25,498	35,486
variance	4.92E7	3.95E7	4.92E7	3.60E7	4.92E7	2.78E7
# observations	131	26	131	20	131	6
t-statistic	-3.91		-2.71		-4.47	
$p(T \geq t)$	3.71E-4		1.13E-2		4.25E-3	
crit. value	1.69		2.45		2.05	
Taxi						
mean	16,751	17,338	16,751	16,234	16,751	19,656
variance	2.83E7	2.42E7	2.83E7	2.20E7	2.83E7	2.30E7
# observations	304	31	304	21	304	10
t-statistic	-0.82		0.32		-1.99	
$p(T \geq t)$	0.42		0.75		7.41E-2	
crit. value	2.02		2.06		2.23	

4.4. Decrease of Demand along Lines during COVID-19

During the recent COVID-19 pandemic, demand decreased significantly (average number of taxi trips from 9 March to 3 May 2020 decreased by a factor of 3.5 as compared to the previous year; this decrease is significant at $\alpha = 0.05$ using a Welch's t-test). Reduced mobility is one of the key levers to reduce the spread of a pandemic, as [37] show for Italy. Surprisingly, the relative demand along public transportation lines also decreased (15.9% of all trips vs. 20.8%). This result is significant according to a Welch's t-test at all commonly used significance levels (see the results in Table 3).

Table 3. Results of the Welch's t-Test for Fraction of Trips Along Lines.

	Before COVID	**During COVID**
mean	0.159	0.208
variance	4.12E-4	2.25E-4
# observations	56	708
t-statistic	-17.7	
$p(T \geq t)$	1.54E-25	
crit. value	2.00	

When combined with the decreasing delays during this period (see Table 4), this provides anecdotal evidence that the increased punctuality made it unnecessary to choose alternative modes of transportation.

Table 4. Changing Public Transit Delay before and during COVID.

	Before COVID		**During COVID**	
	S-Bahn	**U-Bahn**	**S-Bahn**	**U-Bahn**
mean delay in min	1.168	0.837	0.979	0.612
% $1 >$ delay	46.7	50.6	47.3	59.9
% $3 >$ delay ≥ 1	35.1	37.7	37.5	33.5
% $6 >$ delay ≥ 3	13.5	9.9	12.0	6.1
% delay ≥ 6	4.7	1.8	3.2	0.5

4.5. Potential for Fleet Size Reduction

Public authorities can reduce congestion in the road network as well as the necessary fleet size of MoD operators by improving the punctuality of the public transit system, since a higher delay of the public transit service entails higher demand for MoD.

If all the delays were 0, the average demand for taxi would decrease by 2.2%, and the average demand for carsharing would decrease by 0.5%, as given by the shift to the left-most bin in in Figure 9. If all delays were reduced by 50%, the average demand for taxi would decrease by 1.6%, and the average demand for carsharing would decrease by 0.5%. These values are computed by artificially reducing the average delay for each time interval at each location, and correcting the observed demand according to the corresponding carsharing/taxi trip deviation.

These demand reductions can serve as an upper bound for potential fleet size reductions. The actual fleet size reduction may be less due to risk pooling in the presence of stochastic demand. It stands to reason that the number of trips in privately owned cars also reduces, even though the exact values might differ. This indicates that road-congestion might be alleviated by reducing delays of the public transit operator, and making it more reliable.

5. Discussion and Conclusions

In this paper, we study the effect of public transit delays on MoD operators, i.e., carsharing and taxi. If customers judge trips based on some combination of travel time and travel cost, demand for the "outside option" (e.g., MoD) will increase if the travel time for public transit increases. We conduct large-scale experiments using carsharing and taxi trip data and public transit departure data for 10 months in Munich, Germany.

Demand for MoD increases if public transit delays increase. The mean demand for carsharing varies by up to 13.5% and the mean taxi demand varies by up to 52.1%, depending on the extent of the delay.

4.1% of carsharing and 18.8% of taxi demand variance can be explained if public transit is delayed. Thus, it seems as if carsharing customers valued travel time less than taxi customers. Because public transit delay is only one of many influencing factors (besides weather, events, and others), the explanatory power is high.

If the public transit operator were delayed less, the necessary carsharing and taxi fleet could be reduced by up to 0.5% and 2.2%, respectively. Even if delays did not vanish entirely, the number of taxi and carsharing trips could decrease substantially. Thus, public authorities might improve the public transit with the goal of reducing the congestion in their road network. Improvements of the public transit operator to alleviate congestion exceeds all approaches (e.g., [38]) for regulations of MoD discussed in existing literature, and poses an interesting line for future research.

Customers adapt their travel patterns if the public transit service is not operating. Unlike existing literature, we do not find that demand increases during outages. Most likely, a substantial number of travelers decide to delay their trip until the end of the outage, as outages are known upfront. Among the remaining travelers, origin-destination pairs change significantly, given by an increasing Wasserstein distance when comparing the origin–destination distribution to the previous year.

A few comments are in order: we only measure correlation, but no cause–effect relationship. It could also be possible that an increase in carsharing and taxi usage increases the delay for public transit. However, from an application point of view, this is unrealistic for rail traffic, and even the maximum number of carsharing vehicles and taxis should not incur significant delays for road-based public transit (bus, rail replacement services). Additionally, both public transit delays and carsharing/taxi demand might be dependent on an external influence that we did not correct for. While we cannot prove that no external source caused the correlation, a causal relation is the most likely explanation. Further, we must mention that the measured effect is minimal. This is because the S-Bahn has very similar and rather low delays on most instances. It is possible that some passengers choose

MoD, rather than public transit already due to the current mean delay. Our method cannot capture this and, therefore, only returns a lower bound on the influence of public transit delays on MoD demand. Because the data set only permits integer delays and since delays are subject to external influences, the effects remain minimal. More precise data would permit a more extensive analysis. However, this approach is important, since it allows third parties, such as policy makers or new market entrants, in order to measure the effect with data they have available, or can easily collect. The evidence that carsharing and taxi can help in increasing accessibility is rather anecdotal. Insights can be strengthened in future research if data are available prior and posterior to opening new lines in the public transit system. In future work, our results can be used to approximate a customer choice function in a data-driven fashion.

Author Contributions: Conceptualization, L.M. and M.W.; methodology, L.M. and M.W.; software, L.M., M.W. and X.L.; validation, L.M., M.W. and X.L.; investigation, L.M., M.W. and X.L.; resources, M.W.; data curation, L.M., M.W. and X.L.; writing—original draft preparation, L.M. and M.W.; writing—review and editing, L.M. and M.W.; visualization, M.W. and X.L.; supervision, L.M.; project administration, L.M. and M.W. All authors have read and agreed to the published version of the manuscript.

Funding: The work of Layla Martin and Xinyu Li was supported by Deutsche Forschungsgemeinschaft as part of the Research Training Group 2201 (Advanced Optimization in a Networked Economy). The work of Xinyu Li was partially supported by the German Academic Exchange Service. The work of Michael Wittmann was independently funded by the Chair of Automotive Technology.

Acknowledgments: The authors would like to thank IsarFunk Taxizentrale GmbH & Co. KG for providing real world taxi data.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. INRIX. *INRIX Verkehrsstudie: Stau Verursacht Kosten in Milliardenhöhe*; INRIX: Kirkland, WA, USA, 2020. Available online: https://inrix.com/press-releases/2019-traffic-scorecard-german/ (accessed on 5 November 2020).
2. Banister, D. Cities, mobility and climate change. *J. Transp. Geogr.* **2011**, *19*, 1538–1546. [CrossRef]
3. Strong Towns. We Should Be Building Cities for People, Not Cars. 2018. Available online: https://www.strongtowns.org/journal/2018/7/2/we-should-be-building-cities-for-people-not-cars (accessed on 28 February 2020).
4. Liu, Y.; Bansal, P.; Daziano, R.; Samaranayake, S. A framework to integrate mode choice in the design of mobility-on-demand systems. *Transp. Res. Part C Emerg. Technol.* **2019**, *105*, 648–665. [CrossRef]
5. Hess, D.; Brown, J.; Shoup, D. Waiting for the Bus. *J. Public Transp.* **2004**, *7*, 67–84. [CrossRef]
6. Le Vine, S.; Lee-Gosselin, M.; Sivakumar, A.; Polak, J. A new approach to predict the market and impacts of round-trip and point-to-point carsharing systems: Case study of London. *Transp. Res. Part D Transp. Environ.* **2014**, *32*, 218–229. [CrossRef]
7. Nobis, C.; Kuhnimhof, T. *Mobilität in Deutschland—MiD Ergebnisbericht*; Technical Report; Infas, DLR, IVT and Infas 360: Bonn, Berlin, 2018.
8. Van Exel, N.J.A.; Rietveld, P. Perceptions of public transport travel time and their effect on choice-sets among car drivers. *J. Transp. Land Use* **2010**, *2*, 75–86. [CrossRef]
9. Rayle, L.; Dai, D.; Chan, N.; Cervero, R.; Shaheen, S. Just a better taxi? A survey-based comparison of taxis, transit, and ridesourcing services in San Francisco. *Transp. Policy* **2016**, *45*, 168–178. [CrossRef]
10. Albiński, S.; Fontaine, P.; Minner, S. Performance analysis of a hybrid bike sharing system: A service-level-based approach under censored demand observations. *Transp. Res. Part E Logist. Transp. Rev.* **2018**, *116*, 59–69. [CrossRef]
11. Basciftci, B.; Ahmed, S.; Shen, S. Distributionally robust facility location problem under decision-dependent stochastic demand. *Eur. J. Oper. Res.* **2020**, doi:10.1016/j.ejor.2020.11.002. [CrossRef]
12. Salazar, M.; Lanzetti, N.; Rossi, F.; Schiffer, M.; Pavone, M. Intermodal autonomous mobility-on-demand. *IEEE Trans. Intell. Transp. Syst.* **2020**, *21*, 3946–3960. [CrossRef]
13. Guidon, S.; Reck, D.J.; Axhausen, K. Expanding a(n) (electric) bicycle-sharing system to a new city: Prediction of demand with spatial regression and random forests. *J. Transp. Geogr.* **2020**, *84*, 102692. [CrossRef]
14. Vosooghi, R.; Puchinger, J.; Jankovic, M.; Sirin, G. A critical analysis of travel demand estimation for new one-way carsharing systems. In Proceedings of the 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), Yokohama, Japan, 16–19 October 2017; pp. 199–205.
15. Brodeur, A.; Nield, K. An empirical analysis of taxi, Lyft and Uber rides: Evidence from weather shocks in NYC. *J. Econ. Behav. Organ.* **2018**, *152*, 1–16. [CrossRef]

16. Markou, I.; Kaiser, K.; Pereira, F.C. Predicting taxi demand hotspots using automated Internet Search Queries. *Transp. Res. Part C Emerg. Technol.* **2019**, *102*, 73–86. [CrossRef]
17. Markou, I.; Rodrigues, F.; Pereira, F.C. Real-Time Taxi Demand Prediction using data from the web. In Proceedings of the IEEE 2018 21st International Conference on Intelligent Transportation Systems (ITSC), Maui, HI, USA, 4–7 November 2018; pp. 1664–1671.
18. Schmöller, S.; Weikl, S.; Müller, J.; Bogenberger, K. Empirical analysis of free-floating carsharing usage: The Munich and Berlin case. *Transp. Res. Part C Emerg. Technol.* **2015**, *56*, 34–51. [CrossRef]
19. Yang, C.; Gonzales, E.J. Modeling taxi trip demand by time of day in New York City. *Transp. Res. Rec.* **2014**, *2429*, 110–120. [CrossRef]
20. Alemi, F.; Circella, G.; Mokhtarian, P.; Handy, S. What drives the use of ridehailing in California? Ordered probit models of the usage frequency of Uber and Lyft. *Transp. Res. Part C Emerg. Technol.* **2019**, *102*, 233–248. [CrossRef]
21. Correa, D.; Xie, K.; Ozbay, K. Exploring the taxi and Uber demand in New York City: An empirical analysis and spatial modeling. In Proceedings of the 96th Annual Meeting of the Transportation Research Board, Washington, DC, USA, 8–12 January 2017.
22. Kopp, J.; Gerike, R.; Axhausen, K.W. Do sharing people behave differently? An empirical evaluation of the distinctive mobility patterns of free-floating car-sharing members. *Transportation* **2015**, *42*, 449–469. [CrossRef]
23. Tyndall, J. Free-floating carsharing and extemporaneous public transit substitution. *Res. Transp. Econ.* **2019**, *74*, 21–27. [CrossRef]
24. Ampudia-Renuncio, M.; Guirao, B.; Molina-Sánchez, R.; de Álvarez, C.E. Understanding the spatial distribution of free-floating carsharing in cities: Analysis of the new Madrid experience through a web-based platform. *Cities* **2020**, *98*, 102593. [CrossRef]
25. Sprei, F.; Habibi, S.; Englund, C.; Pettersson, S.; Voronov, A.; Wedlin, J. Free-floating car-sharing electrification and mode displacement: Travel time and usage patterns from 12 cities in Europe and the United States. *Transp. Res. Part D Transp. Environ.* **2019**, *71*, 127–140. [CrossRef]
26. Hall, R.W. Passenger waiting time and information acquisition using automatic vehicle location for verification. *Transp. Plan. Technol.* **2001**, *24*, 249–269. [CrossRef]
27. Vande Walle, S.; Steenberghen, T. Space and time related determinants of public transport use in trip chains. *Transp. Res. Part A Policy Pract.* **2006**, *40*, 151–162. [CrossRef]
28. He, Y.; Zhao, Y.; Tsui, K.L. Modeling and Analyzing Impact Factors of Metro Station Ridership: An Approach Based on a General Estimating Equation. *IEEE Intell. Transp. Syst. Mag.* **2020**, *12*, 195–207. [CrossRef]
29. Weikl, S.; Bogenberger, K. A practice-ready relocation model for free-floating carsharing systems with electric vehicles—Mesoscopic approach and field trial results. *Transp. Res. Part C Emerg. Technol.* **2015**, *57*, 206–223. [CrossRef]
30. Jäger, B.; Wittmann, M.; Lienkamp, M. Analyzing and modeling a City's spatiotemporal taxi supply and demand: A case study for Munich. *J. Traffic Logist. Eng.* **2016**, *4*, doi:10.18178/jtle.4.2.147-153. [CrossRef]
31. Uber Engineering. H3—A Hexagonal Hierarchical Geospatial Indexing System. 2020. Available online: https://h3geo.org/ (accessed on 28 February 2020).
32. Jorge, D.; Correia, G. Carsharing systems demand estimation and defined operations: A literature review. *Eur. J. Transp. Infrastruct. Res.* **2013**, *13*, doi:10.18757/ejtir.2013.13.3.2999. [CrossRef]
33. Afian, A.; Odoni, A.; Rus, D. Inferring unmet demand from taxi probe data. In Proceedings of the 2015 IEEE 18th International Conference on Intelligent Transportation Systems, Las Palmas de Gran Canaria, Spain, 15–18 September 2015; pp. 861–868.
34. Negahban, A. Simulation-based estimation of the real demand in bike-sharing systems in the presence of censoring. *Eur. J. Oper. Res.* **2019**, *277*, 317–332. [CrossRef]
35. Deutscher Taxi- und Mietwagenverband e.V. *BZP Geschäftsbericht 2017/2018*; Technical Report; Deutscher Taxi- und Mietwagenverband e.V.: Frankfurt am Main, Germany, 2020.
36. Wittmann, M.; Neuner, L.; Lienkamp, M. A Predictive Fleet Management Strategy for On-Demand Mobility Services: A Case Study in Munich. *Electronics* **2020**, *9*, 1021. [CrossRef]
37. La Gatta, V.; Moscato, V.; Postiglione, M.; Sperli, G. An Epidemiological Neural network exploiting Dynamic Graph Structured Data applied to the COVID-19 outbreak. *IEEE Trans. Big Data* **2020**, doi:10.1109/TBDATA.2020.3032755. [CrossRef]
38. Yan, X.; Levine, J.; Zhao, X. Integrating ridesourcing services with public transit: An evaluation of traveler responses combining revealed and stated preference data. *Transp. Res. Part C Emerg. Technol.* **2019**, *105*, 683–696. [CrossRef]

electronics

MDPI

Article

Hierarchical Temporal Memory Theory Approach to Stock Market Time Series Forecasting

Regina Sousa, Tiago Lima, António Abelha and José Machado *

ALGORITMI Research Center, School of Engineering, Gualtar Campus, University of Minho, 4710-057 Braga, Portugal; regina.sousa@algoritmi.uminho.pt (R.S.); a77788@alunos.uminho.pt (T.L.); abelha@di.uminho.pt (A.A.)

* Correspondence: jmac@di.uminho.pt

Abstract: Over the years, and with the emergence of various technological innovations, the relevance of automatic learning methods has increased exponentially, and they now play a key role in society. More specifically, Deep Learning (DL), with the ability to recognize audio, image, and time series predictions, has helped to solve various types of problems. This paper aims to introduce a new theory, Hierarchical Temporal Memory (HTM), that applies to stock market prediction. HTM is based on the biological functions of the brain as well as its learning mechanism. The results are of significant relevance and show a low percentage of errors in the predictions made over time. It can be noted that the learning curve of the algorithm is fast, identifying trends in the stock market for all seven data universes using the same network. Although the algorithm suffered at the time a pandemic was declared, it was able to adapt and return to good predictions. HTM proved to be a good continuous learning method for predicting time series datasets.

Keywords: time series forecasting; HTM; regression; machine intelligence; deep learning

Citation: Sousa, R.; Lima, T.; Abelha, A.; Machado, J. Hierarchical Temporal Memory Theory Approach to Stock Market Time Series Forecasting. *Electronics* **2021**, *10*, 1630. https://doi.org/10.3390/electronics10141630

Academic Editors: Juan M. Corchado, Josep L. Larriba-Pey, Pablo Chamoso and Fernando De la Prieta

Received: 24 March 2021
Accepted: 6 July 2021
Published: 8 July 2021

1. Introduction

1.1. Contextualization

HTM can be described as the theory that attempts to describe the functioning of the neocortex, as well as the methodology that intends to provide machines with the capacity to learn in a human way [1].

The neocortex is defined as the portion of the human cerebral cortex from which comes the highest cognitive functioning, occupying approximately half the volume of the human brain. The neocortex is understood by four main lobes with specific functions of attention, though, perception, and memory. These four regions of the cortex are the frontal, parietal, occipital, and temporal lobes. The frontal lobe's responsibilities are the selection and coordination of behavior. The parietal lobe is qualified to make decisions in numerical cognition as well as in the processing of sensory information. The occipital lobe, in turn, has a visual function. Finally, the temporal lobe has the functions of sensory as well as emotional processing and dealing with all significant memory. Thus, the algorithm that is presented intends to create a transposition of this portion of the brain, creating a machine with "true intelligence" [2].

The HTM is built based on three of the main characteristics of the neocortex. Thus, it is a system of memory, with temporal patterns and the construction of regions according to a hierarchical structure.

Starting with the first region, the encoder deals with all of the sensory component. This will receive the data in their raw form, converting them into a set of bits, that will later be transformed into a Sparse Distributed Representation (SDR). Transposing into the human organism, the SDRs correspond to the active neurons of the neocortex. Thus, a 1 bit represents an active neuron while a 0 bit represents an inactive neuron. This transformation is achieved by transforming the data into a set of bits while maintaining the semantic

characteristics essential to the learning process. One of the characteristics that proved to be quite interesting is that similar data entries, when submitted to the encoding process, create overlapping SDRs; that is, with the active bits placed in the same positions. Another important characteristic is that all SDRs must have a similar dimensionality and sparsity (the ratio between the number of bits at 1 and the total number of bits) [3]. A certain percentage of sparsity will result in a system's ability to handle noise and under sampling.

The second region, Spatial Pooler (SP), is responsible for assigning the columns according to a fixed number, where each column corresponds to a dendritic segment of the neuron that connects to the input space created by the region described above, the encoder. Each segment has a set of synapses, that can be initialized at random, with a permanence value. Some of these synapses will be active (when connected to a bit with value 1) and consequently will be driven in such a way as to inhibit other columns in the vicinity. Therefore, the SP is responsible for creating an SDR of active columns. This transformation follows the Hebbian learning rule that for each input, the active synapses are driven by inhibiting the inactive synapses. The thresholds dictate whether a synapse is active or not.

The third region, Temporal Memory (TM), starts from the result of the previous two, finding patterns in the sequence of SDRs in order to determine a prediction for the next SDR. At the beginning of the process, all the cells of the active column are also active; however, the region TM is responsible for activating a subset of cells of those same columns when a context is predicted. In case there is no forecast, all the cells remain active. The activation of the previously mentioned subsets of cells is carried out because only in this way can the same entry be represented according to different contexts.

Finally, the classifier is the region in which a decoder calculates the overlap of the predicted cells of the SDR obtained, selecting the one with more overlaps and comparing it with the actual value (if known) [4,5].

Figure 1 describes the typical process of an HTM network.

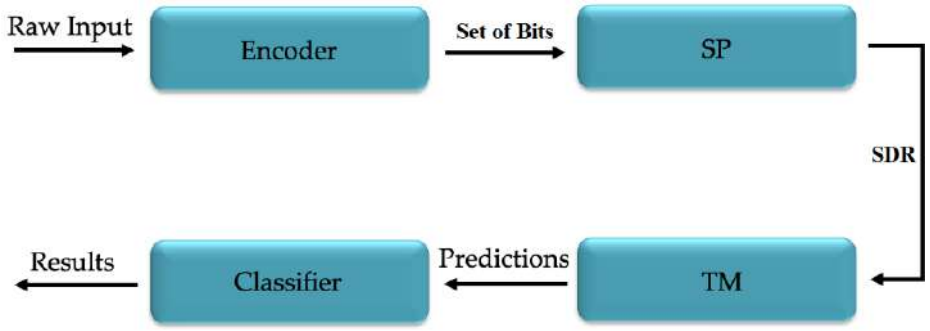

Figure 1. HTM network topology.

1.2. Motivation

HTM is built in three main features of the neocortex: it is a memory system with temporal patterns and its regions are organized in a hierarchical structure. There are many biological details that the theory ignores in case they have no relevance for learning. In short, this approach includes Sparse Distributed Representation (SDR)s, its semantical and mathematical operations, and neurons along the neocortex capable of learning sequences and enabling predictions; these systems learn in a continuous way, with new inputs through time and with flows of information top-down and bottom-up between its hierarchical layers, making them efficient in detecting temporal anomalies. The theory relies on the fact that by mimicking the neocortex, through the encoding of data in a way that gives it a semantic meaning, activating neurons sparsely in an SDR through time will give these systems a power to generalize and learn, not achieved to date with other classic approaches of AI. It is expected to achieve better results and conclusions, while being an intelligence with a higher flexibility when put up against adverse contexts.

1.3. Objectives

The idea of this paper was born from the scope previously mentioned, with the objective to study applications of the HTM theory that are still largely unknown to the pattern learning and recognition community; the applications being studied range from audio recognition, image classification, and time series forecasting with public datasets, that may someday help in anomaly detections in medicine, hospital management, or to act in case of urgency matters. In order to have the confidence to use these systems daily, there is the need for the introduction of new technologies, supported by an AI system with a higher generalization capacity to the ones already in place. With this in mind, the objectives are the following:

- Test and analyze the applications of the HTM theory;
- Compare the HTM theory results against traditional ML technics in terms of:
  - Accuracy and other classification or regression metrics;
  - Computing power/time required;
  - Amount and type of data required;
  - Noise robustness of the algorithms;
  - Possibility to justify the obtained results.

2. State of the Art

Predicting stock market performance is a very challenging task. Even people with an excellent understanding of statistics and probability have difficulty in doing so. Numerous factors combine to make stock prices so volatile that forecasting is at first sight impossible. Adding to all this complexity are all of the political and social factors. Therefore, this article intends to elaborate on a theory and its algorithm on stock market forecasting, determining the future value of a given company's shares. Nevertheless, several studies aim to accept the challenge, and while some statistical and Machine Learning algorithms achieve significant results, the search for closer to ideal results is underway [1,6,7].

There are numerous application fields where HTM can be applied and can produce excellent results. For example, smart cities and their use of sensors, actuators, and mobile devices produce huge streams of data daily, that should be exploited towards innovative solutions and applications [8]. These streams of data are essential for an HTM network that is continuously learning; thus, a problem such as stock market prediction is a good indicator of if HTM can be used in such a scenario, such as in smart cities.

The paper "Forecasting S&P 500 Stock Index Using Statistical Learning Models" [9] defines the primary objective as the forecast of the S&P 500 index movement, using statistical learning models such as logistic regression and naïve Bayes. In this work, an accuracy of 62.51% was obtained. Regarding the dataset, the data were collected between 2004 and 2014, and a transformation of daily prices into daily returns was performed. Similarly, the model described in [10] collects the stock price every 5 min by calculating its return using data for the years 2010 to 2014 from the South Korean stock market. However, in this study, a three-level Deep Neural Network (DNN) model was chosen, using four different representation methods: raw data, Principal Component Analysis (PCA), autoencoder, and restricted Boltzmann machine.

In 2018, ref. [11] proposed a two-stream gated Gated Recurrent Unit (GRU) model and a sentiment word embedding trained on a financial news dataset in order to predict the directions of stock prices by using not only daily S&P 500 stock prices but also a financial news dataset and sentiment dictionary, obtaining an accuracy of 66.32%. More recently, as presented in the article [12], a long short-term memory (LSTM) network was used to predict the future trend of stock prices based on the price history of the Brazilian stock market. However, the accuracy was only 55.9%.

In the same year, in [13], a LSTM network was also used, using an S&P 500 data set for the period from 17 December 2010 to 17 January 2013. In the published document, the objective was well clarified, and it was intended to predict the value of the following

day, based on the last 30 days; the mean absolute percentage error (MAPE) obtained was 0.0410%.

In [14], three different models were proposed to forecast stock prices using data from January 2009 to October 2019: autoregressive integrated moving average (ARIMA), simple moving average (SMA), and Holt–Winters method. The SMA model had the best forecasting performance, with a MAPE of 11.456808% in the test data (January to October 2019).

Another DL approach, by [15], made use of Wavelet Transform (WT), Stacked AutoEncoder (SAE) and LSTM in order to create a network for the stock price forecasting of six different markets at different development stages (although it was not clear which companies' data were used); similarly to [16], 12 technical indicators were taken from the data. The WT component had the objective of eliminating noise, the SAE of generating "deep high-level features", and the LSTM would take these features and forecast the next day closing price. With 5000 epochs and the dataset divided into 80% for training, 10% for validation, and 10% for testing, the average MAPE obtained in six years was of 0.011% for the S&P 500 index.

With the increase in the availability of streaming time series data came the opportunity to model each stream in an unsupervised way in order to detect anomalous behaviors in real-time. Early anomaly detection requires that the system must process data in real-time, favoring algorithms that learn continuously. The applications of HTM have been focused on the matter of anomaly detection. In [17], a comparison between an HTM algorithm against others such as Relative Entropy, K-Nearest Neighbor (KNN), Contextual Anomaly Detector (CAD), CAD Open Source Edition (OSE), Skyline, in the anomaly detection of various datasets of the Numenta Anomaly Benchmark107(NAB) was made. HTM demonstrated that it is capable of detecting spatial and temporal anomalies, both in predictable and noisy domains.

In addition, in [18], an HTM network was compared against ARIMA, Skyline, and a network based on the AnomalyDetection R package developed by Twitter, using real and synthetic data sets. Not only were good precision results obtained using the HTM, but there was also a significant reduction in processing time. In [19], it is claimed that most anomaly detection technics perform poorly with unsupervised data; with this in mind, 25 datasets from the NYSE stock exchange, with historical data of 23 years, were analyzed by an HTM network in order to detect anomaly points. However, no explanation of the parameters used was made and no ground truth is known, making it hard to make conclusions. A synthetic dataset was also used, with known anomaly points—the network failed to detect when the values were too low, only detecting when the data were multiplied by 100—possibly by a faulty encoding process.

Leaving the anomaly detection domain, in 2016, [20] used a HTM model to predict the New York City taxi passenger count 2.5 h in advance, with aggregated data at 30-min intervals, obtaining a MAPE of 7.8%, after observing 10,000 data records, lower than other LSTM models used in the study. By including this reference, it is intended to demonstrate that HTM can be used in various contexts and with quite significant results in most cases. In 2020, ref. [21] used recurrent neural networks, such as LSTM and GRU, to solve the same problem of taxi passenger counting. On this approach, through hyper-parametric tuning and careful data formatting, it is stated that both the GRU model and the LSTM model exceeded the HTM model by 30% in lower runtime.

Kang et al. [22], compared the efficiency in memory and time consumption of an HTM network with a modified version of the network for a continuous multi-interval prediction (CMIP) in order to predict stock price trends based on various intervals of historical data without interruption; the conclusions were that the modified version was more efficient in memory and time consumption for this problem, although no conclusions were taken in terms of accuracy of the predictions.

In 2013, Gabrielsson et al. [16], used a genetic algorithm in order to optimize the parameters of two networks: HTM and Artificial Neural Network (ANN); with two months

of the S&P 500 index data (open, close, high, low, and volume) aggregated by the minute, 12 technical indicators were extracted and fed to the networks. The problem was converted into a classification one, with training, validation, and test datasets, where the classifier was binary—price will or will not rise—following a buy-and-hold trading mechanism. The Profit and Loss (PnL) was used as a performance measure, where the HTM model achieved more than three times the profit obtained by the ANN network.

The arrival of the Covid-19 pandemic brought uncertainty to the financial markets around the globe. According to [23], an increase of 1% in cumulative daily Covid-19 cases in the US results in approximately 0.01% of an accumulative reduction in the S&P 500 index after one day and 0.03% after one month. In [24], a variety of economic uncertainty measures were examined, showing this same uncertainty; also, it was observed that there is a lack of historical parallelism of this phenomenon, due to the suddenness and enormity of the massive job losses. Both studies suggest that the peak of the negative effects in the stock market was observed during March 2020.

3. Why Hierarchical Temporal Memory?

The HTM starts from the assumption that everything the neocortex decides to do is based on both memories as well as the sequence of patterns; this algorithm is based on the theory of a thousand brains. Among many other things, this theory tries to suggest mechanisms to explain how the cortex represents objects as well as their behavior. HTM is the algorithmic implementation of this theory. The great goal is then to understand how the neocortex works and build systems on that same principle. In particular, this method focuses on three main properties:

- Sequence learning;
- Continuous learning;
- Sparse distributed representations.

This method is relatively recent when compared, for example, to neuronal network techniques. Therefore, it is important to highlight the advantages of HTM and why it was chosen. It should be noted that all the statements presented here were based on authors presented in the state of the art.

In short, the reasons why HTM was chosen are:

1. HTM is the most proven model for the construction of intelligence such as brain intelligence;
2. Although it presents some complexity, it is a scalable and comprehensive model for all the tasks of the neocortex;
3. The neuronal networks are based on mathematics while HTM is inspired fundamentally in the biology of the brain;
4. HTM is more noise-tolerant than any other technique presented until today, due to the sparse distribution representations of raw input;
5. It is a fault-tolerant model;
6. It is variable in time, since it is dependent of state as well as of the context it is presented;
7. It is an unsupervised model;
8. Only a small quantity of data are required;
9. No training/testing datasets are required;
10. Few hyper-parameters tuning—most of the parameters from the algorithms are general to the theory and fall into a specific range of values.

However, as in all methods ever presented, there are already trade-offs:

1. The optimization of HTM for GPU can be difficult;
2. HTM is not a mathematically sound solution as the neural network;
3. This theory is recent and therefore still under construction;
4. There are relatively few applications made so far, and although the community is growing, it is not as vast as the neural networks' community.

4. Data and Methods

Since it was not possible to find a representative dataset of the intended case studies, such as ozone values and traffic in cities, among others, the work was applied to time series forecasting of the close values in the stock market, for seven of the S&P 500 index companies: Amazon, Google, HCA Healthcare, Disney, McDonald's, Johnson & Johnson, and Visa.

4.1. Dataset

The selection of a dataset as well as the features to be used may be determinant for the success of the research work. Therefore, these were well thought out, and a script to obtain stock fluctuations for various companies was made, pulling data from Yahoo Finance, ranging from 3 January 2006 until 18 September 2020. Seven datasets were created, each related to an S&P 500 company: Amazon, Google, HCA Healthcare, Disney, McDonald's, Johnson & Johnson, and Visa; the HCA Healthcare dataset only had data from 10 March 2011, and the Visa dataset from 19 March 2008.

To choose from the S&P 500 list of companies, two parameters were considered: first the market capitalization and then the weight index. Companies are typically divided according to market capitalization: large-cap ($10 billion or more), mid-cap ($2 billion to $10 billion), and small-cap ($300 million to $2 billion). Market capitalization refers to the total dollar value of a company's outstanding shares. The market capitalization represents the product between stock price and outstanding shares:

$$Market - Cap = Stock\ Price \times Outstanding\ Shares \quad (1)$$

The S&P 500 uses a market capitalization weighting method, giving a higher percentage allocation to the companies with the highest market capitalization. Therefore, we chose the companies that represented several S&P 500 list levels with the following market capitalization and indexes [24]. The companies chosen are displayed in Table 1.

Table 1. Companies Market Capitalization and Indexes.

Company	Market Capitalization (Billion $)	S&P500 Index
Amazon	1233.4	4.4
Google	1752.64	1.7
Johnson & Johnson	395.3	1.3
Visa	383.9	1.2
Disney	195.3	1.0
McDonalds	139.5	0.5
HCA	164.46	0.14

With this in mind, the seven companies were chosen due to their familiar popularity and because they represent a wide range of business areas—although they did not represent the entire S&P 500 index, these seven datasets were a good sample for the present study, which pretended to investigate how well the HTM theory adjusts to the stock market forecasting, using the same network for different datasets. Another particularity considered was the inclusion of data after the declaration of the Covid-19 pandemic by the World Health Organization (WHO) on 11 March 2020.

The seven datasets had the same fields: date, open, high, low, close, volume and name. Two points were considered: the units of the Open, High, Low, and Close are in USD and the name corresponds to the name of the stock, not of use for forecasting.

Table 2 describes all columns present in the dataset. On the Table 3, it is shown the maximum values of each parameter per company and on the Table 4, the minimum values of the same parameters. A first comparative analysis can be made where it is verified that although all Amazon columns start with significantly lower values than Google, the company's growth was so positive that it ended up surpassing Google with higher values.

Table 2. Description of Dataset columns.

Column	Description
date	Day of the values taken from the stock market
open	Price of the stock at market open
high	Highest price reached in the day
low	Lowest price reached in the day
close	Price of the stock at market close
volume	Number of shares traded
name	The stock's ticker name

Table 3. Maximum values of each parameter.

Company	High	Low	Open	Close	Volume
Amazon	3495	3467	3547	3400	104,329,200
Google	1800	1540	1609	1442	82,151,100
Johnson & Johnson	157	154	153	148	98,440,200
Visa	207	205	212	205	337,533,600
Disney	145	144	148	135	87,048,500
McDonalds	220	211	229	218	86,981,300
HCA	148	147	147	148	81,150,000

Table 4. Minimum values of each parameter.

Company	High	Low	Open	Close	Volume
Amazon	29	26	34	36	881,300
Google	270	235	135	518	520,600
Johnson & Johnson	52	51	53	49	2,323,800
Visa	13	12	13	13	2,188,800
Disney	120	19	20	19	2,165,700
McDonalds	34	35	32	40	963,299
HCA	19	23	20	26	258,800

When plotting the close values for both companies, corresponding to the stock price at the close of the market, it can be observed that there has been a significant increase over the years. By looking at the Figure 2 it can be concluded that, although Amazon presented lower close values at the beginning of 2006, it recovered the difference, obtaining higher values than Google at the end of 2017. The datasets present different patterns and growths, hence the importance of using different companies for this study.

4.1.1. Hierarchical Temporal Memory Network

All data present in the dataset were uploaded to a HTM network which was developed using a python library called Numenta Platform for Intelligence Computing (NUPIC). NUPIC is a machine intelligence platform that allows the implementation of machine intelligence algorithms.

No pre-processing was carried out to the data because they were already very concise and consistent, without any missing or out of range values; also, the network should be able to interpret anomalies on the data and be resistant to noise.

The various regions of the network present the parameters in Tables 5–8.

The parameters presented in the previous tables were one of the most important processes of choice throughout the investigation. While, for example, inputWidth is a value required to guarantee the encoding of data, columnCount, numActiveColumns, boost, and others were carefully tested in order to choose the best one. Therefore, specifically for data encoding, importance was given to the days of the week and the season. The remaining values are numeric and adapted to the value scales.

As for the SP, the default values were maintained for the following parameters: globalInhibition, localAreaDensity, potentialPct, synPermConnected, synPermActiveInc, and

synPermInactiveDec. The remaining parameters: numActiveColumnsPerInhArea, columnCount, and boostStrength were tested and adapted in order to obtain the least possible error.

For the TM region, the parameters tested and adapted according to the results were: cellsPerColumn, maxSynapsesPerSegment, and maxSynapsesPerCell. The remaining parameters were left at the default values: newSynapseCount, initialPerm, permanenceInc, permanenceDec, maxAge, globalDecay, minThreshold, activationThreshold, outputType, and pamLength.

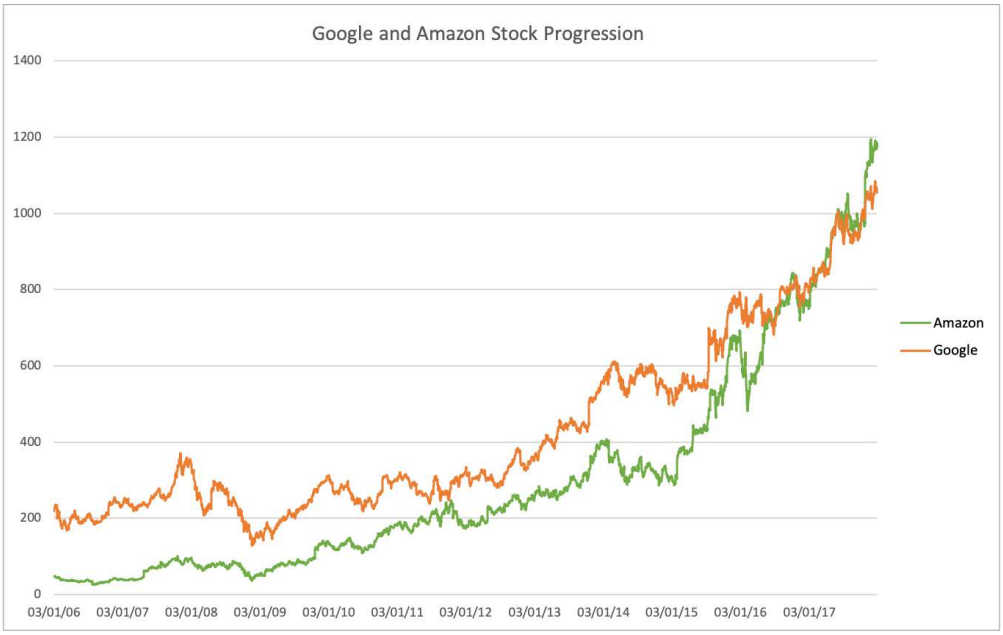

Figure 2. Google and Amazon close value progression in the dataset.

Table 5. Maximum values of each parameter.

Input	Type of Encoder	Parameters
date	DataEncoder	season = dayOfWeek = 3
open	RandomDistributedScalarEncoder	Resolution = 0.5
high	RandomDistributedScalarEncoder	Resolution = 0.5
low	RandomDistributedScalarEncoder	Resolution = 0.5
close	RandomDistributedScalarEncoder	Resolution = 0.5
volume	RandomDistributedScalarEncoder	Resolution = 200
date	DataEncoder	season = dayOfWeek = 3

Table 6. SP Region Parameters.

Parameter	Value
inputWidth	2033
columnCount	4096
GlobalInhibition	1
localAreaDensity	−1
numActiveColumnsPerInhArea	160
potencialPCT	0.85
synPermConnected	0.1
synPermInactiveDec	0.04
synPermInactiveDec	0.005
boostStrength	3

Table 7. TM Region Parameters.

Parameter	Value
inputWidth	2033
columnCount	4096
cellsPerColumn	64
newSynapseCount	20
initialPerm	0.21
permanenceInc	0.1
permanenceDec	0.1
maxAge	0
globalDecay	0
maxSynapsesPerSegment	64
maxSegmentsPerCell	256
minThreshold	12
activationThreshold	16
outputType	Normal
pamLength	1

Table 8. Classifier Region Parameters.

Parameter	Value
Type	SDRClassifier
Alpha	0.25
Steps	1.5

Many of these parameters were left as default, such as the ones related to the synaptic permanence and decay, since they represent the biological link between the known theory of how the neocortex works and its applicability to the network.

4.1.2. Metrics and Evaluation

This study aims to predict the next day's close value of the market for a given company. Three metrics were used to compute the results: root mean square error (RMSQ), MAPE, and absolute average error (AAE) [25].

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{n}\left(\frac{\hat{x}_i - x_i}{x_i}\right)^2} \tag{2}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{(\hat{x}_i - x_i)}{x_i}\right| \times 100 \tag{3}$$

$$AAE = \frac{\frac{1}{n}(\sum_{i=1}^{n}|\hat{x}_i - x_i|)}{\left(\frac{1}{n}\sum_{i=1}^{n} x_i\right)} \tag{4}$$

Since the HTM is supposed to be a continuous learning theory, there are no training/validation/test sets; the data are learned and predicted continuously. To access the learning, the metrics were taken on three moments: to the entire dataset, 365 days before the declaration of the Covid-19 pandemic, and after the declaration. With these three moments, it is possible to gain a better understanding of how quick (in terms of input data needed) the algorithm is to achieve good previsions, while inferring how it adapts to dramatic changes in the input data (in this case, as a consequence of the pandemic).

5. Results

The results were obtained by forecasting the value 'close', concerning the next day, of the stock market for seven different data sets, using the same parameters in the algorithm.

Table 9 shows the values MAPE, RMSE, and AAE obtained for the three different moments, explained in the previous section:

Table 9. Minimum values of each parameter.

	Total			365 Days Before Pandemic			After Pandemic		
Company	**MAPE**	**RMSE**	**AAE**	**MAPE**	**RMSE**	**AAE**	**MAPE**	**RMSE**	**AAE**
Amazon	1.6067	18.8210	8.3973	1.4366	36.4290	25.1878	2.0034	66.3107	51.6145
Google	1.2537	11.9884	6.7297	1.2393	21.8242	14.6977	1.9062	35.3589	25.3776
Johnson & Johnson	0.7320	1.1232	0.6765	0.8593	1.8820	1.1664	1.4166	3.0066	1.9545
Visa	1.2811	1.5859	0.8056	1.1627	2.8037	1.9013	2.1031	5.3197	3.7043
Disney	1.1345	1.2230	0.7204	1.1339	2.1880	1.4229	2.3098	3.5803	2.5156
McDonalds	0.8637	1.5649	0.8731	0.8791	2.6392	1.7090	1.8123	5.3295	3.2158
HCA	1.4504	1.7493	1.0308	1.3894	2.6339	1.8133	3.0940	4.5305	3.1896

In the following graphics (Figures 3–9), the predicted vs. actual values are displayed along the time axis. The algorithm kept a good performance, following the trends of market 'close' value through time, for all datasets. As expected, the algorithm suffered in its previsions around the time of the declared pandemic; however, it was able to achieve some stability afterwards, in line with the possible stability that the stock market can offer in such an unstable time.

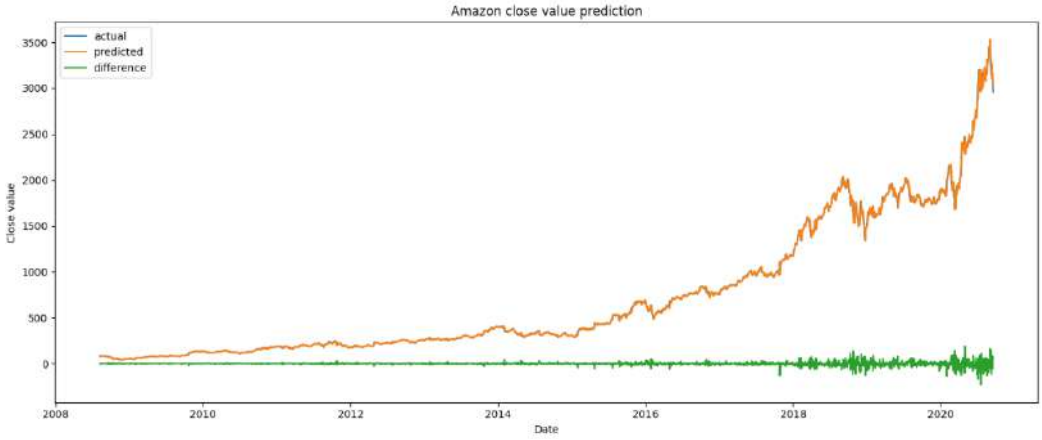

Figure 3. Amazon 'close' value prediction through time.

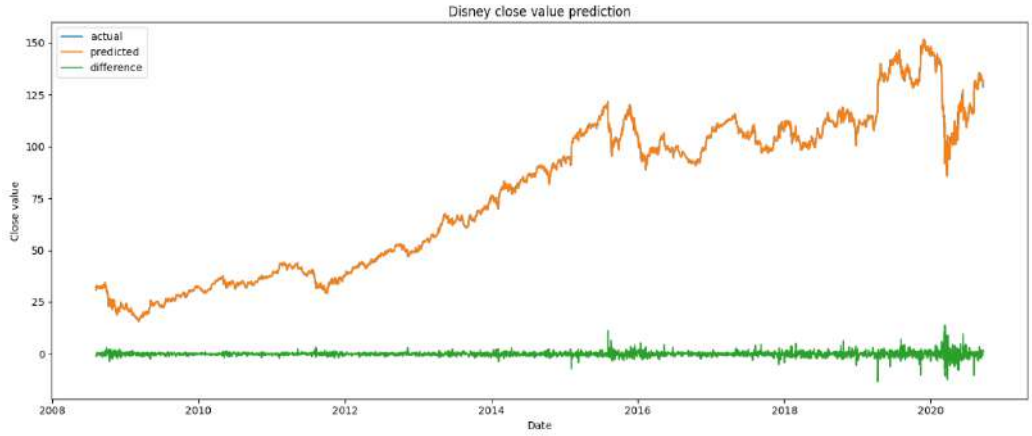

Figure 4. Disney 'close' value prediction through time.

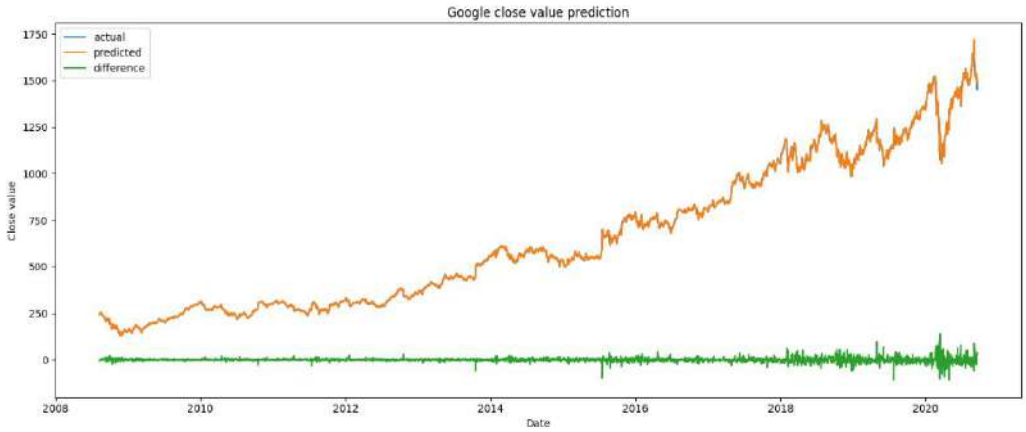

Figure 5. Google 'close' value prediction through time.

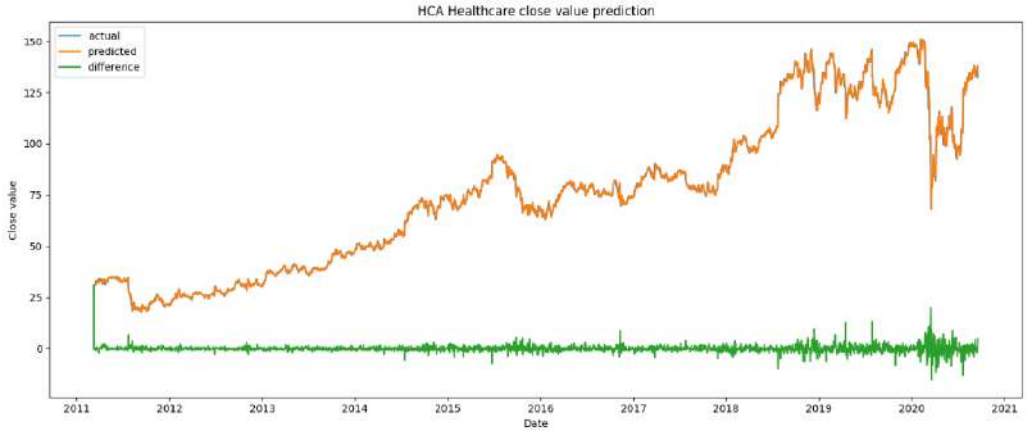

Figure 6. HCA 'close' value prediction through time.

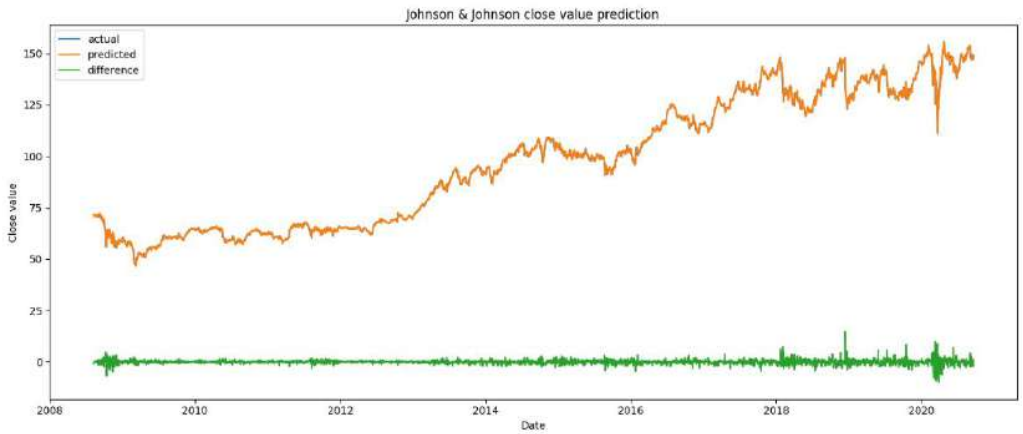

Figure 7. Johnson & Johnson 'close' value prediction through time.

It is also visible by the analysis of the graphics presented that although the value dropped significantly at the beginning of 2020, there is a trend of a continuous rise of the stock.

It is possible to infer that the algorithm learned the patterns quickly, making predictions that were very close to the actual ones with few data. The MAPE values were lower for every dataset in the more stable period before the pandemic, except for the McDonald's and Visa datasets, which received better results in the total period. All MAPE values increased for the post-pandemic period, although not as much for the Amazon dataset—this can be explained by the more stable stock pricing in this company. In general,

the RMSE and AAE values increased through time; since these are not percentage metrics, and the data are not normalized, this increase can be explained by the higher 'close' values in the stock market in the last few years across all datasets.

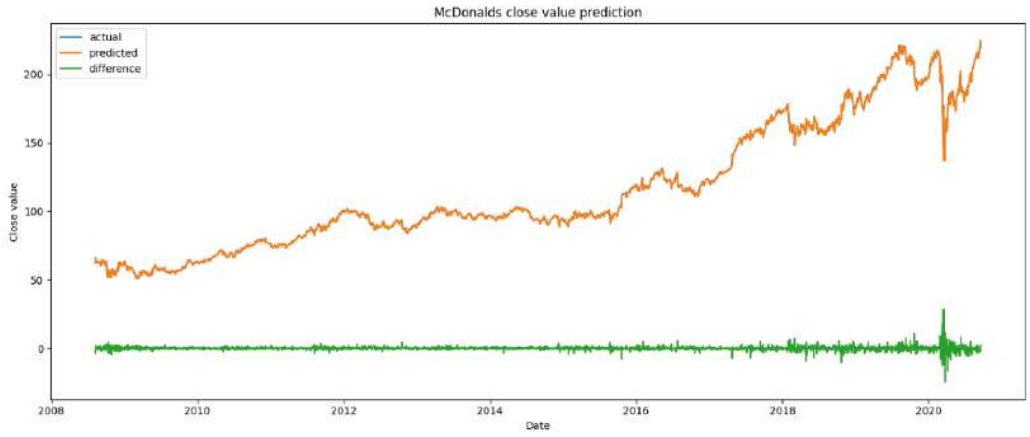

Figure 8. McDonald's 'close' value prediction through time.

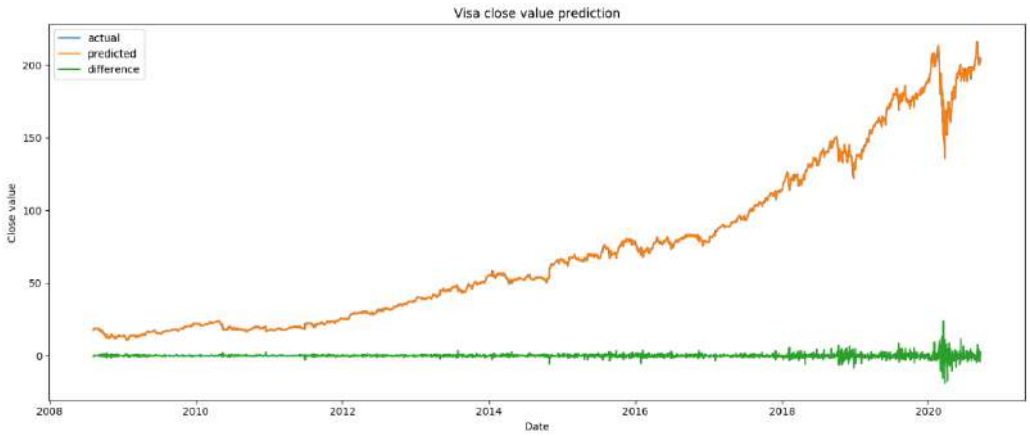

Figure 9. Visa 'close' value prediction through time.

The results obtained in this experiment were very promising, showing that the HTM theory provides a solid framework for time series forecasting, achieving good predictions with few data. Furthermore, the algorithm maintained a good performance across the various datasets: through time, being robust to temporal noise, a bigger complexity of data, and a disruption in the input data caused by the pandemic.

Because of the way HTM works, it is hard to make a rigorous comparison with other methods, which normally divide datasets into training and testing batches.

Besides, in this study, the data used are specific to some S&P 500 companies, ranging from 3 January 2006 until 18 September 2020, contrary to what is observed in the literature, where the time range is typically smaller and no designation of the companies is made—although, some comparisons and findings can be discerned. In [13], the SMA network obtained a MAPE of 11.45% for only a short period of a year, a value worse than what was obtained in the present study for any company for the whole time period available on the datasets. The other two studies presented previously on Section 2, [12,14], related to the forecasting of the next day 'close' value using different LSTM networks, obtained better MAPE values. However, it cannot be stated that these networks perform better, since only a small percentage of the datasets are used for testing and rely on massive training sessions. These methods do not rely on an online continuous learning mechanism such as HTM.

6. Discussion and Conclusions

The advancements of how our brains work biologically may lead to new and revolutionary ways of achieving a true machine intelligence, the aim of the HTM theory. This theory should evolve through the years and help the science community to solve problems typically solved by Machine Learning; specifically Deep Learning in the last few years.

The proposed HTM network obtained good results in the time series forecasting of close values of the stock market, for seven different datasets, through time, proving it can be a great methodology to make predictions while being robust to noise in the data, both in a temporal and spatial axis. It is shown that the network can adapt to different datasets in the same range of problems, with no different hyper-parameter tuning, unlike LSTM and other Deep Learning models; this attribute of HTM models is linked to the known properties of the human cortical neurons and the representation of SDR. Another key difference from other Deep Learning models is that HTM learns continuously, without the need for a specific training dataset; the model learns and predicts continuously. The known experiments where the 'close' value of the stock market is predicted use a classic approach, where training/validation/test dataset tuning is applied to the comparison between models, which is difficult in terms of prediction accuracy; moreover, classically, the data are normalized and suffer a lot of data pre-processing, contrary to the HTM network, where the raw input is only transformed into an SDR, keeping its semantic characteristics.

7. Future Work

As the HTM theory develops, bringing new perspectives of the human intelligence and learning process, such as grid cells [26], it should grab more attention from the data science community, as it will provide a great framework for intelligence and learning.

With regards to future work, there are several possibilities that stand out:

- The combination of this theory with other methods of machine learning. In this way, high dimensional temporal learning problems requiring pre-processing and feature extraction can be solved before creating a sparse representation of the raw input;
- The application of this theory, or even the combination mentioned above, to the in-depth study of the impact of the pandemic on stock prediction;
- The extension of the application of this theory or combination to all S&P 500 companies and to other markets.

We believe that this approach has the most value, since not only does it prove that it is possible to obtain good results with HTM, but it also encourages future research and applications in this same field.

Author Contributions: Each of the authors made substantial contributions to the conception of the article, pleasantly approving the submitted version. Conceptualization, R.S., T.L., A.A. and J.M.; methodology, R.S., T.L., A.A. and J.M.; software, T.L.; validation, A.A. and J.M.; formal analysis, R.S.; investigation, R.S., T.L., A.A. and J.M.; resources, A.A. and J.M.; data curation, R.S., T.L., A.A. and J.M.; writing–original draft preparation, R.S., T.L.; writing–review and editing, R.S.; visualization, R.S., T.L.; supervision, A.A. and J.M.; project administration, J.M.; funding acquisition, J.M. All authors have read and agreed to the published version of the manuscript.

Funding: This work is funded by "FCT—Fundação para a Cieˆncia e Tecnologia" within the R&D Units Project Scope: UIDB/00319/2020. The grant of R.S. is supported by the European Structural and Investment Funds in the FEDER component, through the Operational Competitiveness and Internalization Programme (COMPETE 2020). [Project n. 039479. Funding Reference: POCI-01-0247-FEDER-039479].

Acknowledgments: We thank the administrative staff of the University of Minho for their availability.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

AAE	Absolute Average Error
ANN	Artificial Neural Network
ARIMA	Autoregressive integrated moving average
CAD	Contextual Anomaly Detector
CMIP	Continuous Multi-Interval Prediction
DNN	Deep Neural Network
GRU	Gated Recurrent Unit
HTM	Hierarchical Temporal Memory
KNN	K-Nearest Neighbor
LSTM	Long short-term memory
MAPE	Mean Average Percentage Error
NAB	Numenta Anomaly Benchmark
NUPIC	Numenta Platform for Intelligent Computing
NYSE	New York Stock Exchange
PCA	Principal Component Analysis
PNL	Profit and Loss
RMSE	Root Mean Square Error
SAE	Stacked Autoencoder
SDR	Sparse Distributed Representation
SMA	Simple Moving Average
SP	Spatial Pooler
SVM	Support vector machine
TM	Temporal Memory
WHO	World Health Organization
WT	Wavelet Transform

References

1. Neto, C.; Brito, M.; Peixoto, H.; Lopes, V.; Abelha, A.; Machado, J. Prediction of Length of Stay for Stroke Patients Using Artificial Neural Networks. In *World Conference on Information Systems and Technologies*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 212–221.
2. Ghazanfar, A.A.; Schroeder, C.E. Is neocortex essentially multisensory? *Trends Cogn. Sci.* **2006**, *10*, 278–285. [CrossRef] [PubMed]
3. Purdy, S. Encoding data for HTM systems. *arXiv* **2016**, arXiv:1602.05925.
4. Cui, Y.; Ahmad, S.; Hawkins, J. Continuous Online Sequence Learning with an Unsupervised Neural Network Model. *Neural Comput.* **2016**, *28*, 2474–2504. [CrossRef] [PubMed]
5. Maltoni, D. Pattern Recognition by Hierarchical Temporal Memory. 2011. Available online: http://dx.doi.org/10.2139/ssrn.3076121 (accessed on 12 January 2021).
6. Neves, J.; Martins, M.R.; Vilhena, J.; Neves, J.; Gomes, S.; Abelha, A.; Machado, J.; Vicente, H. A Soft Computing Approach to Kidney Diseases Evaluation. *J. Med. Syst.* **2015**, *39*, 1–9. [CrossRef] [PubMed]
7. Neves, J.; Vicente, H.; Esteves, M.; Ferraz, F.; Abelha, A.; Machado, J.; Machado, J.; Neves, J.; Ribeiro, J.; Sampaio, L. A Deep-Big Data Approach to Health Care in the AI Age. *Mob. Netw. Appl.* **2018**, *23*, 1123–1128. [CrossRef]
8. Liu, C.; Wang, J.; Xiao, D.; Liang, Q. Forecasting S&P 500 Stock Index Using Statistical Learning Models. *Open J. Stat.* **2016**, *6*, 1067–1075. [CrossRef]
9. Chong, E.; Han, C.; Park, F.C. Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies. *Expert Syst. Appl.* **2017**, *83*, 187–205. [CrossRef]
10. Minh, D.L.; Sadeghi-Niaraki, A.; Huy, H.D.; Min, K.; Moon, H. Deep Learning Approach for Short-Term Stock Trends Prediction Based on Two-Stream Gated Recurrent Unit Network. *IEEE Access.* **2018**, *6*, 55392–55404. [CrossRef]
11. Nelson, D.M.Q.; Pereira, A.C.M.; de Oliveira, R. Stock market's price movement prediction with LSTM neural networks. In Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN), Anchorage, AK, USA, 14–19 May 2017; pp. 1419–1426.
12. Liu, H.; Long, Z. An improved deep learning model for predicting stock market price time series. *Digit. Signal Process.* **2020**, *102*, 102741. [CrossRef]
13. Kulkarni, D.; Jadha, D.; Dhingra, D.D. Time Series and Data Analysis and for Stock and Market Prediction. In Proceedings of the 3rd International Conference on Innovative Computing and Communication, Ho Chi Minh City, Vietnam, 14–17 January 2020.
14. Bao, W.; Yue, J.; Rao, Y. A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PLoS ONE.* **2017**, *12*, e0180944. [CrossRef] [PubMed]
15. Gabrielsson, P.; König, R.; Johansson, U. Evolving Hierarchical Temporal Memory-Based Trading Models. In *European Conference on the Applications of Evolutionary Computation*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 213–222.

16. Ahmad, S.; Lavin, A.; Purdy, S.; Agha, Z. Unsupervised real-time anomaly detection for streaming data. *Neurocomputing* **2017**, *262*, 134–147. [CrossRef]
17. Wu, J.; Zeng, W.; Yan, F. Hierarchical Temporal Memory method for time-series-based anomaly detection. *Neurocomputing* **2018**, *273*, 535–546. [CrossRef]
18. Anandharaj, A.; Sivakumar, P.B. Anomaly Detection in Time Series data using Hierarchical Temporal Memory Model. In Proceedings of the 2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, 12–14 June 2019; pp. 1287–1292.
19. Cui, Y.; Surpur, C.; Ahmad, S.; Hawkins, J. A comparative study of HTM and other neural network models for online sequence learning with streaming data. In Proceedings of the 2016 International Joint Conference on Neural Networks (IJCNN), Vancouver, BC, Canada, 24–29 July 2016; pp. 1530–1538.
20. Struye, J.; Latré, S. Hierarchical temporal memory and recurrent neural networks for time series prediction: An empirical validation and reduction to multilayer perceptrons. *Neurocomputing* **2020**, *396*, 291–301. [CrossRef]
21. Kang, H.-S.; Diao, J. An Integrated Hierarchical Temporal Memory Network for Continuous Multi-Interval Prediction of Stock Price Trends. In *Software and Network Engineering*; Springer: Berlin/Heidelberg, Germany, 2012; Volume 413, pp. 15–27.
22. Yilmazkuday, H. COVID-19 Effects on the S&P 500 Index. Available online: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3555433 (accessed on 2 January 2021).
23. Altig, D.; Baker, S.; Barrero, J.M.; Bloom, N.; Bunn, P.; Chen, S.; Davis, S.; Leather, J.; Meyer, B.; Mihaylov, E.; et al. Economic Uncertainty Before and During the COVID-19 Pandemic. *J. Public Econ.* **2020**, *191*, 104274. [CrossRef] [PubMed]
24. Stock Market News. 2021. Available online: https://www.marketwatch.com (accessed on 2 January 2021).
25. Hong, W.C.; Li, M.W.; Fan, G.F. *Short-Term Load Forecasting by Artificial Intelligent Technologies*; MDPI-Multidisciplinary Digital Publishing Institute: Basel, Switzerland, 2019.
26. Klukas, M.; Lewis, M.; Fiete, I. Efficient and Flexible Representation of Higher-Dimensional Cognitive Variables with Grid Cells. 2020. Available online: https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1007796 (accessed on 2 January 2021).

electronics

MDPI

Article

Bus Dynamic Travel Time Prediction: Using a Deep Feature Extraction Framework Based on RNN and DNN

Yuan Yuan [1,2], Chunfu Shao [1,*], Zhichao Cao [3], Zhaocheng He [4], Changsheng Zhu [5], Yimin Wang [4] and Vlon Jang [6]

1 Key Laboratory of Transport Industry of Big Data Application Technologies for Comprehensive Transport, Beijing Jiaotong University, Beijing 100044, China; 13114240@bjtu.edu.cn
2 School of Automotive and Transportation, Shenzhen Polytechnic College, Shenzhen 518055, China
3 School of Transportation and Civil Engineering, Nantong University, Nantong 226000, China; caozhichao@bjtu.edu.cn
4 Guangdong Provincial Key Laboratory of Intelligent Transportation System, Sun Yat-sen University, Guangzhou 510006, China; hezhch@mail.sysu.edu.cn (Z.H.); wangyim6@mail2.sysu.edu.cn (Y.W.)
5 School of Intelligent Equipment, Shandong University of Science and Technology, Huangdao District, Qingdao 266590, Shandong Province, China; zhuchangsheng@sdust.edu.cn
6 Singularity Cloud, Beijing 100000, China; wangqingbaidu@gmail.com
* Correspondence: cfshao@bjtu.edu.cn

Received: 24 September 2020; Accepted: 29 October 2020; Published: 8 November 2020

Abstract: Travel time data is an important factor for evaluating the performance of a public transport system. In terms of time and space within the nature of uncertainty, bus travel time is dynamic and flexible. Since the change of traffic status is periodic, contagious or even sudden, the changing mechanism of that is a hidden mode. Therefore, bus travel time prediction is a challenging problem in intelligent transportation system (ITS). Allowing for a large amount of traffic data can be collected at present but lack of precisely-conducting, it is still worth exploring how to extract feature sets that can accurately predict bus travel time from these data. Hence, a feature extraction framework based on the deep learning models were developed to reflect the state of bus travel time. First, the study introduced different historical stages of bus signaling time, taxi speed, the stop identity (ID) of spatial characteristics, and real-time possible arrival time, signified by fourteen spatiotemporal characteristic values. Then, an embedding network is proposed to leverage a wide and deep structure to mate the spatial and temporal data. In order to meet the temporal dependence requirements, an attention mechanism for a Recurrent Neural Network (RNN) was designed in this research in order to capture the temporal information. Finally, a Deep Neural Networks (DNN) was implemented in this research in order to achieve the dynamic bus travel time prediction. Two case studies of Guangzhou and Shenzhen were tested. The results showed that the performance of the algorithm was more efficient than that of the traditional machine-learning model and promoted by 4.82% compared to the deep neural network applied to the initial feature space. Moreover, the study visualized the weighted cost of attention on the bus's travel time features during a certain running state. Therefore, the study demonstrated the proposed model enabled to understand the characteristic data of transit travel time with visualization.

Keywords: dynamic bus travel time prediction; wide and deep; data fusion; attention; recurrent neural network; deep neural networks

1. Introduction

Bus travel time prediction is an important component of an intelligent transport system (ITS). The precise capturing of real-time travel information facilitates the choice of an optimal route by a traveler. Additionally, with unforeseen events occurring, traffic managers adjust departure schedules in real time to ensure the service quality of a system [1,2]. Nevertheless, the travel time of the same bus route in the same city is dynamic due to the nature of bus operation because of frequent traffic congestion, traffic accidents, and road construction. Therefore, it is necessary to focus on a real-time and dynamic bus travel time prediction model in depth in order to further improve traffic efficiency.

Bus travel time prediction has three dependencies. (1) Time dependence [3]: Due to the strong periodicity of passenger demand, bus scheduling also has a certain periodicity. Moreover, bus travel time also depends on the tendency of recent historical travel times. (2) Spatial dependence: The travel time of a particular line is influenced not only by the current traffic state variables of the running line but also by the traffic state variables of the entire bus line [4]. (3) Exogenous dependence: Some exogenous variables, weather conditions, and emergencies may have a great impact on traffic timing prediction [5]. However, driven by big traffic data, a challenge arises: can one gain broad utilization of the latent knowledge hidden in big traffic data in order to predict bus travel time?

Currently, the original statistical-based parameter models (such as K-Nearest Neighbor (KNN) or ARIMA) or machine learning models (such as Support Vector Machine (SVM)) are experiencing more and more difficulty in meeting the requirements of big data in some areas, while the research field of neural networks is active [6,7]. Recently, the neural network shallow prediction model has been used in most scenarios [8]. However, these models have limitations when dealing with large historical data sets and complex nonlinear functions [9].

Deep learning integrates multi-layer architecture and regression to extract inherent features in an end-to-end way. Based on the analysis of a large amount of real-time and historical traffic data, a deep neural network model can deal with the nonlinear characteristics of traffic data and obtain more precise prediction results [10]. However, real-time dynamic bus travel time prediction is very complex, and it involves complex space-time features [11,12]. Moreover, the potential traffic status and traffic events are in a hidden mode. Therefore, the development of a deep learning model is not well suited to capturing the deeper characteristics of bus travel time effectively [13].

For the critical issue of interpreting the space-time features of bus travel time, data-driven methods and neural network methods have been doubted to have this ability [5]. However, there have been a few research literature references that have focused on the diverse traffic features affecting the final prediction of bus travel time. Therefore, this research aimed to explore a new methodology for handling a large number of spatio-temporal features by using deep learning models for the prediction of bus travel time.

In order to solve the problem of focusing on big data feature extraction for bus travel time prediction, in this study, a dynamic real-time bus travel time prediction method was proposed based on a deep learning feature extraction framework and data fusion. In this research, bus travel times were divided into running times and dwelling times, and Global Positioning System (GPS) speeds were added for taxis and buses, as well as travel times based on real-time speeds in order to predict dynamic bus travel times, as indicated in Figure 1. In summary, the main contributions of the proposed approach are those reported below.

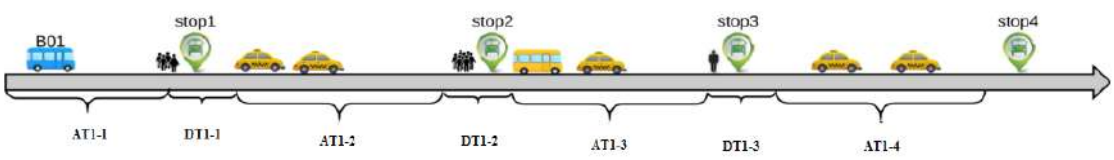

Figure 1. Examples of bus and taxi traveling process.

Based on the prediction of bus travel time, in this research, a new heterogeneous feature extraction framework was proposed based on the recurrent neural network (RNN) model of embedding wide and deep (WD) and an attention mechanism. The framework was proposed in order to gain a deep understanding of the spatio-temporal features and intrinsic connections of the characteristics related to bus travel time and to visualize the connections.

Fourteen spatial and temporal features were introduced, including stop Identities (IDs) as special characteristics, bus dwelling times at different historical levels, real-time GPS bus speeds with real-time possible transit times obtained based on real-time bus speeds as temporal features. These features have not been analyzed together in previous surveys. Lastly, multiple super positions of the RNN and Deep Neural Networks (DNNs) were employed to reduce the residual heterogeneous data fusion and real-time dynamic bus travel time prediction. A novel system for real-time dynamic bus travel time prediction was offered.

To verify the model's stability and generalization ability, the model was tested on the datasets of the Guangzhou No. 226 bus and the Shenzhen No. 113 bus. These buses ran along the main roads in large urban centers. Both of the experiments achieved good results. Other studies never tested their models in different cities.

2. Literature Review

Ever since the rapid development of deep learning methods occurred, the potential for processing large-scale high-dimensional data has been maturing [3,10,14–18].

Recurrent Neural Network (RNN), which is a distinctive construction of deep learning models, is widely used to solve sequence problems [19]. This type of network extends a DNN by repeatedly connecting hidden layers in different timestamps. In this network structure, memory units can dynamically model sequence data. Lately, some studies in the field of transportation has begun to seek RNN to solve the problem of time series predictions, such as traffic flow [20], traffic speed [10], and travel time prediction [21]. Petersen et al. (2019) and He et al. (2020) developed an RNN architecture for the prediction of bus travel times. They demonstrated that the network could capture long-term time dependencies in traffic data, as shown in Table 1 [6,22].

Table 1. A comparison of travel time prediction approaches.

Paper	Model	Feature Extraction	Classification	Factors			Number of Cities	Size of Cities
				AVL	Speed	Dwelling Time		
[23]	SVM, ANN	Cluster	Temporal	Yes	Bus	Historical mean	1 city	megacity
[4]	SVM ANN KNN	No	Temporal	Yes	Taxi	Predicted	1 city	megacity
[22]	CNN RNN	No	Spatio-temporal	Yes	No	No	1 city	metropolis
[6]	RNN	Cluster	Temporal	Yes	No	Historical mean	1 city	megacity
[5]	DNNS	PCA, Cluster	Spatio-temporal	Yes	No	No	1city	metropolis
Ours	RNN DNNs	Embedding Wide and-Deep Attention	Spatio-temporal	Yes	Bus and taxi	Multiple stages	2 cities	Megacities

Note: AVL means automatic vehicle location.

Deep Neural Networks (DNN) has deep fully connected neural layers. An individual DNN does not require the manual extraction of features, and it learns in a supervised way. For our specific problem, the factors that caused congestion, queue delays, and traffic flow came from the fuzzy interaction with complex features. DNN is a multi-layer deep structure that can extract features from data and reveal important potential or hidden structures. Furthermore, DNN provides a powerful and new way to learn how these features interact. Abdollahi et al. (2020) trained a deep, multi-layer perceptron to predict bus travel time [5].

Although the exploration of deep learning models with applications to bus travel time prediction has achieved delightful results, there are still some limitations in these fields. A comparison of the latest bus travel time prediction studies is shown in Table 1.

There are few existing studies on bus travel time prediction using deep learning methods. It has been even rarer to study real-time dynamic bus travel time prediction. In the only studies, although the deep learning methods had a powerful ability to handle large amounts of data and high-dimensional data, the gap between large-scale data and its shallow structure, the gap between full connectivity and rich features [5,13], and the hidden patterns of potential traffic states and traffic events made it difficult for the above models to derive representative features from the rich feature data set. In other words, there has been a lack of systematic, perfect, and in-depth feature learning. Therefore, it is necessary to develop a deep-seated deep learning architecture that fully reflects the features of bus travel time prediction.

The existing studies of the prediction of bus travel time with feature learning still belong to the category of shallower feature learning. Examples include geospatial feature analysis, principal component analysis (PCA), and unsupervised learning algorithms (K-Means) to extract spatial features, and a deep-stacked auto-encoder (SAE) to represent low-dimensional features [5,23]. Using the deep structure of a Recurrent Neural Network (RNN) in time, the historical sequence information was automatically remembered in the model structure [6,22]. The spatial features of the data were extracted from the Convolutional Neural Network (CNN) for use by the Long Short-Term Memory (LSTM) network [22]. DNNs were also used to predict bus travel times after feature extraction [5]. However, most of the research on bus travel time has been shallow in terms of the feature learning structures [5,6,23], lack of feature learning [4,22], or lack of feature learning depth and related breadth. Therefore, it is of great significance to develop a deep feature extraction structure that fully reflects the characteristics of travel time.

The study proposed a neural network that integrated embedded, wide and deep algorithm, and attention mechanisms, and introduced them into a dynamic bus travel time prediction model for design. The extraction framework made use of the non-static space-time correlation existing in urban public transport networks and discovered complex models that traditional methods could not capture. Our study also visualized the RNN model to interpret the impact of various spatial-temporal features on the prediction of dynamic bus travel times, which challenged the traditional neural network approach in the public transport field.

3. Prediction Model

3.1. Feature Extraction Framework

The underlying feature extraction framework was proposed. The framework was composed of Embedding, Wide and Deep, and Attention models.

3.1.1. Embedding

One-hot encoding is one of the most common methods used in dealing with discrete data. Taking Wednesday as an example, it is the third day of a week, and (0, 0, 1, 0, 0, 0, 0) is used to represent three out of seven. One-hot encoding treats each dimension independently, but these representations might not be capable of catching the similarity of each variable; for example, Saturday and Sunday during peak periods might be similar. Additionally, one-hot encoding is too sparse, which is difficult for a deep learning model to deal with [24]

Embedding is a particularly effective method to solve the problems mentioned above, which can be formalized into the following expression:

$$embedding = map\left(X \in R^{N\times 1} \rightarrow X_E \in R^{N\times d}\right),$$

where N denotes the words, d is the embedding size, X is the feature, X_E is the recoded features, and R is the data feature set.

Similar to the data structure mentioned in Section 3.2, the features hours (time of data), day (day of week), and distance, which was used as the station ID instead of bus travel distance, were discrete

data features. In order to capture more similarity for each feature, the study implement an embedding model for each feature, as shown in Figure 2.

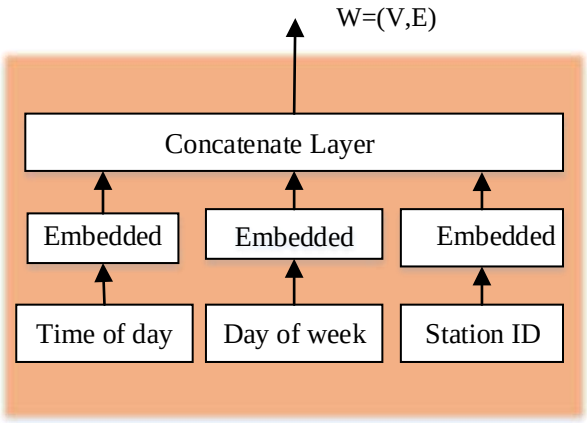

Figure 2. Embedding extraction of discrete features.

3.1.2. Wide and Deep

The bus travel time prediction task included both discrete features and continuous features. The dimensions of the discrete features were much smaller than those of continuous features, and the model would be more susceptible to the impact of continuous data if these features were directly input into the deep model for training. To solve this problem, our study were inspired by the designation of Wide and Deep, shown in Figure 3, for which the core idea was to combine the memory ability of the linear model with the generalization ability of the deep model. In this study, discrete features were applied, such as hours, day, and distance to the wide side, and continuous features were applied to the deep side.

a. The Wide Component

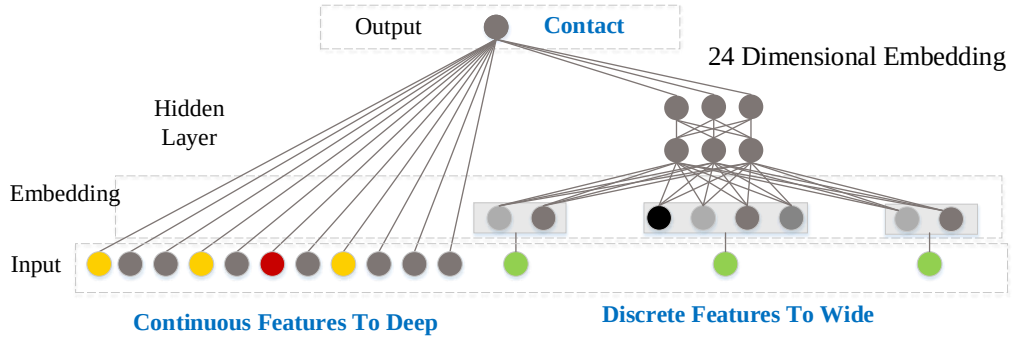

Figure 3. Illustration of the Wide and Deep model improved based on Cheng et al. (2016) [25].

Since the wide side had a high memory ability, it could be used to map the interrelationships after the embedding of discrete features turned into continuous features. Therefore, the discrete features were input into the wide side.

The wide side emphasized features that had often co-occurred in the past, also known as "frequent co-existence features." For example, "Monday", "7:30–9:30," and "station 2–3" often appeared together. The relationship between these three terms allowed us to explain why they occurred so often together. In fact, the memory could be effectively captured by adding interaction items to a broad learning model. The wide side is a generalized linear model for which the form is

$$embedding_{fd} = W^{T}_{V \times N} \times one\_hot(f_d) + b \quad (1)$$

where, $embedding_{fd}$ denotes the predicted discrete outputs, which were treated as traffic state features, W is a $V \times N$ matrix, and V is the set size of the corresponding discrete features. $one\_hot(f_d)$ is the one-hot encoding corresponding to the discrete features.

b. The Deep Component

Generalization is the use of new feature interactions that have occurred rarely or never in historical data, such as "V3 = 40.1" rarely co-occurring with "DT1 = 3" at the same time. Therefore, the wide side could not be used to predict situations that had occurred rarely or never in historical data. However, deep neural networks could find correlations between invisible features.

The deep side had strong feature generation ability, so continuous features were input into the deep side. This allowed the model to capture correlations between different continuous features. The learning model for the expression of continuous features can be expressed as follows:

$$feature_{fc} = W^T_{M \times N} \times f_c + b \tag{2}$$

where, $feature_{fc}$ represents the Continuous features in the bus operation data, W is the vector $M \times N$, M is the size of the continuous features, N is the size of the embedding, f_c is the hidden layer of the neural network, and b is the offset.

c. Joint Training of the Wide and Deep Model

Finally, the features calculated from the two branches were spliced together to obtain the features extracted from the original data. These features can be expressed as

$$feature_f = embedding_{fd} \oplus feature_{fc} \tag{3}$$

where, $feature_f$ is formed by combining discrete features and continuous features, and $\oplus$ is the split joint.

3.1.3. Attention Mechanism

In this study, the attention mechanism was introduced into the task, and our attention-based RNN model that used spatial-temporal features to predict dynamic bus travel times and capture the importance of spatial-temporal features at different locations was proposed.

The attention model performed element-wise multiplication with each feature matrix to obtain a weighted feature matrix, as shown in Figure 4:

$$attn\_feature_t = attn \otimes feature_t. \tag{4}$$

The goal of the attention model was to learn an attention weight matrix $attn\_featureT_t$. In this study, an RNN model was proposed in which h_t was used to learn weights at different states. Each element could be interpreted as the relative importance of $T^f\left(feature_f atT\right)$. The activation function *sigmoid* between the output and the hidden layer could limit the output to between 0 and 1:

$$attn\_featureT_t = sigmoid \times \left(W^T h_t\left(T^f\right) + b\right) \tag{5}$$

In the formula, W is a T^f matrix, and h_t is a mapping between the input and hidden neurons. In this study, Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU) was used in a fully-connected RNN network. Then the spatial-temporal matrix T^f of the historical bus journey point by point was multiplied with the attention matrix $attn\_featureT_t$ (as shown in Formula (2) to obtain a weighted bus journey time matrix for further learning. Therefore, the final attended feature was

$$attn\_T_{rnn} = attn\_featureT_t \otimes T^f \tag{6}$$

In the formula, $attn\_T_{rnn}$ is the weighted eigenvector, and ⊗ represents the multiplication of the corresponding elements one by one.

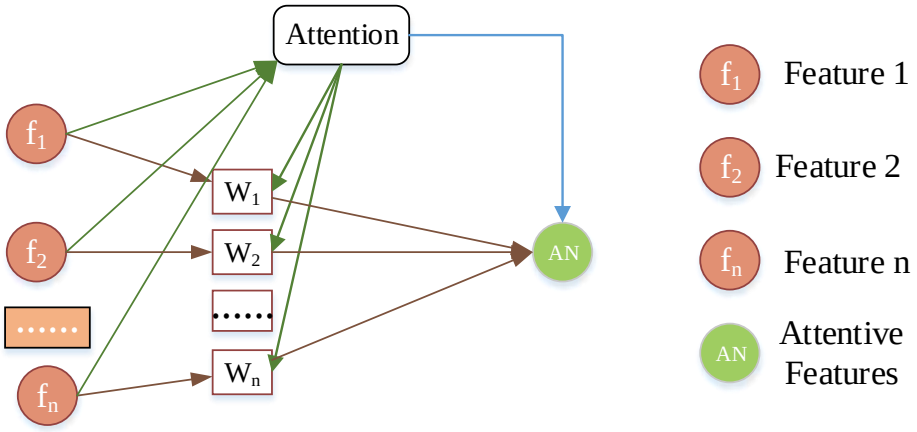

Figure 4. Framework of attention model.

The RNN was used to model the series data, and the RNN hidden features were used to weigh the features. The formation process of the weighted travel time matrix is shown in Figure 5.

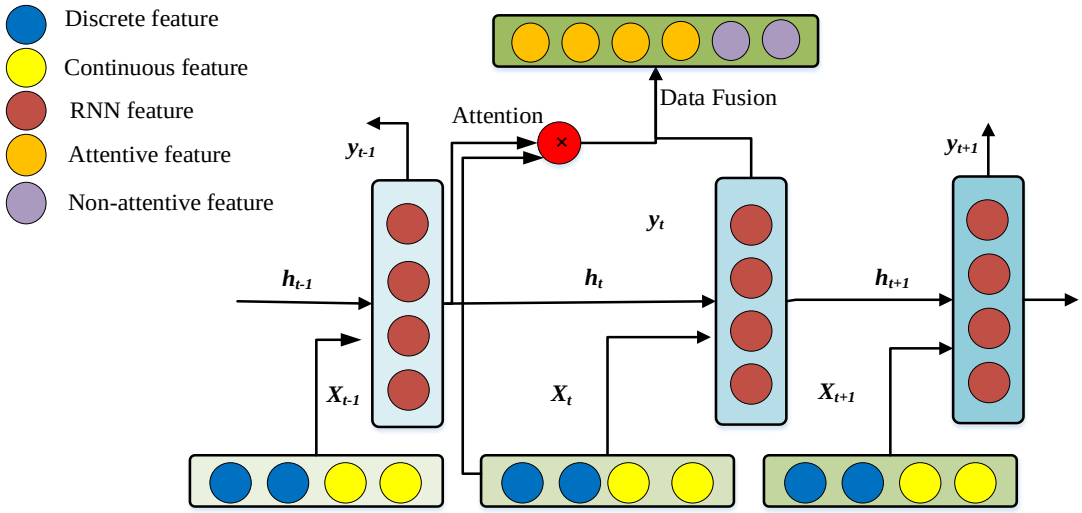

Figure 5. Illustration of the RNN and the attention model.

3.2. Dynamic Bus Travel Time Prediction

The bus travel time prediction procedures underlying are formed by embedding module, Wide and Deep module, RNN and DNN.

Step1: embedding model compresses and encodes discrete data, extracting correlations between discrete features.

Step2: since the wide side had a high memory ability, it was used to map the interrelation ships after the embedding of discrete features turned into continuous features. The deep side captures correlations between different continuous features. The features calculated from the two branches were spliced together to obtain the features extracted from the original data. The wide and deep module that fused discrete and continuous features are shown in Figure 6.

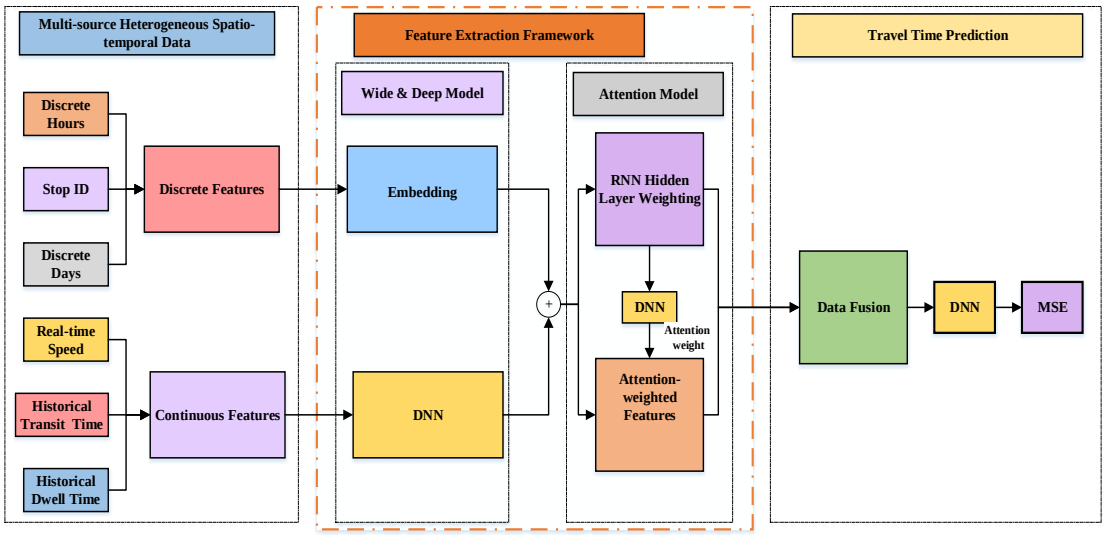

Figure 6. Overview of the model developed for the bus dynamic travel time prediction.

Step3: all of the features were weighted by the attention module. The weighted features were fed into the RNN and DNN models.

DNN is a fully connected deep learning model, which has better ability to obtain the optimal solution. However, there is an insoluble problem with fully connected DNN: it is impossible to model changes in time series. In a normal fully connected network, the hidden layer of DNN can only receive the input at the upper layer at the current moment, while in RNN, the output of neurons can act directly on itself in the next period. In other words, the hidden layer of a recursive neural network can not only receive the input of the previous layer, but also get the input of the current hidden layer at the previous moment. The significance of this change is that it makes the neural network capable of historical memory [26]. In principle, an infinite amount of historical information is well suited for tasks with long-term relevance, such as speech and language. The memory function of RNN is particularly suitable for memorizing and mining sequence data. The multiple combinations and superposition of DNN and RNN can capture the characteristics of the permissible sequence in bus travel time prediction and obtain the optimal solution. Meanwhile, the residual errors can be eliminated by multiple combinations and superposition.

Then Mean Squared Error (MSE) was used in this study to train the model to predict the bus dynamic travel time, as shown in Formula (7):

$$loss_t = \frac{1}{2}\left|target_t - \widetilde{target_t}\right|^2, \tag{7}$$

In the formula, $loss_t$ is a loss function, $target_t$ is the real travel time at time t, and is the predicted value at time t.

The model was built to be end to end, and all of the parameters in the model were trained together. Our general training process is listed as Algorithm 1.

Algorithm 1: Process for Model Training

Input:
    Discrete data f_d in the training dataset
    Continuous data f_c in the training dataset
    Travel time $target_t$ in the training dataset
Output:
    Step 1: Initialize all parameters
    Step 2: Feature extraction
    Step 3: Feed f_d to the Embedding model to obtain $embedding_{fd}$
    Step 4: Feed f_c to the DNN model to obtain $feature_{fc}$
    Step 5: Concatenate all $\{embedding_{fd}, feature_{fc}\}$ as $feature_f$
    Step 6: For all states in a series of traffic data, do
    $SAMPLES = [[f_1, f_2, f_3, \ldots, f_T]]$
        End for stage I feature extraction
        Training Algorithm
        Repeat
    Step 7: Randomly choose a batch of samples in *SAMPLES*
    For each state of the above
    Get RNN output $output_t$
    Get Attention features $attn\_feature_t$
    End of stage II feature extraction
    Concatenate both $\{output_t, att\_feature_t\}$
    Get forecasted travel time $target_t$
    Compute loss on and $target_t$ using the mean standard error
        Back propagation
        Until convergence
End training

4. Data Collection and Feature Definition

4.1. Data Collection

We evaluated our approach using a large number of buses and taxi GPS data, as well as the bus Automatic Vehicle Monitoring (AVL) data collected by the Transport Department of Guangzhou and Shenzhen in the south of China, which are metropolises with populations of over 14.9 million people and 13.2 million people, respectively.

The bus travel time prediction could be divided into a main road with a signal and a road without a signal. Our experiment in Guangzhou and Shenzhen included different signal periods for multiple intersections connected to each other, which was more challenging for the accuracy of urban main road prediction [27].

To test the No. 226 Bus line in Guangzhou City (23.2 km, 28 stations), the dates for 27 sections and the corresponding areas were collected from 5 October 2014, to 9 November 2014. The No. 226 bus ran through the artery roads (such as Huangpu Road and Dongfeng Road). The running time of the vehicles was 6:00–22:00, and the departure time was 10 min.

The Shenzhen data set used data for the No. 113 bus (19.5 km, 23 stations) with 23 sections and the corresponding areas collected from 20 March 2018, to 5 August 2018. The buses ran through the main road, ShenNan Avenue. The running time of the vehicles is 6:10–23:00, and the departure time was about 4–8 min.

4.2. Features and Definition

Firstly, the main reason why existing estimation approaches could not achieve excellent accuracy is the fact that the travel times are impacted by various factors, such as different weather conditions [28,29], temporal variation of peak and off-peak hours [4,30,31], boarding passenger information [32–34], and real-time traffic conditions [35,36]. Some work focus on analyzing the impacts of different factors. In the study of He [37], the traffic state reports from Twitter information is added as additional data support to predict travel times. The results show that knowing real-time traffic condition helps to increase the estimation accuracy. From the analysis results of the above studies, we can observe that the traffic conditions are uncertain and important for travel time prediction [38]. However, bus GPS data are usually infrequent. Especially, the penetration rate of buses in the traffic network is low at low speed. It is less insensitive to irregular traffic conditions than taxis. It can be observed that only limited studies exist that analyses the influence of real-time traffic flow conditions on bus travel times and the correlation between them [4].

Secondly, the data of Shenzhen city is of 2016. In 2016, the working hours of bus lanes in Shenzhen were from 7:30 am to 9:30 am and from 17:30 pm to 19:30 pm on weekdays (except statutory holidays).Taxes are usually allowed to travel on bus lanes during non-bus lane working hours. In addition, in the field observation of taxi operation, it is found that sometimes passengers will park in the bus lane when getting on or off the taxi. As a result, taxis sometimes run on bus lanes.

Moreover, the data of Guangzhou comes from the time when bus lanes have not been implemented. Therefore, at that time, buses and taxis were traveling together. Therefore, bus GPS and taxi GPS were taken into account when considering the traffic status. Additionally, Different studies have different definitions of real-time. Nikolas Julio [39] defined the dynamic travel time prediction as 10 min when studying the use of traffic shock waves and machine learning algorithms to predict bus speed in real time. Qichongb [40] Predicts bus real-time travel time basing on both GPS and RFID data based on the assumption that the traffic flow keeps the same level in an interval of 30 min although he collects GPS data every 30 s. Hans [41] forecasts Real-time bus route state using particle filter and mesoscopic modeling with four loop detectors installed along the same corridor. Archived data provides access to volume and occupancy information collected approximately every minute. In order to predict the dynamic bus travel time, this paper adds the real-time GPS speed data of the bus every 20 s to the feature for dynamic bus travel time prediction, which effectively improves the prediction accuracy.

Allowing for the data of Shenzhen city based on 2016-year when the exclusive hours of bus lanes in Shenzhen were from 7:30 am to 9:30 am and from 17:30 pm to 19:30 pm on weekdays (except statutory holidays). Besides, taxes are indeed allowed to travel on bus lanes during non-exclusive-bus operating hours. In addition, in the field observation of taxi operation, it is found that sometimes passengers will park in the bus lane when getting on or off the taxi. As a result, taxis always run on bus lanes. Moreover, the fundamental data derived from bus GPS and taxi GPS were taken into account in the paper, which are assumed to represent the traffic status of the PT and road transit, respectively.

We selected fourteen characteristic data sets related to bus travel time prediction, including discrete data, continuous data, spatial data, and time data, as shown in Table 2. In this paper, the features of exited studies is refined into multiple stages, and expanded to the speed of bus and taxi rather than that of one kind vehicle, thus making it more comprehensive to reflect the traffic state.

The existed studies on dynamic travel time using deep learning model, especially the dynamic bus travel time, has not yet been considered. Therefore, based on the above eigenvalues, we added the real-time speed collected within 20s of the bus prediction time into the deep learning model proposed in this paper to predict the dynamic bus travel time.

Table 2. Features and definition.

Features	Data type	Temporal/Spatial	Definition
dt1	Continuous	Temporal	Average bus dwell time within 30 min.
dt2	Continuous	Temporal	Average bus dwell time in 30 min at this point in the last week.
dt3	Continuous	Temporal	Average bus dwell time within 30 min on the same day of the last week.
at1	Continuous	Temporal	Average bus travel time within 30 min.
at2	Continuous	Temporal	Average bus travel time within 30 min of the last week.
at3	Continuous	Temporal	Average bus travel time within 30 min on the same day of the last week.
V1	Continuous	Temporal	Average velocity of the probe vehicles within 5 min.
V2	Continuous	Temporal	Average velocity of the probe vehicles in 5 min at this point in the last week.
V3	Continuous	Temporal	Average velocity of the probe vehicles within 5 min on the same day of the last week.
Real-time speed	Continuous	Temporal	The real-time bus speed was used to reflect the real-time traffic status in the bus lane.
Possible time	Continuous	Temporal	The distance between stops divided by the real-time speed of the bus
Day	Discrete	Temporal	Day of the week, day = {1,2,3,4,5,6,7}.
Hours	Discrete	Temporal	Hours of the day, hours = {8,9 . . . ,20}.
Stop-ID	Discrete	Spatial	This reflected the spatial relationship between stops and the impact of different stops on the bus travel time prediction.

5. Evaluation

5.1. Establishment of the Experiment

5.1.1. Platform Configuration

The experimental platform hardware components used in this study were an Intel Core i7 8700 @3.2 GHz and 32G DDR4 Memory. The platform software was Centos7.5. Our experiment was operated using Python 3.6.8 and TensorFlow 1.10.0.

The experimental data can be found in Section 3, and the data features are shown in Table 2.

5.1.2. Missing Data

In the process of collecting data, missing data could not be avoided. In our experiments for this study, the records with missing values were discarded because of the use of an RNN, but all the states for a whole line were not discarded for the study. Instead, our study put the site information in as a discrete feature on the wide side for feature learning, which has been talked about previously. Therefore, for the entire bus travel time sequence, there may have been a sequence, such as 1-2-5-8-10, for which a station with missing data was dropped.

5.1.3. Hyper Parameters

In the experiment, the size of embedding for the research was set to four at the wide side, and the number of hidden DNN nodes at the deep side was set to 16. Finally, the features were concatenated. Each state was converted to 28-dimensional features.

5.2. Evaluation Criteria

Three metrics are often used to evaluate the performance of traffic prediction models. They are the mean absolute percentage error (MAPE) [10], mean absolute error (MAE) [4,10,13], and root mean square error (RMS) [10,13]:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|x_i - \widetilde{x_i}\right|, \tag{8}$$

$$MAPE = \frac{100}{n}\sum_{i=1}^{n}\left|\frac{x_i - \widetilde{x_i}}{x_i}\right|, \tag{9}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \widetilde{x_i})^2}, \tag{10}$$

where, N is the number of samples, i is the number of stations, x_i is the real bus travel time, and $\widetilde{x_i}$ is the predicted bus travel time.

The MAE and MAPE are indicators of regression tasks. Compared with the MAE and MAPE, the RMSE is more sensitive to outliers, and it can amplify larger prediction deviations. It is often used to compare the stability of different prediction models. The MAPE provides prediction errors based on the percentage difference between observed and predicted bus travel times as a measure of the prediction accuracy of the statistical prediction methods. These performance indicators provide a deep understanding of the nature of prediction errors [10].

5.3. Experiment Results

This section describes the evaluation of the accuracy of our approach for this study based on six types of experiments with our proposed model compared to the existing models.

5.3.1. Different Method

To study different methods of accuracy, the following results were compared for the test set of MAPE indicators in this research: The historical average estimate HAV, using only historical information and not joining the floating vehicle average speed information SVR1, historical information with the floating vehicle average speed SVR2 and a prior probability distribution of the bus travel time, the use of Bayesian theory to modify the SVR2-Bayes theorem of the SVR2 experiment results, the linear model, a neural network based on depth within the DNN [5] and RNN [6], and our proposed methods. See Table 3 and Figure 7 [5,6].

Table 3. Results for GuangZhou261.

Error Index	HAV	SVR	SVR1	SVR 2	ANN	SVR 2-Bayes	SVR 2-Bayes2	Our Study
MAPE (%)	18.5	17.98	17.98	16.4	17.23	14.93	14.29	**8.43**

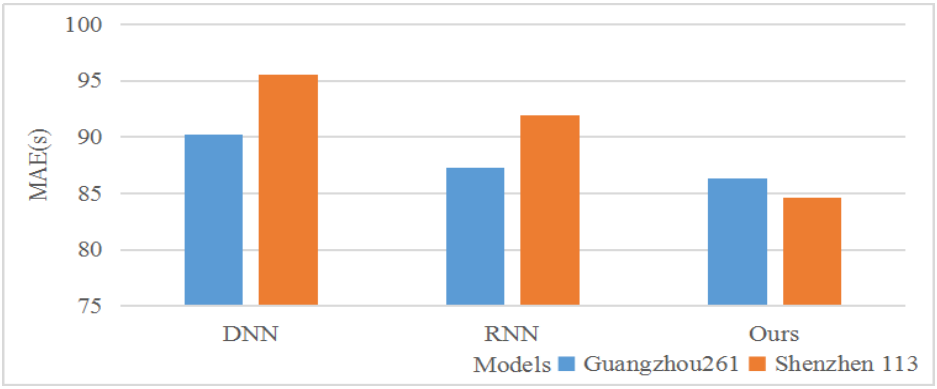

Figure 7. Comparison of different datasets

Table 3 indicates that the DNN [5] and RNN [6] represented relatively advanced artificial intelligence algorithms. The DNN had nearly 5% more absolute promotion than the SVR2-Bayes2 model did. After replacing the model with the RNN, the MAPE could be further reduced by capturing the interdependence between different sites, indicating that the spatial-temporal relationship between sites had a certain impact on the prediction accuracy. However, the RNN itself did not pay attention to the importance of different features in different states. Therefore, our study combined the RNN with Wide and Deep and attention mechanisms to form a feature extraction framework. The accuracy of the

proposed RNN network was further reduced by 0.5% and relatively improved by 5% compared with the RNN network alone.

Based on Guangzhou PT center dataset, in order to study the changes in the travel time at different times for each station, the predicted and true values of the bus travel time model for 8:00 AM were randomly selected. It can be seen from Table 4 that our model could reduce the MAPE by 4–7% compared with svr2-bayes2, indicating that our algorithm had a good performance during the peak or flat peak times.

Table 4. MAPE values for different methods in different periods.

MAPE / Time	SVR2 (%)	SVR2-Bayes (%)	SVR2-Bayes2 (%)	Our Study (%)
Morning (08:00–09:00)	15.99	14.21	13.83	9.51
Evening (17:00–19:00)	16.24	13.69	13.62	6.43
Flat peak	14.37	14.38	12.25	5.21

5.3.2. Different Dataset

The MAE and MAPE are indicators of regression tasks. For different scenarios, we use MAE and MAPE two standards to evaluate the error. First, for a complete bus line, compare the data sets of Guangzhou and Shenzhen with RNN, DNN and our own algorithm, and use MAE to evaluate the error time to compare the operation of the entire line. This is important for traffic managers or bus scheduling and dispatching personnel. The MAPE provides prediction errors based on the percentage difference between observed and predicted bus travel times as a measure of the prediction accuracy of the statistical prediction methods. Then, for the morning peak, evening peak and peace peak, we use MAPE for comparison and evaluation of the stability of different prediction models. These performance indicators provide a deep understanding of the nature of prediction errors. The results as below.

(1) With using the data for the No. 261 bus in Guangzhou and No. 113 bus in Shenzhen, we verify the generalization performance of our model. The results showed that the proposed algorithm in this research had better performance for the whole routes than the DNN or the RNN for both the Guangzhou and Shenzhen datasets, as shown in Figure 7.

(2) We chose the peak period of the working day with a more complex traffic state and the flat peak period with a simple traffic state as the research period, and we compared the algorithms. It can be seen from Table 5 that for the maximum morning rush hour of the MAPE in Guangzhou, the error of the one-hour bus journey time was about 6.5 min. For the maximum evening peak of the MAPE in Shenzhen, the error of the one-hour bus journey time was about 5.6 min, which indicated that this model had good generalization ability, and it solved the problem that the proposed deep learning algorithms were suitable for the traffic states of different cities.

Table 5. Comparison of the results of the model for Guangzhou and Shenzhen.

MAPE / Time	Guangzhou (%)	Shenzhen (%)
Morning (08:00–09:00)	10.79	8.74
Evening (17:00–19:00)	9.11	9.33
Flat peak	8.17	7.97

5.3.3. Real-Time Bus Speed Information for Prediction

In order to get closer to the application scenario, our study divided their model into two scenarios in the prediction process. The two scenarios comprised a model based entirely on historical data and a

model based on bus real-time speed parameter correction. Our study use Model-Hist and Model-Real for the two scenarios.

The hyper parameters of these two different models were the same. The only difference was that in the model based on real-time sensor data correction, the speed of a real-time bus sensor was added to the model as a feature on the deep side. The results used only historical data and real-time bus speed data with historical data, as shown in Table 6.

Table 6. Comparative experiment for the historical data model and the real-time data model.

Error Indexes	Model-Hist	Model-Real
MAPE (%)	8.14	3.32
MAE	84.61	38.85
RMSE	108.9	51.49

After the addition of real-time bus speed data, the MAPE of the bus travel time forecast decreased by 4.82% compared with the historical data alone. This indicated that the MAPE value obviously decreased after considering the real-time speed of the bus, which in turn indicated that the real-time speed information of the bus had a great influence on the bus travel time prediction.

This was consistent with the view that the speed data of a taxi could reflect the traffic status [4], but it was also valuable to add the real-time speed of a bus to reflect the traffic status of bus lines. Because the bus often ran in the bus lane, the combination of the real-time speed of the bus, the historical speed of the taxi, and the historical speed of the bus could reflect the traffic status of the bus route more comprehensively and accurately. It also confirmed the work of Ma et al. (2019), who said that in their future work they would focus on using an existing taxi or another type of traffic data to estimate the newly designed or sparsely recorded bus travel time [4].

Figure 8 shows the data for about one week from 9:00 to 10:00 for the morning bus travel time of the site actual arrival time and the predicted travel time and the cumulative error figure. With the bus real-time speed in the model, the bus travel time gap between the predicted values and the real value was relatively small.

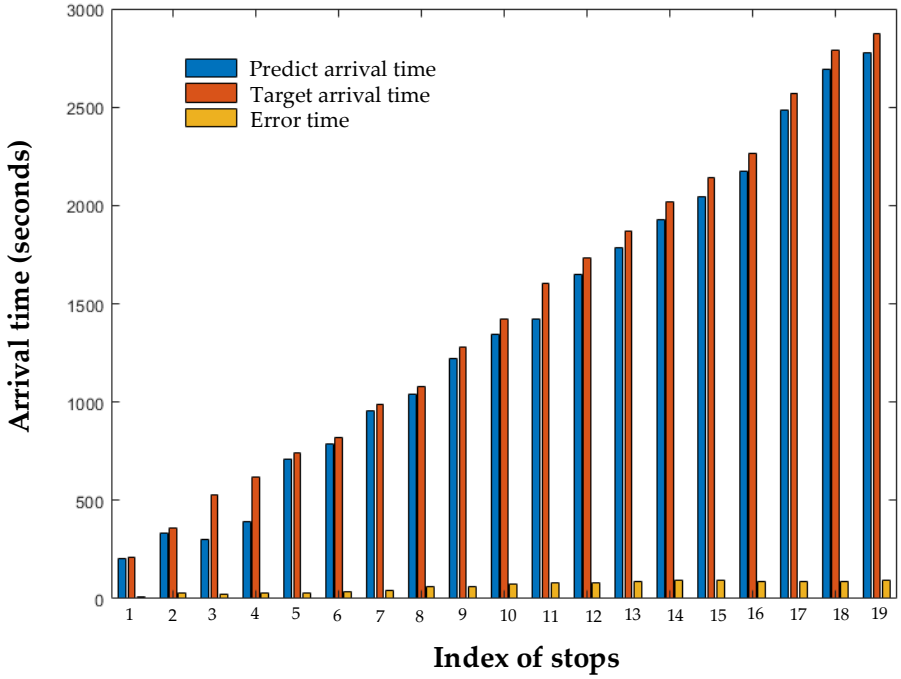

Figure 8. Actual/predicted arrival time and cumulative error diagram for each site.

5.3.4. Wide and Deep

In order to enable the model to capture as many differences between discrete and continuous features as possible, our study introduced the WD (Wide and Deep) model into the RNN model. With compared the influence of this module to the results for two different data sets in Guangzhou and Shenzhen, the results of the comparison are shown in Table 7.

Table 7. The impact of Wide and Deep.

Method	Guangzhou		Shenzhen with Real-Time Speed	
	Without W&D	**With W&D**	**Without W&D**	**With W&D**
MAPE (%)	8.81	8.43	3.42	3.32
MAE	87.27	86.31	40.16	38.85
RMSE	119.95	120.11	54.09	51.49

The model with the WD module was improved for the Guangzhou and Shenzhen data to different degrees, which proved that the discrete data and the continuous data played different roles in the model. The wide side could effectively memorize discrete features, while the deep side could effectively generalize continuous features.

5.3.5. Attention

The reason for using attention-based temporal and spatial architecture was that there was spatial-temporal correlation among the traffic variables that predicted the bus travel times. For the task of bus travel time prediction, our study thought that the spatial-temporal relationships of the data might have different influences on the prediction results. Therefore, in this study, the attention module was used to weight the spatial-temporal features. Under the condition of fixed hyper parameters, the effects of adding attention mechanism or removing the attention mechanism on the prediction results of the model were compared in this research.

As shown in Table 8, the results showed that the attention (Attn) model was helpful for improving the accuracy of the bus travel time prediction whether the data sets of Guangzhou or Shenzhen were used and whether the historical data was used alone or combined with the real-time bus speed.

Table 8. Influence of the attention mechanism on the prediction results for Guangzhou and Shenzhen.

Method	Guangzhou with Hist		Shenzhen with Real-Time Speed	
	Without Attn	**With Attn**	**Without Attn**	**With Attn**
MAPE (%)	8.53	8.43	3.42	3.32
MAE	87.67	86.31	39.33	38.85
RMSE	124.32	120.11	52.29	51.50

To verify the mechanism of attention, in this study, the weighted coefficient of attention for the bus's travel time features was visualized during a certain running state. Figure 9 shows the heat map of the spatial feature temporal features. In the visual feature map, the red areas represent higher response values, and the blue areas represent lower response values. By analyzing the attention scores learned by the attention model described in Section 4.1, our study were able to learn the view of the proposed method for the propagation mechanism of bus travel time prediction.

To further understand the propagation mechanism learned by the attention model used in the proposed method, the evolution of the attention scores was analyzed with respect to the impact on different bus stop IDs and the influence in the whole bus travel time prediction. Generally, it can be seen from Figure 9 that whether discrete (day, hour, stop ID) or continuous (v, dt, at, real-time speed, possible time) features were used, and whether temporal (v, dt, at, real-time speed, possible time, day,

hour) or spatial (stop ID) features were used; all of the features had different impacts on the prediction of bus travel times and bus stops.

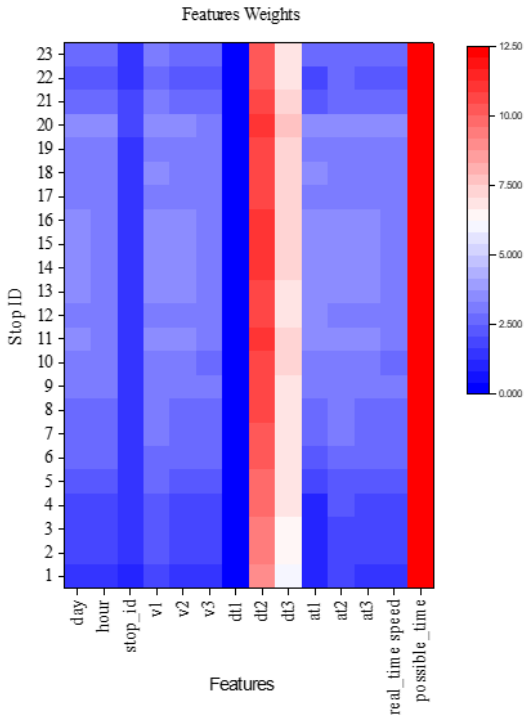

Figure 9. Average features weighting matrix.

As shown in Figure 9, our study could observe that temporal feature dt2 (average bus dwell time in 30 min at this point in the last week.) performed a rather important function in the model. This reflected the fact that there was an influence from the complex boarding mode, the bus dwell time was very unstable [4], and there were different basic modes, which had a significant impact on the total travel time. Additionally, dt2 had a different impact on different bus stops. In comparison, dt3 (average bus dwell time within 30 min on the same day of the last week) also had a moderate impact and dt1 (average bus dwell time within 30 min) had almost no impact on different bus stops.

Compared with previous studies, Ma et al. (2019) [4] did not forecast the dwelling time of each bus stop as a part of the total travel time of a bus [4], and Xu (2017) [23] and He et al. (2020) [6] did not use the full historical average bus dwelling times [6,24]. According to our heat map, shown in Figure 9, DT2 and DT3 have a greater weight on the prediction of bus travel time. DT2 is Average bus dwell time in 30 min at this point in the last week. DT3 is average bus dwell time within 30 min on the same day of the last week. It indicates that the bus travel time is influenced by the periodicity of dwelling time. Hence, it was a better decision to choose DT2 and DT3 simultaneously because the two features of bus dwelling time worked well in the prediction of bus travel time. Based on real-time information, it was important for the accuracy of real-time bus travel time prediction, especially the possible real-time transit times converted from real-time bus speeds. However, previous studies only considered the real-time bus speeds [23], rather than the possible real-time transit times. Furthermore, many research works of traffic prediction have emphasized the importance of spatial information [5,22]. The spatial feature of a bus stop ID had a certain impact on bus travel time prediction, and it had different influences on different bus stops. However, the impact is less prominent yet.

5.3.6. Hyper Parameters

In the experiment, the influences of different operation units of GRU and LSTM on the prediction results were compared for the study, as shown in Table 9.

Table 9. The influence of different hyper parameters on the model.

Error Indexes	Model-LSTM	Peepholes	Model-GRU
MAPE (%)	3.33	3.32	3.34
MAE	39.15	38.85	39.20
RMSE	52.05	51.50	52.27

Although the LSTM model was better than the GRU model, different computing units had little influence on the final prediction results of the model, which may have been because none of the computing units could capture the characteristics of migration between different states.

6. Conclusions

From the perspectives of time and space, the bus travel times of public transportation are dynamic/uncertain. The gap between a massive amount traffic data and its shallow features and the gap between full connection and rich features make it difficult to obtain representative features from datasets with rich features. The potential traffic state and traffic events belong to a hidden mode, so travel time prediction is a challenging problem of ITS. Therefore, it is of particular importance to develop a deep-seated architecture that fully reflects the characteristics of transit travel time.

We proposed an embedded network lever WD structure to solve the spatial data and designed an attention mechanism for the RNN to capture the temporal information. Finally, the system used the deep neural network model composed of the RNN and the DNN. The model could capture the non-static spatiotemporal correlation of the urban bus travel time. This enabled the model to generalize the learning model in the cross-temporal and spatial prediction. The model could be used to predict the dynamic travel times of buses. Its effect was better than those of the historical average method, traditional SVR model, SVR-Bayes optimization model, single DNN [5], and RNN [6,21], as shown in Figure 7. The main contributions of this study were as follows.

- Based on the prediction of bus travel time, the study proposed a new heterogeneous feature extraction framework based on the RNN model of embedding, WD, and an attention mechanism in order to gain a deep understanding of the space-time features and intrinsic connections of the characteristics related to bus travel time, and to visualize these features and connections, as shown in Tables 7 and 8, Figure 9.
- Fourteen historical spatial and temporal features were introduced, including the stop IDs as spatial features, bus dwelling times at different historical stages, and real-time GPS bus speeds with real-time possible transit times obtained based on real-time bus speeds as temporal features as shown in Table 2. Especially, the real-time bus speeds is important to improve the dynamic bus travel time prediction, which can be seen in Table 6. These features have not been studied together in previous studies
- The multiple super composition of the RNN and DNNs were carried out to reduce the residual heterogeneous data fusion and real-time dynamic bus travel time prediction. A new scheme for real-time dynamic bus travel time prediction was provided as shown in Figures 7 and 8.
- To verify the model's stability and generalization ability, the model was tested on the data sets of the Guangzhou 226 bus and the Shenzhen 113 bus. The buses ran on the main roads in big cities. Both of the experiments achieved good results. Few of the existing studies tested their models in different cities as shown in Figure 7 and Table 5.

For future work, our study will keep exploring the presented systems in the following directions. In addition to further improving the accuracy of the model, we will extend from one bus line to the bus lines of the entire road network. The existing models tested individual bus routes. The comparison can prove the validity of the model, but our study hold the point that more factors need to be considered. Therefore, it is a feasible choice to try to input the entire road network as a model. With the development of in-deep learning technology, this effect can be achieved through a deep image

convolution network of reference image processing [42], which is an important direction of our future research. The effect of missing data on the prediction is obvious. When the missing rate was more than 5%, the performance of the model decreased significantly when only speed was used as the input. When using the multi-attribute fusion, the model had good performance, not only when the error value was low but also when the error growth rate was low, particularly when compared with the model of Liu et al. (2018) [10]. In this research, the missing data were not considered thoroughly enough. In the future, more kinds of sensors (such as ground loops, videos, and geomagnetism) can be considered in order to repair the missing data and to further improve the accuracy.

Author Contributions: Wrote the manuscript, Y.Y. and C.S.; provided relevant information, discussed the data, and corrected the manuscript, Z.C., Z.H. and C.Z.; revised the manuscript, Y.Y., C.S., Z.C., Y.W. and V.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by National Natural Science Foundation of China (Grant No. 51678044), National Natural Science Foundation of China (Grant No. 52072025), Joint Funds of the National Natural Science Foundation of China (U1811463), National Natural Science Foundation Youth Fund (Y820631001). The study also had the support of the Guangdong Key Laboratory of Intelligent Transportation of Sun Yat-Sen University and the Shenzhen Transportation Committee.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Cao, Z.; Ceder, A. Autonomous shuttle bus service timetabling and vehicle scheduling using skip-stop tactic. *Transp. Res. Part C Emerg. Technol.* **2019**, *102*, 370–395. [CrossRef]
2. Cao, Z.; Ceder, A.; Zhang, S. Real-time schedule adjustments for autonomous public transport vehicles. *Transp. Res. Part C Emerg. Technol.* **2019**, *109*, 60–78. [CrossRef]
3. Yu, H.; Wu, Z.; Wang, S.; Wang, Y.; Ma, X. Spatiotemporal Recurrent Convolutional Networks for Traffic Prediction in Transportation Networks. *Sensors* **2017**, *17*, 1501. [CrossRef]
4. Ma, J.; Chan, J.; Ristanoski, G.; Rajasegarar, S.; Leckie, C. Bus travel time prediction with real-time traffic information. *Transp. Res. Part C Emerg. Technol.* **2019**, *105*, 536–549. [CrossRef]
5. Abdollahi, M.; Khaleghi, T.; Yang, K. An integrated feature learning approach using deep learning for travel time prediction. *Expert Syst. Appl.* **2020**, *139*, 112864. [CrossRef]
6. He, P.; Jiang, G.; Lam, S.-K.; Sun, Y. Learning heterogeneous traffic patterns for travel time prediction of bus journeys. *Inf. Sci.* **2020**, *512*, 1394–1406. [CrossRef]
7. Laña, I.; Del Ser, J.; Velez, M.; Vlahogianni, E. Road Traffic Forecasting: Recent Advances and New Challenges. *IEEE Intell. Transp. Syst. Mag.* **2018**, *10*, 93–109. [CrossRef]
8. Lv, Y.; Duan, Y.; Kang, W.; Li, Z.; Wang, F.-Y. Traffic Flow Prediction With Big Data: A Deep Learning Approach. *IEEE Trans. Intell. Transp. Syst.* **2014**, *16*, 1–9. [CrossRef]
9. Bengio, Y. *Learning Deep Architectures for AI*; Now Publishers Inc.: Boston, MA, USA, 2009.
10. Liu, Q.; Wang, B.; Zhu, Y. Short-Term Traffic Speed Forecasting Based on Attention Convolutional Neural Network for Arterials. *Comput. Civ. Infrastruct. Eng.* **2018**, *33*, 999–1016. [CrossRef]
11. Li, L.; Li, Y.; Li, Z. Efficient missing data imputing for traffic flow by considering temporal and spatial dependence. *Transp. Res. Part C: Emerg. Technol.* **2013**, *34*, 108–120. [CrossRef]
12. Tan, H.; Feng, J.; Feng, G.; Wang, W.; Zhang, Y.-J. Traffic Volume Data Outlier Recovery via Tensor Model. *Math. Probl. Eng.* **2013**, *2013*, 1–8. [CrossRef]
13. Wu, Y.; Tan, H.; Qin, L.; Ran, B.; Jiang, Z. A hybrid deep learning based traffic flow prediction method and its understanding. *Transp. Res. Part C Emerg. Technol.* **2018**, *90*, 166–180. [CrossRef]
14. Casas, N. Deep deterministic policy gradient for urban traffic light control. *arXiv* **2017**, arXiv:1703.09035.
15. El Hatri, C.; Boumhidi, J. Fuzzy deep learning based urban traffic incident detection. *Cogn. Syst. Res.* **2018**, *50*, 206–213. [CrossRef]
16. Gu, Y.; Lu, W.; Qin, L.; Li, M.; Shao, Z. Short-term prediction of lane-level traffic speeds: A fusion deep learning model. *Transp. Res. Part C Emerg. Technol.* **2019**, *106*, 1–16. [CrossRef]
17. Wang, Y.; Zhang, D.; Liu, Y.; Dai, B.; Lee, L.H. Enhancing transportation systems via deep learning: A survey. *Transp. Res. Part C Emerg. Technol.* **2019**, *99*, 144–163. [CrossRef]

18. Xu, C.; Ji, J.; Liu, P. The station-free sharing bike demand forecasting with a deep learning approach and large-scale datasets. *Transp. Res. Part C Emerg. Technol.* **2018**, *95*, 47–60. [CrossRef]
19. Tsoi, A.C.; Back, A. Discrete time recurrent neural network architectures: A unifying review. *Neurocomputing* **1997**, *15*, 183–223. [CrossRef]
20. Li, C.; Wang, J.; Ye, X. Using a Recurrent Neural Network and Restricted Boltzmann Machines for Malicious Traffic Detection. *NeuroQuantology* **2018**, *16*, 823–831. [CrossRef]
21. Duan, Y.; Lv, Y.; Wang, F.Y. Travel Time Prediction with LSTM Neural Network. In Proceedings of the IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), Rio de Janeiro, Brazil, 1–4 November 2016.
22. Petersen, N.C.; Rodrigues, F.; Pereira, F.C. Multi-output bus travel time prediction with convolutional LSTM neural network. *Expert Syst. Appl.* **2019**, *120*, 426–435. [CrossRef]
23. Xu, H.; Ying, J. Bus arrival time prediction with real-time and historic data. *Clust. Comput.* **2017**, *20*, 3099–3106. [CrossRef]
24. Liu, Y.; Liu, Z.; Jia, R. DeepPF: A deep learning based architecture for metro passenger flow prediction. *Transp. Res. Part C Emerg. Technol.* **2019**, *101*, 18–34. [CrossRef]
25. Cheng, H.T.; Koc, L.; Harmsen, J.; Shaked, T.; Shah, H. Wide & Deep Learning for Recommender Systems. In Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, Boston, MA, USA, 15 September 2016.
26. Liu, Y.; Wang, Y.; Yang, X.; Zhang, L. Short-term travel time prediction by deep learning: A comparison of different LSTM-DNN models. In Proceedings of the IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), Yokohama, Japan, 16–19 October 2017.
27. Oh, S.; Byon, Y.J.; Jang, K.; Yeo, H. Short-term travel-time prediction on highway: A review on model-based approach. *KSCE J. Civ. Eng.* **2017**, *22*, 298–310. [CrossRef]
28. Cheng, Z.; Chow, M.Y.; Jung, D.; Jeon, J. A big data based deep learning approach for vehicle speed prediction. In Proceedings of the IEEE 26th International Symposium on Industrial Electronics (ISIE), Edinburgh, UK, 19–21 June 2017.
29. Ma, Z.; Koutsopoulos, H.N.; Ferreira, L.; Mesbah, M. Estimation of trip travel time distribution using a generalized Markov chain approach. *Transp. Res. Part C Emerg. Technol.* **2017**, *74*, 1–21. [CrossRef]
30. Kumar, B.A.; Vanajakshi, L.; Subramanian, S. Pattern-based bus travel time prediction under heterogeneous traffc conditions. In Proceedings of the 93rd Annual Meeting—Transportation Research Record, Washington, DC, USA, 12–16 January 2014.
31. Watkins, K.E.; Ferris, B.; Borning, A.; Rutherford, G.S.; Layton, D. Where Is My Bus? Impact of mobile real-time information on the perceived and actual wait time of transit riders. *Transp. Res. Part A Policy Pr.* **2011**, *45*, 839–848. [CrossRef]
32. Chien, S.I.; Ding, Y.; Wei, C. Dynamic Bus Arrival Time Prediction with Artificial Neural Networks. *J. Transp. Eng.* **2002**, *128*, 429–438. [CrossRef]
33. Rahman, M.; Wirasinghe, S.; Kattan, L. Analysis of bus travel time distributions for varying horizons and real-time applications. *Transp. Res. Part C Emerg. Technol.* **2018**, *86*, 453–466. [CrossRef]
34. Yang, M.; Chen, C.; Wang, L.; Yan, X.; Zhou, L. Bus arrival time prediction using support vector machine with genetic algorithm. *Neural Netw. World* **2016**, *26*, 205–217. [CrossRef]
35. Brakewood, C.; Macfarlane, G.S.; Watkins, K. The impact of real-time information on bus ridership in New York City. *Transp. Res. Part C Emerg. Technol.* **2015**, *53*, 59–75. [CrossRef]
36. Xinghao, S.; Jing, T.; Guojun, C.; QiChong, S. Predicting bus real-time travel time basing on both GPS and RFID data. In Proceedings of the 13th COTA International Conference of Transportation Professionals (CICTP), Shenzhen, China, 13–16 August 2013.
37. He, J.; Shen, W.; Divakaruni, P.; Wynter, L.; Lawrence, R. Improving traffic prediction with tweet semantics. In Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, 3–9 August 2013.
38. Shalaby, A.; Farhan, A. Prediction Model of Bus Arrival and Departure Times Using AVL and APC Data. *J. Public Transp.* **2004**, *7*, 41–61. [CrossRef]
39. Zhou, M.; Wang, D.; Li, Q.; Yue, Y.; Tu, W.; Cao, R. Impacts of weather on public transport ridership: Results from mining data from different sources. *Transp. Res. Part C Emerg. Technol.* **2017**, *75*, 17–29. [CrossRef]

40. Julio, N.; Giesen, R.; Lizana, P. Real-time prediction of bus travel speeds using traffic shockwaves and machine learning algorithms. *Res. Transp. Econ.* **2016**, *59*, 250–257. [CrossRef]
41. Hans, E.; Chiabaut, N.; Leclercq, L.; Bertini, R.L. Real-time bus route state forecasting using particle filter and mesoscopic modeling. *Transp. Res. Part C Emerg. Technol.* **2015**, *61*, 121–140. [CrossRef]
42. Zhou, Y.; Yao, L.; Chen, Y.; Gong, Y.; Lai, J. Bus arrival time calculation model based on smart card data. *Transp. Res. Part C Emerg. Technol.* **2017**, *74*, 81–96. [CrossRef]

electronics

MDPI

Article

Reputation System for Increased Engagement in Public Transport Oriented-Applications

David García-Retuerta [1,*], Alberto Rivas [1], Joan Guisado-Gámez [2], Eleni Antoniou [3], Pablo Chamoso [1]

[1] BISITE Research Group, University of Salamanca, Calle Espejo 2, 37007 Salamanca, Spain; rivis@usal.es (A.R.); chamoso@usal.es (P.C.)
[2] Data Management Group, Universitat Politècnica de Catalunya, Jordi Girona 1-3, 08034 Barcelona, Spain; joan@ac.upc.edu
[3] AETHON Engineering Consultants, 25 Em. Benaki Str, 10678 Athens, Greece; e.antoniou@aethon.gr
* Correspondence: dvid@usal.es

Abstract: Increasing user engagement is one of the biggest challenges when a new application is developed. An engaged user is one who finds a product valuable; highly engaged users generate profit. This study focuses on increasing user engagement in a transport application, via a user reputation score feature. The score is to reward application users and activity organisers, as well as to motivate beginners by offering a high reputation score in the first days of use. The algorithms are based on exponential and logarithmic functions, and were first tested on synthetic data. Real-world tests have shown that the algorithms behave as expected, but the COVID-19 pandemic created a disturbance which prevented any user from achieving the maximum score and many users from registering altogether. Data show positive results, although the real number of users is not sufficient to certify a correct behaviour. Further tests will be carried out when transport activities return to normal.

Keywords: reputation algorithm; users' reputation; transport; software application

Citation: García-Retuerta, D.; Rivas, A.; Guisado-Gámez, J.; Antoniou, E.; Chamoso, P. Reputation System for Increased Engagement in Public Transport Oriented-Applications. *Electronics* **2021**, *10*, 1070. https://doi.org/10.3390/electronics10091070

Academic Editor: Claus Pahl

Received: 30 March 2021
Accepted: 28 April 2021
Published: 30 April 2021

1. Introduction

The vast amount of data generated on the Internet can be converted into highly valuable information if a proper analysis is carried out. Analysing and filtering the information is especially necessary in cases where the user can interact directly with the content offered in the service. Analysis mechanisms, like those applied in recommender systems, are capable of extracting knowledge in systems that manage large volumes of information. This type of system ensures a satisfactory user experience by providing users with the content they are looking for. New innovative solutions have been proposed in recent years to improve urban transport. Mobility services such as bike-sharing, car-sharing, intermodal public transport and the concept of "Mobility as a Service" (MaaS) are effectively shifting demand away from private vehicles [1]. Moreover, smartphone penetration rates are increasing all over the world, facilitating iteration with public transport users via an application. Applications can become an important element of a city, improving citizens' experience and increasing the quality of tourism [2]. As a result, the development of a new app can provide new functionalities and enhancements to a city's infrastructure.

In recent years, mobile apps with user-generated content have become highly popular (TripAdvisor, Amazon, BlaBlaCar, ResearchGate, etc). The trust-building mechanisms of these apps have been enhanced so that a stranger on the internet can be seen as a "trustman" [3], based on the ideal "in truth we trust". Therefore, these apps expand the source of trustworthy information from a few acquaintances to the whole app community, which is of great value to users [4].

Smartphones have the ability to assist users with the completion of tasks (utilitarian), to entertain them (hedonic) and to connect them with others (social) [5]. These three

incentives can boost user engagement in a mobile app. Furthermore, a good balance between short-term rewards and medium-term rewards must be found, so that a gradual engagement is achieved. Otherwise, if the user does not perceive an increase in value or perceives a high initial value but no lasting value, then app engagement will lower considerably [6].

When users perceive high value and user engagement is high, the app community can grow on the principles of the gift economy—where valuables are not sold, but given, without an explicit agreement for immediate or future rewards. Webpages like Wikipedia have proven this concept to be highly effective and successful among the internet community [7].

The My-TRAC application has been developed to provide new public-transport-oriented functionalities. Its value lies in presenting practical alternatives to the use of private vehicles by enabling citizens to make better use of public transport. The application creates a healthy user community and a trustworthy source of information.

The main objective of this article is to propose a reputation algorithm to facilitate recommendations on a series of trip-related activities, such as the purchase of tickets, selection of the most appropriate means of transport, tourist activities, etc., which the users will be able to use as a guide while planning their trips.

This work is organised as follows: a review of existing reputation systems is presented in Section 2. Section 3 describes the proposal. Section 4 presents the assessment made with synthetic data and the pilot data. Finally, Section 5 presents the conclusions.

2. Background

Advisory systems provide advice and help solve problems that are normally solved by human experts [8]. In any community, individuals whose opinion is considered more important are normally trusted more. The knowledge of human experts can then be extracted and coded to automatise the process. Reputation systems are a kind of advisory system that allow users to rate each other in online communities so as to build trust through reputation [9].

Numerous proposals for reputation algorithms have been put forward over the years. They are generally quite context-dependent. This is because each problem entails the study of the best solution and, in most cases, it is not enough to have one generic proposal or to apply a specific proposal to a different problem. It is always necessary to adapt the approach to the new problem. From the analysis of the state of the art, it can be inferred that context-dependent solutions generally perform better than those that do not consider the context [10]. Additionally, some platforms have been designed to ease the development of such systems, as an essential part of any developing Smart City [11,12]

The subsections that follow present different existing reputation systems can be divided into groups of academic and commercial proposals.

2.1. Academic Proposals

Among the scientific proposals in the state of the art, two of them stand out (PageRank and EigenTrust). PageRank is the most popular of all the reputation algorithms, presented in [13] and used previously by Google to order the websites in its search engine in an objective and mechanical way. Four years later, researchers from Stanford University proposed an algorithm for reputation management in peer-to-peer (P2P) networks, called EigenTrust and described in [14]. With its application, they managed to minimize the impact of malicious peers on the performance of a P2P system.

PathTrust [10], has been presented more recently. It is based on a model that exploits the graph of relationships among the participants of virtual organizations. Its authors indicate that the system is based on the two previous algorithms (PageRank and EigenTrust); however, they are not directly applicable because their personalization is very limited.

Below is a brief description of how each algorithm works, along with its advantages and disadvantages.

- PageRank [13]: **Advantages**: This algorithm converges in about 45 iterations. Its scaling factor is roughly linear in $\log(n)$. It uses graph theory to link the pages. An important component of PageRank is that its calculation can be personalized. PageRank can estimate web traffic and can predict backlinks. **Disadvantages**: PageRank is based on random walks on graphs. This algorithm assumes the behaviour of a "random surfer", but if a real Web surfer ever gets into small loops of web pages, the PageRank will have false positives. This method of random surfer assumes that periodically the surfer "gets bored" and jumps to another random page;
- EigenTrust [14]: **Advantages**: This reputation system is among the most well-known and successful reputation systems. It satisfactorily solves different problems existing in P2P systems, which is the context in which the algorithm was designed. **Disadvantages**: The main drawback of this system is its reliance on a set of pre-trusted peers, which causes nodes to centre around them. As a consequence, other peers are ranked low despite being honest, marginalizing their role in the system [15];
- PathTrust [10]: **Advantages**: This model of reputation (using the trust relationships amongst the participants) is resistant against the attack of faking positive feedback. A group of attackers collaborates to boost their reputation rating by leaving false, positive feedback for each other. In this model of reputation, this will only strengthen the trust relationship among the attackers, but will not necessarily strengthen the path of the attacker to an honest inquirer, such that their reputation does not affect the honest inquirer. Another benefit of exploiting established relationships in member selection is the formation of long-term relationships. **Disadvantages**: The trust relationship between two participants is formed on the basis of past experience with each other. A participant leaves a feedback rating after each transaction, and these ratings are accumulated to a relationship value. Therefore, one user can boost a positive or negative feedback.

2.2. Commercial Proposals

Currently, the most important reputation system proposals are those used by commercial applications. Generally, commercial reputation systems directly focus on assigning users a reputation score within that commercial system (for example, the reputation systems of TripAdvisor, Waze, Amazon and BlaBlaCar).

The conclusion drawn from the review of the state of the art is that all the existing reputation system proposals, especially those of commercial systems, focus exclusively on their context. This implies that a specific algorithm has to be designed to obtain good results. To do this, it is essential to identify the factors and the extent to which they have a direct influence on reputation.

In the same way, although each type of parameter has its weight, which defines its impact on the final score, each occurrence of the parameter may affect the associated factors differently. It is, therefore, necessary to determine how the score assigned to each occurrence of a parameter evolves over time.

Moreover, in the majority of the analysed commercial proposals, the user must know the highest possible reputation level that can be reached in the system. This allows them to understand the relevance of the different scores.

3. Proposal for User and Users' Choices Reputation Algorithm

My-TRAC is an app devoted to the research and development of user-centric services that enhance the passengers' multimodal door-to-door experience. This helps citizens develop greater confidence in, and adhesion to, multimodal transport services. Furthermore, My-TRAC improves adaptation to the users' needs through the provided data, statistics and trends from the passengers' experiences while using the proposed platform. An example of the user interface can be found in Figure 1.

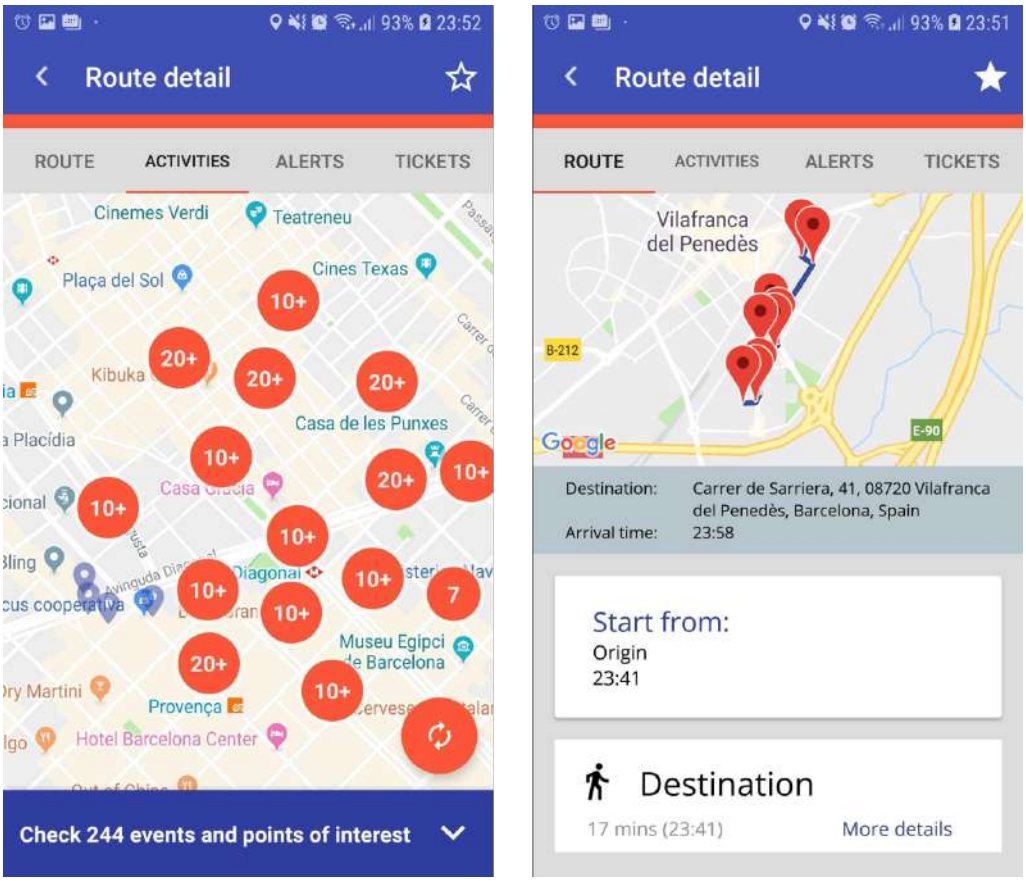

Figure 1. The user interface of the My-TRAC app.

This section describes the algorithms used on My-TRAC to assign a reputation score to each user and each user choice, activity or Point Of Interest (POI), representing their ranking within the system. The two algorithms share a common basis; however, each is used for a different purpose: one calculates the users' reputation and the other one calculates the reputation of the choices made by users. Therefore, each algorithm uses different factors and metrics. As a result, each subsection describes either the part dedicated to the users' reputation algorithm or the users' choices' reputation algorithm.

The proposed model is based on a mixture of exponential and logarithmic functions to create a system of distributed trust, a idea not yet fully explored in the literature. For example, the most common research lines base their mapping functions on the definitions of clever distances [16], graph analysis [13] or the definition of a set of rules affecting to the trust relationships among users [10]. The main advantage of the current proposal is that the mapping functions can be easily adapted or extrapolated to new systems, by just analysing the importance of the considered features and selecting a suitable function for those parameters, resulting in a higher versatility than other works.

This section is structured as follows: the factors identified as essential to determine a user's reputation in the system are presented below. Then, the metrics associated with each of the individual factors are shown, followed by the description of the mechanism that provides the initial score, which is the output of both algorithms. Finally, the proposed adaptive weight mechanisms of both algorithms are described. They adapt the weight of the factors according to the dynamic characteristics of the application where the algorithms are applied. Thus, the role of this mechanism is to re-establish the limits of each factor over time, as the number of users or the number of existing ratings changes with time, providing an adequate maximum score.

3.1. Mathematical Description of the Reputation System

The reputation system is based on a combination of logarithmic and exponential functions to map the inputs onto their corresponding reputation. Metrics are related to each of the identified factors. Therefore, each metric determines the reputation score provided by its corresponding factor and each factor has its own metric. Besides, metrics affect the overall reputation, as it is calculated as the sum of the scores of all the factors. Each metric provides a final score which is calculated as the percentage reached by that user over the total weight of each factor, and these final scores are added to obtain the user/activity reputation.

The number of instances required to reach the maximum score is established for each factor. In addition, the slope of any of the parameter functions determines how fast or how slowly the value for that parameter increases. In this case, the *slope* parameter refers to the steepness, incline, or grade of the function. It has been established that the evolution is not linear, just like ResearchGate's calculation of its "RG Score". Thus, the growth in the score of a specific factor will either be logarithmic or exponential, following Equations (1) and (2).

$$scoreParameter_i = y_{maximum} \times \frac{\log(slope \times \frac{x}{x_{maximum}} + 1)}{\log(2 + slope)} \quad (1)$$

The logarithmic equation shown in Equation (1) is useful in cases where the slope should be greater in the initial instances and then gradually decrease in subsequent instances. For example, to encourage new users to rate activities, the first few ratings the user gives will have a considerable effect on their reputation, however, the user will not be able to continue gaining reputation at the same rhythm after producing a considerable amount of ratings. Instead, further ratings will have a smaller impact on the reputation of the user. Logarithmic growth is regulated by the $slope$ variable of the equation, whereas the maximum number of instances is regulated by $x_{maximum}$. This factor will be dynamic due to the usage characteristics of the social network. Therefore, in the case of ratings provided by users, the maximum score $x_{maximum}$ can take a value of 200, meaning that a user with more than 200 ratings will obtain a 100% initial score, which will greatly contribute to the final score. In cases where the usage patterns of the application imply that users give a large number of ratings, the factor $x_{maximum}$ is adjusted dynamically, so that $x_{maximum} = 2 \times avgRatingsByUser$. Finally, the factor $y_{maximum}$ can reach 1, so that each factor will have a score between 0 and 1.

$$scoreParameter_j = y_{maximum} \times (\frac{x}{x_{maximum}})^{slope} \quad (2)$$

The exponential equation shown in Equation (2) is useful for factors in which the weight of the initial instances is lesser and becomes more important in the system as the number of instances grows. For example, a user that opens the application three times does not notice a significant increase in their reputation in the system; however, a user who opens the application 200 times is considered a regular user, and therefore obtains a pertinent reputation.

Although the mathematical approach described above is not directly based on any existing work to determine reputation, these types of equation are well known and widely used in the literature for multiple purposes.

Mathematically speaking, the most similar proposed work can be found in [17], where the authors present a trust management system based on reputation mechanisms. The mechanisms proposed in this paper base the evolution of reputation on the number of assessments that follow a logarithmic distribution.

3.1.1. User Reputation Mathematical Model

User reputation is calculated using a mixture of exponential and logarithmic functions. These are selected in order to maximise user engagement, providing them with fast rewards for some easy tasks (logarithmic growth) and slow rewards until they complete a challenging task (exponential growth).

All the equations related to the users can be found in Table 1.

Table 1. Equations related to the user reputation, inputs, outputs and factors.

Factor	Metrics (Equations)	Inputs	Outputs
Days registered	$s_1 = w_{date} \times \frac{c_{date} - r_{date}}{M_{date}}$	c_{date}, r_{date}	s_1
List of valuations	$s_2 = w_{valuations} \times \frac{\log(100 \times \frac{n_{valuations}}{M_{valuations}} + 1)}{\log(2+100)}$	$n_{valuations}$	s_2
Number of uses of the application	$s_3 = w_{uses} \times \left(\frac{n_{uses}}{M_{uses}}\right)^2$	n_{uses}	s_3
Number of chosen routes	$s_4 = w_{routes} \times \left(\frac{n_{routes}}{M_{routes}}\right)^2$	n_{routes}	s_4

The following variables are used as input:

- c_{date} refers to the current date.
- r_{date} refers to the registration date.
- $n_{valuations}$ refers to the number of valuations of the user.
- n_{uses} refers to the number of times the user opened the app.
- n_{routes} refers to the number of routes the user has chosen to travel.

The following variables are the obtained outputs:

- s_1 is the initial score of days registered.
- s_2 is the initial score of the list of valuations.
- s_3 is the initial score of the number of uses of the application.
- s_4 is the initial score of number of chosen routes.

M represents the number of occurrences of a given parameter to provide the maximum value/weight that it is capable of providing over the total reputation (w).They refer to maximum and weight, respectively. Both are static (but editable) variables obtained from the database. The subscript indicates to which factor they are related.

The final score S is defined as shown in Equation (3):

$$S = \sum_{i=1}^{4} s_i \tag{3}$$

The pseudocode of this procedure can be found below in Algorithm 1.

Algorithm 1 User Reputation Calculation.

```
def f_log(x, x_max, slope=100):
    return log(slope × (x/x_max)+1)/log(2+slope)
def f_exp(x, x_max, slope=2):
    return pow((x/x_max), slope)
reputation = f_log(User.valuations, Maximum.valuations) × Weight.valuations
  + f_exp(User.uses_app, Maximum.uses_app) × Weight.uses_app
  + f_log(User.tickets_purchased, Maximum.tickets_purchased) ×
  Weight.tickets_purchased + (User.days_registered/Maximum.registration) ×
  Weight.registration + f_log(User.groups, Maximum.groups) × Weight.groups
if reputation > 100 then
    reputation = 100
else if reputation < 0 then
    reputation = 0
```

This novel proposal could provide commercial systems with the following advantages: (i) dynamic adaptation of the reputation to the information of the system (non-linear growth), (ii) dynamic adaptation at parameter level, varying the specific weight that each parameter has on the final reputation, (iii) engaging the users with well-modelled changes in its reputation score.

To this end, mechanisms similar to those used in well-known proposals, that have been proven to work well (such as the one presented in [17]), have been integrated with the peculiarities of My-TRAC, which determine the information to be used.

3.1.2. Users' Choices Reputation Mathematical Model

The users' choices (activities and POIs) reputation are calculated using a mixture of linear and logarithmic functions. They are selected in order to maximise users' engagement, providing the users' choices with fast rewards initially, and then basing the rewards on the average star rating received.

All the equations related to users' choices can be found in Table 2.

Table 2. Equations related to the activities' reputation, inputs, outputs and factors.

Factor	Metrics (Equations)	Inputs	Outputs
n-star ratings weighted average	$s_1 = w_{rating} \times \sum_{i=1}^{n} \left(\frac{\sum_{k=1}^{n} (re_{user_k} \times ra_{user_{k,i}})}{\sum_{k=1}^{n} re_{user_k}} \times \frac{1}{M_{rating}} \right)^2$	re_{user}, ra_{user}	s_1
Number of views of the activity	$s_2 = w_{views} \times \frac{\log(50 \times \frac{n_{views}}{n_{days} \times M_{views}} + 1)}{\log(2+50)}$	n_{days}, n_{views}	s_2

The following variables are used as input:

- $ra_{user_{k,i}}$ is the rating of the k-th user on the i-th activity. The number of users who rated this activity is defined as n.
- re_{user_k} is the reputation of the k-th user who rated the activity. The number of users who rated this activity is defined as n.
- n_{days} is the number of days since the activity was created.
- n_{views} is the number of views of the activity.

The following variables are the obtained outputs:

- s_1 is the initial score of N-star ratings weighted average.
- s_2 is the initial score of the number of views of the activity.

M and w are static (but editable) variables obtained from the database. They refer to maximum and weight, respectively. The subscript indicates which factor they are related to.

The final score, S, is defined as shown in Equation (4):

$$S = \sum_{j=1}^{2} s_j \tag{4}$$

The pseudocode of this procedure can be found below in Algorithm 2.

3.2. Updating the Parameters and Their Weights

The information on My-TRAC is not static; instead, it evolves over time. This obliges the metrics that are part of the reputation algorithms to adapt to the information. For this reason, it is crucial to implement mechanisms that update configurable factors in each of the metrics.

For example, during the pilot stage, when the application begins to obtain real user data, the system will start from zero. In the beginning, a lower number of instances of each factor will be required to obtain a significant final reputation score of a user/activity. The number of instances required will be much higher after a year of system functioning.

Algorithm 2 Activity Reputation Calculation.

```
def f_log(x, x_max, slope=100):
    return log(slope × (x/x_max)+1)/log(2+slope)
def f_exp(x, x_max, slope=2):
    return pow((x/x_max), slope
numerator = 0
denominator = 0
for activity in ratings_database do:
    numerator += activity.rating × activity.reputation_user
    denominator += activity.reputation_user
avg_rating_reputation ← numerator / denominator
reputation = f_log(Activity.views, days_registered × Maximum.views, 50) × Weight.views + f_exp(avg_rating_reputation, Maximum.average_rating) × Weight.average_rating
if reputation > 100 then
    reputation = 100
else if reputation < 0 then
    reputation = 0
```

According to the number of instances of each one of those factors, the metrics that make up the algorithm can be adapted and the values can be updated automatically or manually. Both the weight that the parameter has on the final reputation and number of occurrences that a parameter must have to obtain the maximum score can be updated.

Regarding the weight of the parameter in the final reputation, it is set *a priori* but can be changed at any given point in order to correct certain anomalies or to encourage desired behaviours. On the other hand, there are two ways of updating the number of occurrences that a parameter must have to obtain its maximum score:

- Manually: when an expert administrator/developer decides that it should be changed for some reason.
- Automatically: depending on the evolution of the information on the platform. For example, the rating an activity has in the system will not remain the same; it is going to change over time and, according to its evolution, the maximum weight of this parameter in the system can increase or decrease (if it receives many ratings, its weight will decrease).

In the first version of the model, the system's automatic adaptation has not been evaluated because the data we are using at this stage are not sufficient to test it effectively.

4. Evaluation and Results

The evaluation of the proposed algorithms has been tested using two complementary methods: creation of synthetic data and deployment of a pilot program. Synthetic data are meant to simulate the behaviour of users when the app has gained popularity and is already established, and the pilot phase provides a clear picture of how the algorithms will behave in the beginning of the application deployment phase.

The only way of evaluating the correct functioning of the algorithm with the synthetic data is the following: to analyse whether the obtained output behaves as expected and then draw conclusions as to whether the reputation score assigned to different users corresponds to the initial idea, as a function of the values of each of the parameters affecting the reputation score.

Due to the initial lack of available data of real users, synthetic data were generated in order to evaluate the proposed method. A total of 2000 simulated users were randomly generated considering the following attributes:

1. Gender of users (i.e., male, female).
2. Age group: Three age groups were considered (i.e., 16–30, 31–50, 51–90).

3. Volume of performed actions. Two possibilities: users that performs a small number of actions and users that performs many actions.
4. Type or category of the performed actions. Six major categories of actions were defined for the generated dataset: sports, eating, history, dancing, cinema, shopping.

Considering the above attributes, the generated dataset contained information about the demographics of the users and the number of actions performed for each category. The generation of a synthetic user is carried out by the data generator, which randomly chooses the gender, the age group, and the volume of actions and based on the number of actions performed by the mean users of the category that the user is applied to. The generator randomly calculates (based on a uniform distribution) the number of actions of the generated user for each of the six types of action.

Pilot data were used to analyse the real-world behaviour of the models, in a initial phase. As a result, it is expected that many users register but do not make any usage of the app. As the functionalities of the app are still limited, user engagement is likely to be lower than in the real application.

The evaluation of the obtained results is a subjective task; however, it is important to verify that the algorithms behave as expected. Section 4.1 describes the tests carried out related to the artificially generated users and their results. Section 4.2 describes the real-world experiment and its results.

4.1. Reputation Models Evaluation—Synthetic Data

When creating the synthetic dataset, the aim is to simulate the behaviour of real users and users' choices in the most realistic possible way. This method will provide an *a priori* idea of how well the system works.

4.1.1. Users' Reputation Evaluation

Evaluation methodology. This simulation aims to model the use of the system by users. Therefore, no inactive users will be generated, even though, in a real system, they could become the majority.

In this way, there will be a set of users who use the system a lot, a larger set who use it frequently and an even larger one who use it sporadically. This has involved the creation of three ranges of usage possibilities when creating the data.

This distribution of users is easily observed by analysing the scoreboard of the commercial applications that made their scoreboard public. For example, on Waze [18], one of the tools analysed in Section 2, a user with 100,000 points can reach the maximum level "Waze Royalty", which means they are among the 1% most active users in the country, while the top users listed on the scoreboard have more than one million points.

Results. Figure 2 shows the distribution of reputation among system users. On the x-axis, there are reputation intervals, and on the y-axis, the number of users with a reputation within those intervals.

The resulting scores present a Gaussian distribution which denotes a desirable behaviour—this is the distribution that would be expected from many natural phenomena.

4.1.2. Users' Choices Reputation Evaluation

Evaluation methodology. On the other hand, the users' choices reputation algorithm, which determines the reputation of the activities and POIs included on My-TRAC, has also been evaluated using synthetic data.

In this case, the only case-specific restrictions that have been applied when generating the synthetic dataset are:

- The identifier is a unique integer from 1 to 1000.
- The inclusion date is between 1 September 2017 (start of the project) and 18 December 2018 (the date on which the evaluation was carried out).
- The number of views of an activity is higher than its number of ratings.

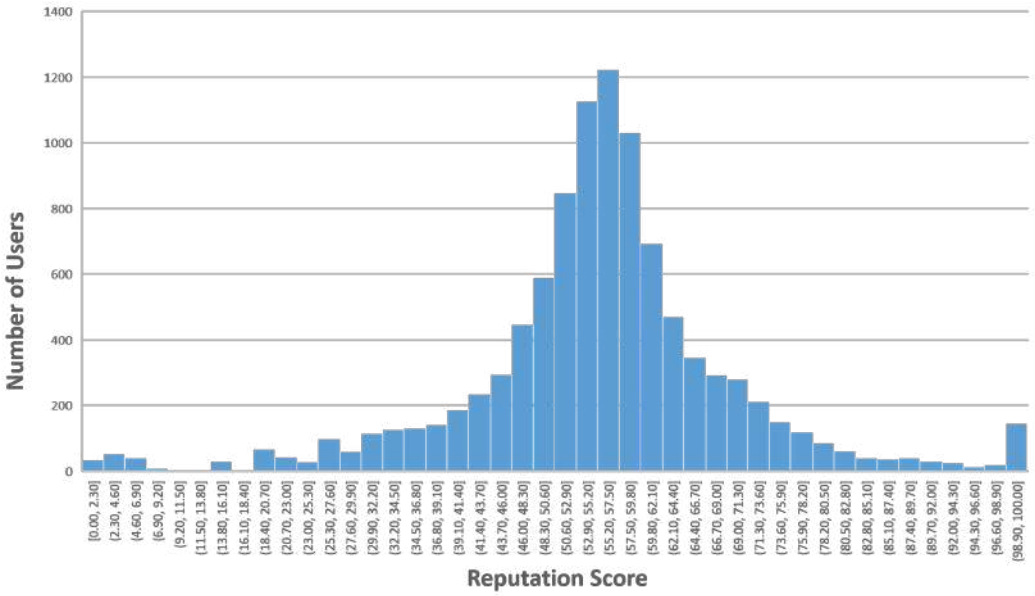

Figure 2. Representation of the results obtained with synthetic data for the evaluation of the user reputation model.

Results. The distribution of the reputation of the 1000 synthetically created activities is shown in Figure 3, which shows, on the x-axis, the reputation values of the activities and on the axis, and the number of activities that there are in the different reputation intervals.

It can be observed that there is no activity with a reputation of less than 21, because the synthetic data were created to test the performance of the models with active users and successful activities. These circumstances are not expected to exist in reality, where it is expected that there may be activities that receive no ratings at all during the pilot stage.

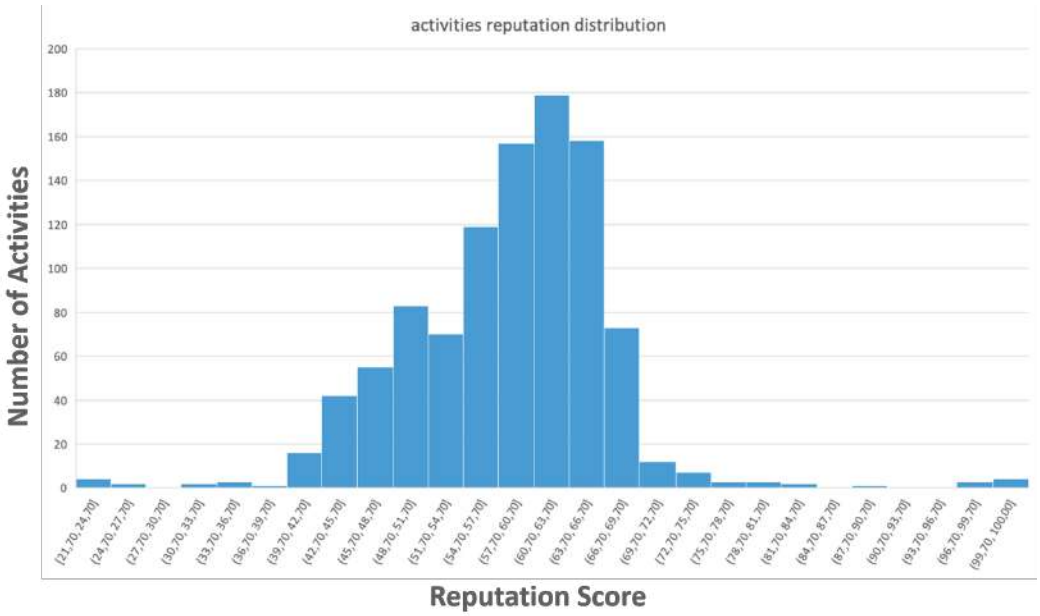

Figure 3. Distribution of the activities' reputation with the generated synthetic data

4.2. Reputation Models Evaluation—Pilot Study

Evaluation methodology. The previously designed reputation models have been evaluated in the pilot phase. A variation in the initial model has been designed and its joint use with the Social Market (another functionality of My-TRAC) is proposed. The Social Market is a means of encouraging use of the application, as it enables the users to exchange

the points they have obtained for rewards. The system allows the user to earn free tickets as a reward, in exchange for a set number of points. The number of obtained points is directly related to the user's reputation. It is designed to encourage the user to make more frequent use of the application.

It is necessary to remember that there are reputation models for both users and activities. However, a specific variation in the user reputation model has been designed for the current phase and integrated in the Social Market.

Thus, the version of the reputation model that has undergone major evaluation and been tested by the users in the pilot phase is the original proposed model, with a slight variation. What is different is that the date on which the users register does not affect their reputation.

In the initial version of the model, a very active user who has been registered for a few days would have a greater reputation than a user who has been registered for much longer and who has also used the features of the application (used it sometimes, for example). There are two main reasons for designing a variant for the pilot model:

1. The duration of the pilots is the same for all users and if a user has used the application more times than another user, they should get a higher reward, independently of the date of registration.
2. If the date has a negative effect on the user's reputation, i.e., the more time passes, the less reputation the user will have if they do not participate. This would cause the user's points on the Social Market to decrease even though the user has not spent them. This is an undesirable situation for the evaluation of the model.

Therefore, the score obtained by the users in this phase is a decimal value between 0 and 100, where 0 is the initial reputation value for a user who has just registered, and 100 points can be reached by carrying out repeated interactions with the application. For example, every time an activity or POI is valued, a certain reputation value is assigned according to the previously defined metrics.

These points can be redeemed at the Social Market, where each user's points will be updated periodically at 0:00 (CET) each day. The points on the Social Market have been updated periodically to control possible fraudulent behaviour by users who create multiple accounts, automate actions and obtain rewards illegally at the time. In this way, the development team can act as a moderator if this type of behaviour is detected and proceed accordingly, for example, by deleting the user's account for non-compliance with the terms and conditions of use.

However, although the score that users have been able to visualize throughout the pilot phase is the score that is provided with the user reputation version created for integration with the Social Market, this section of the document also presents the results that would have been obtained with the reputation version not linked to the reward points and the users' choice reputation models version. Thanks to this, it is possible to check how the models operate in the presence of real data, although, after carrying out the evaluation, it can be anticipated that the volume of information that has been collected is again insufficient.

Results. Due to the pandemic, strict mobility restrictions have been implemented, affecting the information that have been collected; this is different from the information we would have expected under normal circumstances.

More specifically, there are 171 valid users out of a total of 206 (which means that 35 decided to delete their account). It can be seen in the results presented below that not all of them have interacted with the tool. This was expected, as it commonly happens in any type of application, as some users download the application and register but never use it.

The pilot was open to everyone who wanted to register, and an advertising campaign was carried out in The Netherlands, Athens (Greece) and Barcelona (Spain) to encourage participation.

The results and evaluation of each of the data models are presented below.

4.2.1. User Points (Social Market Version)

The results of the adapted version of the model for the Social Market reward points calculation, are presented below. They were obtained after carrying out the pilots with different graphs incorporated in the panel of the analysis tool mentioned above.

Figure 4 shows the distribution of the points allocated for the total number of users (206), i.e., both active and non-active users, grouped by ranges of 10 units.

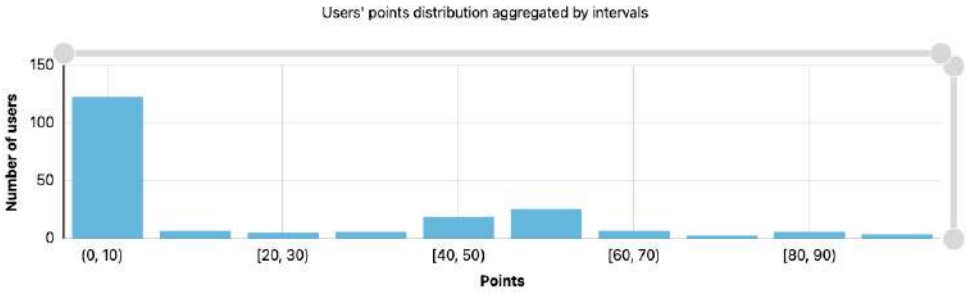

Figure 4. Reward points for all users (206) points grouped ranges of 10 units.

It can be seen that there is a set that encompasses the majority of users (123), and this distorts the results. This is due to the fact that the majority of users have not interacted with the application at all or hardly at all.

To analyse this situation in greater detail, Figure 5 shows the same type of graph as the prior one, but, in this case, the groupings of points are made by unit rather than in groups of 10.

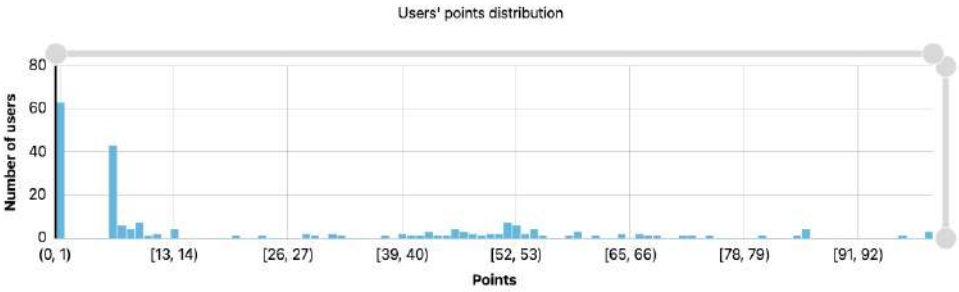

Figure 5. Reward points for all users (206) points grouped by units.

It can be seen that there are 63 users with the minimum value of reputation, which implies that they have registered and have not carried out any more activities, while there are 43 users who have obtained the score that corresponds to a one-time use of the application.

Let us consider the users who have not interacted in any way with the application as non-active users, thus providing 143 active users, and proceed to analyse the results again. Figure 6 again shows a graph with the distribution of users according to their points grouped in ranges of 10.

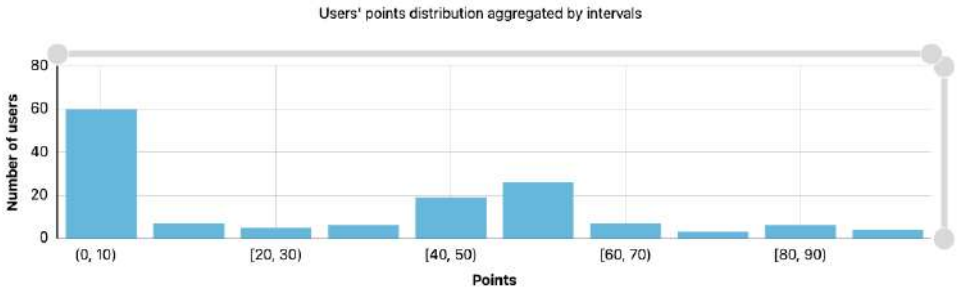

Figure 6. Reward points for active users (143) points grouped in ranges of 10 units.

As with all users, the group of very inactive users still stands out, as they almost have not interacted with the tool, so if we filter the graph by leaving out the first range of values (from 0 to 10), we obtain a graph that is a better representation of the behaviour of the "average" users of the application, as shown in Figure 7.

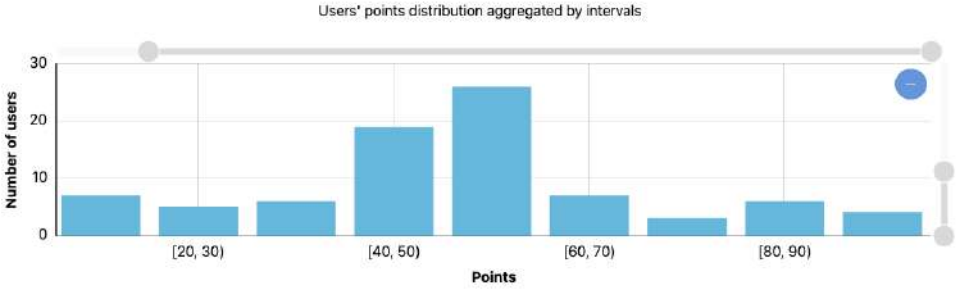

Figure 7. Reward points for active users (143) filtered and grouped in ranges of 10 units.

As mentioned above, the number of users who have participated in the pilots was not significant enough to draw relevant conclusions regarding the functioning of the reputation models; however, a very similar behaviour to the one expected can be observed, which was obtained by generating synthetic data following a series of criteria intended to represent the real behaviour of users. The expected results are represented in the document by the graph shown in Figure 2.

It can be seen that the pursued objective has been achieved: the users who initially participate add points to their reputation score with relative ease until they reach the average values. It becomes more difficult for a user to go above the average reputation values, motivating users to continue to use the app to increase their score, thus increasing their loyalty.

However, a certain number of users were expected to have the highest score and this was not achieved, possibly because users have not been able to travel as much as expected due to the restrictions caused by the COVID-19 pandemic and because 100% of the app's functionality is still not available.

An analysis of user activities was carried out, which provided points to better understand the type of activity carried out by the users of the app. For example, Figure 8 shows the points awarded to users according to the number of times they have used the app.

It can be seen that 67.5% of the users obtained a reward of between 0 and 1 points for using the app, i.e., they were less active, while 20.4% of the users obtained between 9 and 10 points (the maximum) for using the app.

A similar analysis can be made for the score given to users depending on the number of times they have requested a route and followed it. This analysis is shown in Figure 9.

In this case, it can be seen that 80.3% of users were awarded between 0 and 3 points for following suggested routes, while only 2.4% of users obtained between 27 and 30 points (the maximum) for having followed suggested routes. This clearly shows that very few users used this functionality (40 to be exact).

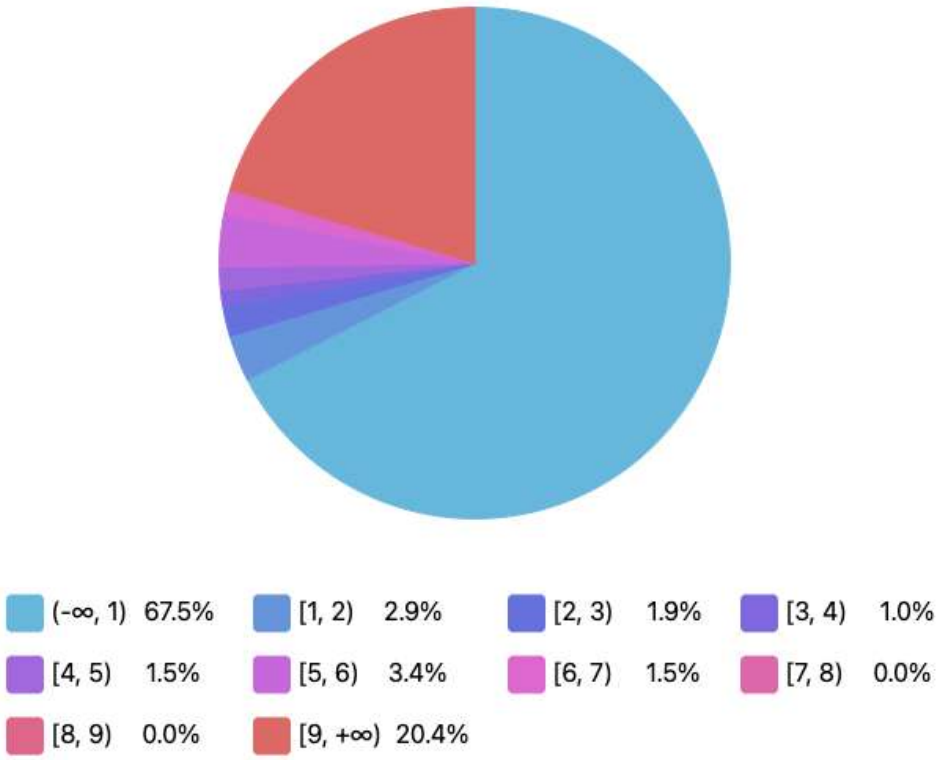

Figure 8. Users (% and size of sectors) who have obtained a certain amount of reward points (colour) depending on how many times they have opened the app.

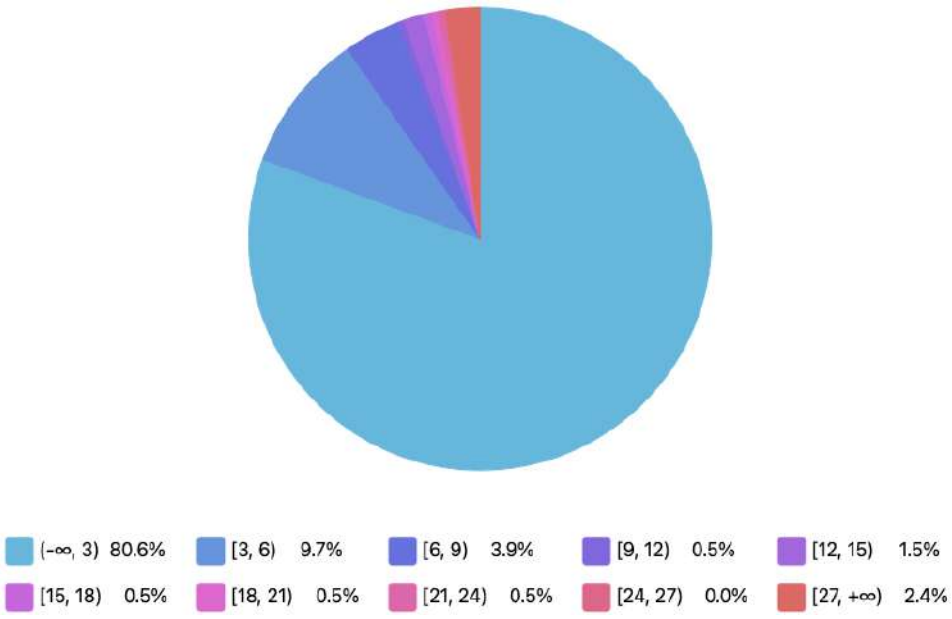

Figure 9. Users (% and size of sectors) who have obtained a certain amount of reward points (colour) depending on how many times they followed a suggested route.

4.2.2. Users' Reputation

Although the first version of the User Reputation Model was not used for the reasons outlined above, it is possible to carry out an assessment to demonstrate how the system would have performed.

In this case, out of the 206 total real-world live users, no one had a reputation of 100, because active users stopped being active before the date of the assessment, and this negatively affected the maintenance of their score at the highest value. The maximum reputation in this case was 86, achieved by two users. To represent this, 10 groupings with equal ranges were created, which are shown in Figure 10.

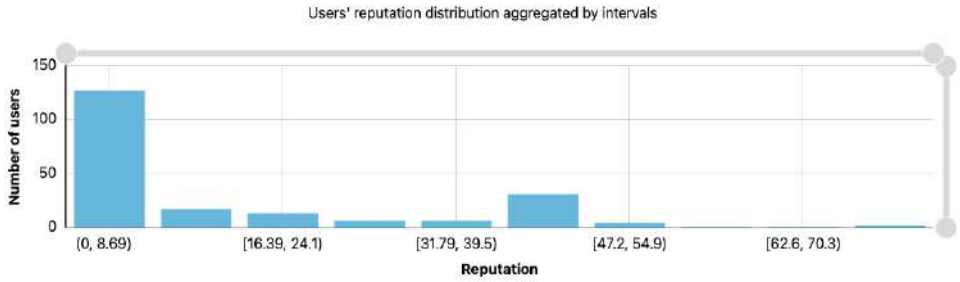

Figure 10. Reputation values for all users (206), scores grouped in 10 ranges.

The distribution is not exactly the same as with the reward points, but, in the same way, the majority continues to remain in low values, mainly due to inactivity, so the results are evaluated by discarding this set of users and focusing again on the 143 real-world live users, who have at least interacted with the app. The distribution of their reputation is shown in Figure 11.

As participation has been lower than expected due to mobility restrictions, it can be seen that the majority of users have a below-average reputation, although the group with the highest number of users is in the intermediate reputation zone, as expected.

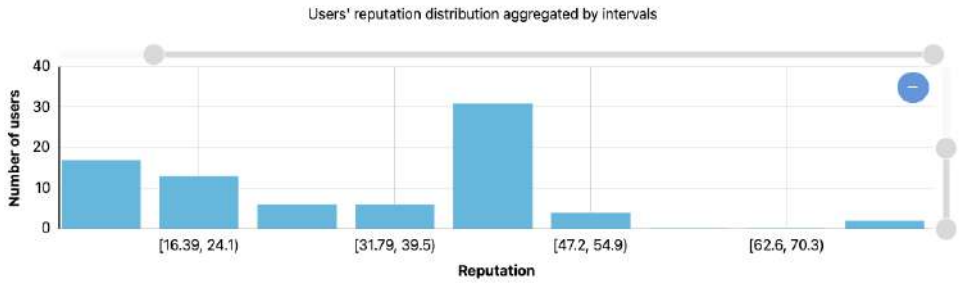

Figure 11. Reputation values for active users (143), scores grouped in 9 ranges.

4.2.3. POIs and Activities' Reputation

As far as the reputation of POIs and activities is concerned, the evaluation that can be made on the basis of the information obtained from the pilots would not truly reflect a real scenario, since the interaction of the real-world live users with this functionality on My-TRAC has not been sufficient. The vast majority of POIs and activities have not been interacted with, so they have no reputation, as can be seen in Figure 12.

If the results are evaluated, leaving aside the activities and POIs that have not been interacted with, the results shown in Figures 13 and 14 are obtained.

Figure 13 shows that users only interacted with two activities, for which they have an average reputation, while Figure 14 shows that users have interacted with a total of 37 POIs.

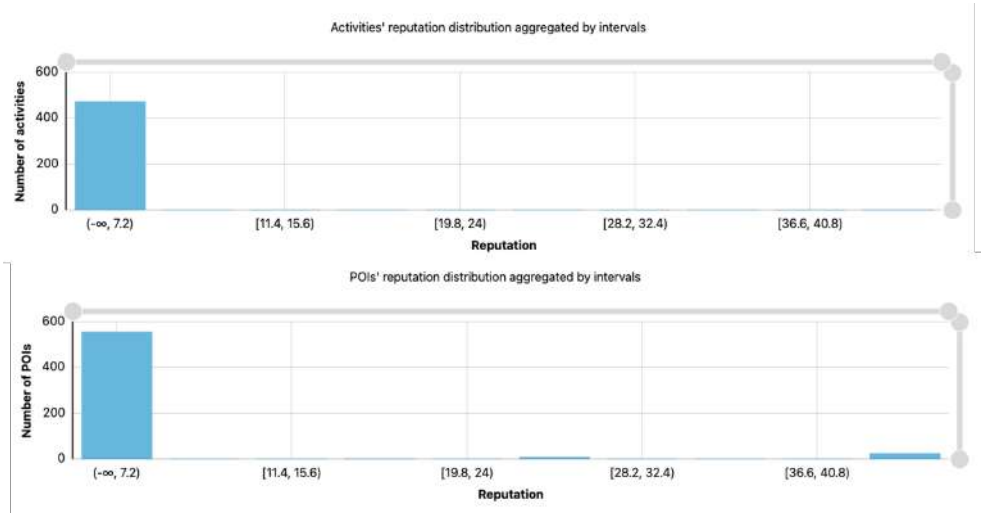

Figure 12. Reputation values for all the activities and all the POIs

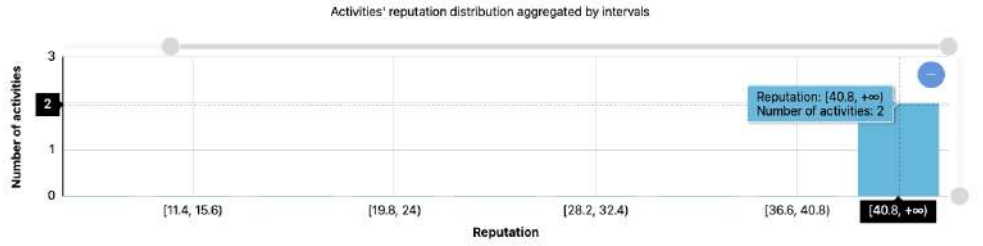

Figure 13. Reputation values for the activities with reputation value > 0.

On the one hand, we can conclude that users interact with POIs more than with the activities offered by the app, despite the fact that there is an even number of options (473 activities and 556 POIs). On the other hand, it can be concluded that users are, in most cases, satisfied with the POIs they visit, as 25 of the 37 POIs they have interacted with have high reputations.

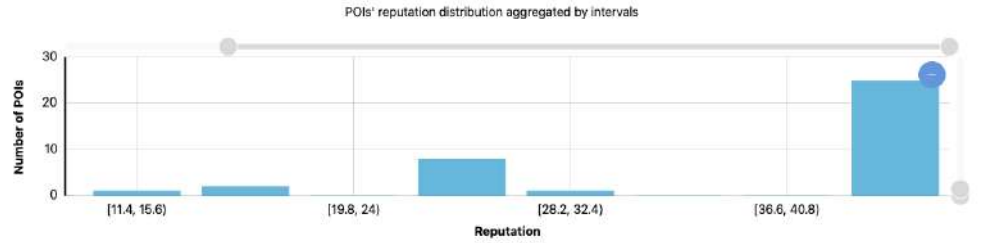

Figure 14. Reputation values for the POIs with reputation value > 0.

5. Conclusions

Following analysis of the results, the conclusion is that, although the results seems to follow the value distribution patterns that were sought with the initially defined models, the number of active users is still not sufficient to certify that, in a real scenario, it will behave as expected.

However, using the data obtained from the pilots and the simulations, the obtained results were satisfactory, as no unexpected behaviours were detected. Moreover, it is clear that the algorithm encourages users to participate more actively by giving them points rapidly, and that reaching the maximum score is such a difficult task that users need to be engaged before achieving it.

The reputation scores seem to form a normal or Gaussian distribution, with peaks on the higher or lower end, resulting from optimal user behaviour in the synthetic data and

from a low participation in the pilots, respectively. In general, active real-world users tend to cluster around the reputation value 50 (the maximum reputation value is 100), which is a desirable result. It does not demotivate users by maintaining their low score, and does not cause them to become bored by giving them the maximum score often. Most users will have around 50 points (out of 100), creating healthy competition against similar users, as they try to surpass their equals and not to be left behind.

Activities and POIs also take advantage of having the same basis; therefore, analogous results are obtained and a similar purpose is fulfilled.

It can, therefore, be concluded that, even though there was not enough data, the goal of allowing users to determine the relevance of users and the actions was fulfilled in the case study conducted on the My-TRAC platform.

This research and its results can be taken advantage of by any user who needs to develop a similar system and apply it in a real-world scenario. For example, a new video platform could adapt the developed basic functions (logarithmic and exponential) to assign a reputation to the content creator and content consumers.

The main limitations of this work are related to the limited data gathered during the pilots phase, adversely affected by the effects of the COVID-19 pandemic on mobility. Moreover, user engagement is measured through the distribution of the reputation scores: an indirect measurement instead of a direct one.

Regarding future research on this topic, user engagement will be measured when the application is launched. The parameters' limits will be updated in order to obtain a Gaussian distribution shape, with a moderate number of users obtaining the maximum score. If the resulting distribution has several peaks or is chaotic in any sense, more input will be used to obtain a better modelling of the users' worth.

Author Contributions: Conceptualization, P.C. and A.R.; methodology, D.G.-R.; software, D.G.-R.; validation, J.G.-G., E.A. and P.C.; formal analysis, J.G.-G., E.A. and P.C.; investigation, D.G.-R.; resources, J.G.-G., E.A. and P.C.; data curation, J.G.-G., E.A. and P.C.; writing—original draft preparation, D.G.-R.; writing—review and editing, D.G.-R.; visualization, P.C.; supervision, J.G.-G., E.A. and P.C.; project administration, J.G.-G., E.A. and P.C.; funding acquisition, P.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research has been supported by the European Union's Horizon 2020 research and innovation program under grant agreement No 777640.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kamargianni, M.; Li, W.; Matyas, M.; Schäfer, A. A critical review of new mobility services for urban transport. *Transp. Res. Procedia* **2016**, *14*, 3294–3303. [CrossRef]
2. Dickinson, J.E.; Ghali, K.; Cherrett, T.; Speed, C.; Davies, N.; Norgate, S. Tourism and the smartphone app: Capabilities, emerging practice and scope in the travel domain. *Curr. Issues Tour.* **2014**, *17*, 84–101. [CrossRef]
3. Chun, B.N.; Bavier, A. Decentralized trust management and accountability in federated systems. In Proceedings of the 37th Annual Hawaii International Conference on System Sciences, Big Island, HI, USA, 5–8 January 2004; p. 9.
4. Li, N.; Mitchell, J.C.; Winsborough, W.H. Design of a role-based trust-management framework. In Proceedings of the Proceedings 2002 IEEE Symposium on Security and Privacy, Berkeley, CA, USA, 12–15 May 2002; pp. 114–130.
5. Kim, Y.H.; Kim, D.J.; Wachter, K. A study of mobile user engagement (MoEN): Engagement motivations, perceived value, satisfaction, and continued engagement intention. *Decis. Support Syst.* **2013**, *56*, 361–370. [CrossRef]
6. Lalmas, M.; O'Brien, H.; Yom-Tov, E. Measuring user engagement. *Synth. Lect. Inf. Concepts Retr. Serv.* **2014**, *6*, 1–132. [CrossRef]
7. Bruns, A. *Blogs, Wikipedia, Second Life, and beyond: From Production to Produsage*; Peter Lang: New York, NY, USA, 2008; Volume 45.
8. Forslund, G. Toward cooperative advice-giving systems: A case study in knowledge-based decision support. *IEEE Expert* **1995**, *10*, 56–62. [CrossRef]
9. Josang, A.; Ismail, R. The beta reputation system. In Proceedings of the 15th Bled Electronic Commerce Conference, Bled, Slovenia, 17–19 June 2002; Volume 5, pp. 2502–2511.
10. Kerschbaum, F.; Haller, J.; Karabulut, Y.; Robinson, P. Pathtrust: A trust-based reputation service for virtual organization formation. In Proceedings of the International Conference on Trust Management, Pisa, Italy, 16–19 May 2006; Springer: Berlin/Heidelberg, Germany, 2006; pp. 193–205.

11. Corchado, J.M.; Chamoso, P.; Hernández, G.; Gutierrez, A.S.R.; Camacho, A.R.; González-Briones, A.; Pinto-Santos, F.; Goyenechea, E.; Garcia-Retuerta, D.; Alonso-Miguel, M.; et al. Deepint. net: A Rapid Deployment Platform for Smart Territories. *Sensors* **2021**, *21*, 236. [CrossRef] [PubMed]
12. Garcia-Retuerta, D.; Chamoso, P.; Hernández, G.; Guzmán, A.S.R.; Yigitcanlar, T.; Corchado, J.M. An Efficient Management Platform for Developing Smart Cities: Solution for Real-Time and Future Crowd Detection. *Electronics* **2021**, *10*, 765. [CrossRef]
13. Page, L.; Brin, S.; Motwani, R.; Winograd, T. *The PageRank Citation Ranking: Bringing Order to the Web*; Technical Report; Stanford InfoLab: Stanford, CA, USA, 1999.
14. Kamvar, S.D.; Schlosser, M.T.; Garcia-Molina, H. The eigentrust algorithm for reputation management in p2p networks. In Proceedings of the 12th international conference on World Wide Web, Budapest, Hungary, 20–24 May 2003; pp. 640–651.
15. Kurdi, H.A. HonestPeer: An enhanced EigenTrust algorithm for reputation management in P2P systems. *J. King Saud Univ. Comput. Inf. Sci.* **2015**, *27*, 315–322. [CrossRef]
16. Adler, B.T.; De Alfaro, L. A content-driven reputation system for the Wikipedia. In Proceedings of the 16th International Conference on World Wide Web, Banff, AB, Canada, 8–12 May 2007; pp. 261–270.
17. Zacharia, G.; Maes, P. Trust management through reputation mechanisms. *Appl. Artif. Intell.* **2000**, *14*, 881–907. [CrossRef]
18. Waze. Your Rank and Points—Connected Citizens Program. 2019. Available online: https://wiki.waze.com/wiki/Your_Rank_and_Points (accessed on 12 January 2019).