High performance analysis for road traffic control

Sara Moukir

université PARIS-SACLAY

High performance analysis for road traffic control

Analyse haute performance pour le contrôle de trafic routier

Thèse de doctorat de l'Université Paris-Saclay

École doctorale n° 580 : Sciences et Technologies de l'Information et de la Communication (STIC)
Spécialité de doctorat : Informatique et Mathématiques
Graduate School : Informatique et sciences du numérique
Référent : Université Versailles Saint-Quentin-en-Yvelines

Thèse préparée dans les unités de recherche **Li-Parad** (Université Paris-Saclay, UVSQ) et **Maison de la Simulation** (Université Paris-Saclay, UVSQ, Inria, CNRS, CEA), sous la direction de **Nahid EMAD**, Professeure, et du co-encadrant de **Loïc DORBEC**, Directeur chez Eiffage Energie Systèmes

Thèse soutenue à Paris-Saclay, le 26 août 2024, par

Sara MOUKIR

Composition du jury

Membres du jury avec voix délibérative

Edouard AUDIT Directeur de Recherche, Université Paris Saclay, France	Président
Taisuke BOKU Professeur, University of Tsukuba, Japon	Rapporteur & Examinateur
Ewa DEELMAN Professeure, University of South California, USA	Rapporteuse & Examinatrice
Marc BUI Professeur, Université Paris 8 et Ecole Pratique des Hautes Etudes	Examinateur
Alex FENDER PhD, Sr. Engineering Manager, NVIDIA, USA	Examinateur

THESE DE DOCTORAT

NNT : 2024UPASG039

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Analyse haute performance pour le contrôle de trafic routier
Mots clés : Analyse de données massives, Simulation de trafic routier, Intelligence artificielle distribuée et systèmes multi-agents, Système dynamique complexe et hétérogène, Programmation parallèle et distribuée multi-niveaux, Calcul haute performance

Résumé : La réduction des temps de trajet et de la consommation d'énergie dans les réseaux routiers urbains est cruciale pour le bien-être collectif et la durabilité environnementale. Depuis les années 1950, la modélisation du trafic a été un axe central de la recherche. Avec l'évolution des capacités informatiques, des simulations sophistiquées représentant fidèlement les complexités du trafic routier ont émergé, essentielles pour évaluer les technologies sans perturber le trafic réel. Les systèmes de transport deviennent plus complexes avec des informations en temps réel, nécessitant des modèles de simulation adaptés. Les simulations multi-agents, analysant les comportements individuels dans un environnement dynamique, sont particulièrement efficaces pour cette tâche, permettant de comprendre et de gérer le trafic urbain en représentant les interactions entre les voyageurs et leur environnement. Simuler de grandes populations de voyageurs dans les villes a longtemps été une tâche exigeante en termes de ressources informatiques. Les technologies avancées permettant la distribution des calculs sur plusieurs ordinateurs ont ouvert de nouvelles possibilités. Cependant, de nombreux simulateurs de mobilité urbaine n'exploitent pas pleinement ces architectures distribuées, limitant leur capacité à modéliser des scénarios complexes. L'objectif principal de cette recherche est d'améliorer la performance algorithmique et computationnelle des simulateurs de mobilité. Nous développons et validons des modèles de distribution génériques et reproductibles pouvant être adoptés par divers simulateurs de mobilité multi-agents, surmontant ainsi les barrières techniques pour analyser les systèmes de transport complexes dans des environnements urbains dynamiques. Nous utilisons le simulateur de trafic MATSim, reconnu pour la simulation de trafic multi-agents, pour tester nos méthodes génériques. Notre première contribution applique l'approche "Unite and Conquer" à MATSim. Cette méthode accélère les simulations en exploitant les architectures informatiques modernes. L'approche multiMATSim réplique plusieurs instances de MATSim sur plusieurs nœuds de calcul avec des communications périodiques, chaque instance fonctionnant sur un nœud séparé, utilisant les capacités de multithreading de MATSim pour améliorer le parallélisme. La synchronisation périodique assure la cohérence des données, tandis que les mécanismes de tolérance aux pannes permettent à la simulation de se poursuivre même en cas d'échec de certaines instances. Cette approche optimise l'utilisation des ressources informatiques selon les capacités spécifiques de chaque nœud. La deuxième contribution explore les techniques d'intelligence artificielle pour accélérer l'obtention des résultats de la simulation. Nous utilisons des réseaux de neurones profonds pour prédire les résultats des simulations MATSim. Les réseaux de neurones sont entraînés sur des données de simulations précédentes pour prédire les différentes sorties de la simulation. Les résultats sont comparés à ceux de MATSim pour évaluer leur précision. En résumé, nos contributions fournissent de nouvelles variantes algorithmiques et explorent l'intégration du calcul haute performance et de l'IA dans les simulateurs de trafic multi-agents. Nous démontrons l'impact de ces modèles et technologies sur la simulation de trafic, en abordant les défis et les limites de leur mise en œuvre. Notre travail met en évidence les avantages des architectures émergentes et des nouveaux concepts algorithmiques pour améliorer la robustesse et la performance des simulateurs de trafic, avec des résultats prometteurs.

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Title : High performance analysis for road traffic control
Keywords : Big data analysis, Road traffic simulation, Distributed artificial intelligence and Multi-agent systems, Complex and heterogeneous dynamic system, Multi-level parallel and distributed programming, High-performance computing

Abstract : The need to reduce travel times and energy consumption in urban road networks is critical for improving collective well-being and environmental sustainability. Since the 1950s, traffic modeling has been a central research focus. With the rapid evolution of computing capabilities in the 21st century, sophisticated digital simulations have emerged, accurately depicting road traffic complexities. Mobility simulations are essential for assessing emerging technologies like cooperative systems and dynamic GPS navigation without disrupting real traffic. As transport systems become more complex with real-time information, simulation models must adapt. Multi-agent simulations, which analyze individual behaviors within a dynamic environment, are particularly suited for this task. These simulations help understand and manage urban traffic by representing interactions between travelers and their environment. Simulating large populations of travelers in cities, potentially millions of individuals, has historically been computationally demanding. Advanced computer technologies allowing distributed calculations across multiple computers have opened new possibilities. However, many urban mobility simulators do not fully exploit these distributed architectures, limiting their ability to model complex scenarios involving many travelers and extensive networks. The main objective of this research is to improve the algorithmic and computational performance of mobility simulators. We aim to develop and validate generic and reproducible distribution models that can be adopted by various multi-agent mobility simulators. This approach seeks to overcome technical barriers and provide a solid foundation for analyzing complex transport systems in dynamic urban environments. Our research leverages the MATSim traffic simulator due to its flexibility and open structure. MATSim is widely recognized in the literature for multi-agent traffic simulation, making it an ideal candidate to test our generic methods. Our first contribution applies the "Unite and Conquer" approach to MATSim. This method accelerates simulation speed by leveraging modern computing architectures. The multiMATSim approach involves replicating several MATSim instances across multiple computing nodes with periodic communications. Each instance runs on a separate node, utilizing MATSim's native multithreading capabilities to enhance parallelism. Periodic synchronization ensures data consistency, while fault tolerance mechanisms allow the simulation to continue smoothly even if some instances fail. This approach efficiently uses diverse computational resources based on each node's specific capabilities. The second contribution leverages artificial intelligence to accelerate the process of obtaining simulation results. Specifically, we use deep neural networks to predict MATSim simulation outcomes. Neural networks are trained on data from previous simulations to predict simulation's outputs. The outputs are compared to MATSim results to assess accuracy. In summary, our contributions provide new algorithmic variants and explore integrating high-performance computing and AI into multi-agent traffic simulators. We aim to demonstrate the impact of these models and technologies on traffic simulation, addressing the challenges and limitations of their implementation. Our work highlights the benefits of emerging architectures and new algorithmic concepts for enhancing the robustness and performance of traffic simulators, presenting promising results.

Dedication and Thanks

My thoughts first turn to my parents. To my father, who instilled in me the values of self-improvement, integrity, and hard work. Your example has always been a source of inspiration, pushing me to give my best. Thank you for being such an exceptional man. To my mother, for all the love, patience, rigor, and discipline she has shown. You have always encouraged me to follow my passions and believe in myself.

Thanks to your values and all the support you have given me, I am where I am today. I am deeply grateful to have had you as role models. I dedicate this work to you with endless gratitude. May you be proud of me and rest in peace.

To my big sisters and my wonderful nephews, this work is also dedicated to you. Your support and daily encouragement are invaluable. You have been a constant source of motivation and comfort. Thank you for being there for me.

To Professor Nahid Emad, who offered me this invaluable opportunity, guided this thesis with wisdom, and accompanied me with kindness throughout its duration. Thank you for all the knowledge you have imparted to me, your patience, and your unwavering support. I am honored to have worked under your direction and to have benefited from your experience. Thank you for this exceptional experience.

To Jean-Michel Batto, my mentor and friend, who helped me grow both professionally and scientifically. It is also thanks to you that I had the chance to complete this thesis. Your support, your wise advice, and your friendship have been invaluable. I can never thank you enough for all you have done for me.

To my circle of friends, for their encouragement during difficult times and their support, and for always believing in me throughout this journey.

Finally, to all those who contributed in one way or another to this journey, I express my deepest gratitude.

Contents

Chapter 1

Introduction

1.1 Motivations

The need to reduce travel times and energy consumption in urban road networks has become a major issue for improving collective well-being and environmental sustainability. Initiated in the 1950s with methods based on fluid mechanics, traffic modeling has long been at the heart of research in this field. With the advent of the 21st century, the rapid evolution of computing capabilities has paved the way for the detailed representation of multiple simple phenomena, allowing their assembly into sophisticated digital simulations to accurately describe the complexities of road traffic.

The advancement of mobility simulations proves essential in various applications, particularly facilitating the assessment of the impact of emerging technologies such as cooperative systems, the adoption of new mobility services like carpooling, and the use of dynamic GPS navigation systems. The main asset of these simulations lies in their ability to exhaustively model a variety of scenarios without disrupting real traffic flow.

However, as transport systems become enriched with a multitude of connected

entities and access to real-time information becomes widespread, simulation models must adapt to this new complexity. The predictability of behaviors within modern transport networks thus becomes increasingly difficult. Faced with this reality, multi-agent simulations, centered on the analysis of individual behaviors within a partially perceived and constantly changing environment, emerge as an appropriate approach for the design of advanced simulation systems. These simulations facilitate the faithful representation of interactions between travelers and their environment, playing an essential role in understanding and improving the management of urban traffic.

1.2 Problem Statement

Simulating real populations of travelers in a large city, potentially numbering several million individuals, has long been seen as an excessively demanding task in terms of computational resources, bordering on impossibility until recently. Nevertheless, the advent of advanced computer technologies, allowing the distribution of calculations across a vast network of computers, has marked a turning point, opening the door to previously unexplored possibilities. Despite these advances, a large number of current urban mobility simulators do not fully exploit these distributed architectures, which limits their ability to model scenarios involving a large number of travelers, modes of transport, and networks of considerable size.

This limitation directly impacts the ability of researchers and urban planners to predict the consequences of regulatory policies and information dissemination strategies in extensive networks, especially in contexts where travelers are connected and receive information in real time. Faced with this challenge, the main objective of this research is to contribute to improve algorithms and computational performance of these mobility simulators. In this end, we strive to develop and

validate generic and reproducible distribution models that could be adopted by a variety of multi-agent mobility simulators, or more broadly, by any agent-based simulation system. This approach aims to overcome current technical barriers and provide a solid foundation for analyzing complex transport systems in dense and dynamic urban environments.

1.3 Contributions

Within the context of this thesis, we have made two main contributions by leveraging the MATSim traffic simulator[31]. The choice of MATSim as the foundational platform is motivated by its flexibility and open structure, making it suitable for the integration of new approaches. MATSim is recognized for its applicability to a variety of mobility simulation contexts, rendering it an appropriate candidate to test the effectiveness of generic methods in the domain of traffic simulation.

The first contribution is based on the application of a technique for accelerating the convergence of iterative methods, called 'Unite and Conquer' (UC) [22] and used in high-performance linear algebra, to the MATSim simulator. The important characteristics of UC (such as intrinsic multi-level parallelism, heterogeneity, fault tolerance etc.) are all elements that bring their share of potential performance improvements to the adaptation of the approach in our case. This method aims to enhance the speed of simulations by leveraging the capabilities offered by modern computing architectures. This research approach emphasizes a high-performance-oriented design to optimize the temporal efficiency of simulations, particularly for scenarios involving extensive urban networks.

The second contribution delves into the application of artificial intelligence techniques, primarily, to expedite the simulation process. The idea is to use AI to predict certain simulation outcomes without necessitating the full execution of

these simulations, thus offering a pathway to accelerate the attainment of preliminary results. Although this approach is exploratory and new, it aims to assess the extent to which AI can complement or potentially simplify traditional simulation steps, particularly for repetitive or well-defined scenarios.

In summary, these contributions provide new efficient algorithmic variants and explore and measure how advances in HPC and AI can be integrated into multi-agent traffic simulators. They aim to provide concrete examples of the impact of these models and technologies on traffic simulation, while recognizing the challenges and limitations associated with their implementation.

To provide a more detailed comprehensive understanding of our contributions, we describe in the following the parallel programming model and implementation strategies employed in this thesis.

The Unite and Conquer approach was applied into MATSim by leveraging its inherent multi-level parallelism, fault tolerance, and ability to handle heterogeneous workloads. This approach, termed multiMATSim, involves replicating several instances of MATSim across multiple computing nodes with periodic communications. MATSim natively supports multithreading[31], and in the implementation of multiMATSim on a multi-node parallel machine, each instance corresponds to a single computing node. The following explains the architectural and algorithmic elements of this approach, highlighting the correspondence between them.

The process involved several key steps:

- **Distributed instances**: The approach involves deploying multiple instances of MATSim, with each instance running on a separate computing node. This setup enables parallel execution of the simulation tasks across different nodes. Each node handles a portion of the overall workload, utilizing MATSim's native multithreading capabilities to further enhance parallelism. This multi-

level parallelism allows both inter-node and intra-node concurrency, maximizing computational efficiency.

- **Synchronous and Asynchronous periodic communication**: Periodic synchronous communication between instances is implemented to ensure consistency and coherence in the simulation data. This periodic synchronization helps maintain the accuracy of the distributed simulation by regularly updating and coordinating the state of each instance. Nontheless, an important feature of UC approach is the ability to perform these communications asynchronously. The implementation corresponding to this case is more complex and will be presented in a later chapter.

- **Fault tolerance**: Mechanisms are in place to ensure fault tolerance in the system. If an instance of MATSim fails, its tasks are not redistributed; instead, the other instances continue to operate without interruption, ensuring that the simulation does not crash entirely. This design allows the simulation to continue smoothly even if one or more instances stop working.

- **Heterogeneity**: The approach allows for different workloads to be assigned to different computing nodes. This enables the efficient use of diverse computational resources and optimizes performance based on the specific capabilities of each node, ensuring that each node is utilized to its full potential.

Artificial intelligence and multi-agents simulators

Our second approach consists of using AI for modeling road traffic simulators. More specifically, we use deep neural networks to predict the results of the MATSim simulation. The process involved:

- **Parallel implementation on a single node**: The training of neural networks was initially implemented in parallel on a single computing node. This

proof-of-concept approach utilizes the available CPU resources to train the models efficiently.

- **Neural network training**: Using machine learning algorithms, neural networks are trained on data collected from previous simulation runs. This training process aims to create models that could predict key metrics such as travel times and congestion levels.

- **Evaluation**: The outputs of the trained neural network are evaluated by comparing them to the outputs generated by MATSim. This comparison helps assess the accuracy and reliability of the artificial intelligence models.

- **Future expansion**: While the current implementation is a proof-of-concept running on a single node, the approach is designed to scale. Large-scale experimental results will be available soon. This implementation could be extended to a distributed neural network training setup, leveraging multiple nodes to handle larger datasets and more complex models.

These programming and implementation strategies highlight the practical steps taken to realize the theoretical contributions of this thesis. By detailing the technical aspects, we aim to provide a clear understanding of the methodologies used to achieve the reported performance improvements and the potential applications of these techniques in real-world traffic simulation scenarios.

Finally, these contributions provide new efficient algorithmic variants and explore and measure how advances in HPC and AI can be integrated into multi-agent traffic simulators. They aim to provide concrete examples of the impact of these models and technologies on traffic simulation, while recognizing the challenges and limitations associated with their implementations.

1.4 Organization

State of the Art

We begin with a detailed state of the art, tracing the evolution of multi-agent traffic simulators and discussing recent advances in this field. This includes an analysis of the underlying principles, the challenges encountered, and how HPC concepts contribute to overcoming these limitations. We also evaluate current trends and anticipate future developments.

Unite and Conquer Approach

The core of this thesis is the introduction and application of the Unite and Conquer approach to multi-agent traffic simulators. We justify the choice of this method, explain its fundamental principles, and present the results obtained through specific case studies. The potential impact and future prospects of this approach are also examined.

High-Performance Hardware and Software Architectures

This chapter focuses on the hardware and software architectures underlying the study and effective development of traffic simulations. We explore the fundamentals of targeted hardware architectures, including the Fugaku Supercomputer, and discuss the growing importance of specialized software support.

Modeling of Multi-Agent Traffic Simulators

We then detail our specific contribution to the modeling of multi-agent traffic simulators, presenting two main approaches: one based on the UC method, mul-

tiMATSim, and the other on AI.

Parallel Programming Models for multiMATSim

The discussion continues with an analysis of parallel programming models applied to multiMATSim, highlighting the algorithmic choices and implementation considerations.

Experimental Results

Experiments conducted on the Ruche [54] (a french cluster of heterogeneous CPU with 232 nodes, 9000 cores and GPU nodes with 28 nodes, 68 GPUs) and the Fugaku Supercomputer architectures are described, presenting the results obtained and offering a comparative evaluation of the effectiveness of our methods.

Chapter 2

State of the Art

2.1 Introduction

In the context of urban infrastructure planning and management, road traffic simulation plays a pivotal role. This field has seen significant evolution with the advent of multi-agent traffic simulators, marking a paradigm shift in how transport systems are modeled. By focusing on individual behaviors and interactions, these tools provide a deep understanding of traffic flows, proving vital for developing smart and environmentally friendly mobility solutions. However, these simulators face major challenges, particularly in terms of computational time efficiency. This limitation is a significant obstacle, especially for modeling large-scale or high-resolution scenarios. The primary goal of this chapter is to provide a comprehensive overview of the progress made in the field of multi-agent traffic simulators, highlighting the challenges related to their performance and exploring potential improvement strategie.

This chapter begins with a detailed examination of the evolution of multi-agent traffic simulators, from their macroscopic beginnings to the current, highly sophisticated models. The analysis highlights how these systems have gradually

integrated increasing levels of complexity and realism, more accurately reflecting the dynamics inherent in urban traffic. It then addresses the current challenges and limitations, focusing on computational time performance issues, and underscores the need for advanced computing solutions. This study aims to investigate the contributions of HPC concepts to the advancement of multi-agent traffic simulation models. This section discusses how HPC, which provides the ability to rapidly execute complex calculations and process vast volumes of data, can help overcome performance-related obstacles. Specific case studies are presented to illustrate the impact of applying HPC principles to different simulators, especially in terms of improving computational time, efficiency, and the ability to handle more complex and extensive scenarios. Finally, this state of the art proposes a discussion on current and future trends in the field of multi-agent traffic simulators, particularly related to artificial intelligence, highlighting the most recent innovations. It also identifies emerging research opportunities, based on current gaps and developing trends, thus opening new perspectives for future advancements in this critical field.

2.2 Multi-Agent Traffic Simulators

2.2.1 History and Development

Understanding the evolution of multi-agent traffic simulators is essential to fully appreciate the complexity and usefulness of these tools in modeling our transport networks. The development of these simulators reflects a constant quest for accuracy and efficiency, evolving with technological advances and urban planning needs. This section traces the journey from the initial conceptualizations of traffic flow to today's sophisticated simulators, capable of faithfully representing the dynamic interactions between drivers, vehicles, and infrastructure.

The Beginnings: Macroscopic and Microscopic Models, and the Advent of Computers

The history of traffic simulators took off in the early 1950s[40] [63], a period during which macroscopic models were introduced alongside the advent of computers. These initial models viewed traffic as a homogeneous and continuous flow, offering a very simplified but global view of urban movements. Although they marked a significant advance, their ability to capture the subtleties and individual interactions was nevertheless limited.

With the increase in computer processing power in the 1980s, a new era of modeling emerged with the advent of microscopic models. Benefiting from technological advances in computing, these models represented a significant progress in traffic simulation thanks to their ability to simulate the behavior of each vehicle individually. They allowed for the capture of complex vehicle-to-vehicle interactions, greatly enriching our understanding of traffic dynamics[26].

The Era of Agent-Based Simulators

First and foremost, it is essential to define what an "agent" is in the context of traffic simulators. An agent is an autonomous entity capable of perceiving its environment through sensors and acting upon that environment via effectors. In the context of traffic simulation, an agent typically represents an individual vehicle, pedestrian, or any other actor in the transport system, endowed with the ability to make decisions and execute actions based on its objectives, internal state, and environmental conditions.

A multi-agent system, in turn, is a collection of agents interacting within a shared environment. These interactions can be cooperative, competitive, or neutral, but they are highly relevant for modeling the complexity of real traffic systems. In these systems, each agent operates independently while concurrently consider-

ing the actions and states of other agents. This approach enables a dynamic and faithful simulation of traffic phenomena at both the microscopic and macroscopic levels. Moreover, these methodologies facilitate the observation of the emergence of macro-level phenomena from micro-level interactions. Such insights underscore the utility of agent-based modeling in comprehending complex dynamics across various contexts. For a comprehensive exploration of this topic, readers are encouraged to consult "Agent-Based and Individual-Based Modeling" by Railsback and Grimm[60], which is regarded as a seminal text in the field of agent-based systems and simulations.

The 1990s and 2000s represented a period of profound transformation with the introduction of agent-based simulators. Platforms such as MATSim, SUMO [4], and VISSIM [24], along with others [51], have outperformed previous models by incorporating multi-agent elements. This evolution enabled more detailed and accurate modeling of individual decisions and behaviors. This era was also marked by the increasing integration of artificial intelligence technologies, opening new avenues for the simulation and prediction of traffic behaviors.

Integrating Advanced Technologies

Entering the new millennium has been characterized by a remarkable synergy between traffic simulators and cutting-edge technologies. The adoption of data from intelligent transportation systems and urban sensors has enriched simulators with real data, significantly improving their accuracy and relevance. This period is also dedicated to exploring the potential of high-performance computing in research, which significantly enhances the ability to process large volumes of data, making simulators faster and more capable of handling increasingly complex traffic scenarios. This historical evolution of multi-agent traffic simulators illustrates a convergence between technological progress and the growing need to understand

and manage increasingly complex urban transport systems. As such, it acts as the cornerstone for our investigation into incorporating HPC concepts into these simulators, a step we recognize as vital for tackling modern traffic modeling challenges.

2.2.2 Principles and Operation

Multi-agent traffic simulators are an important tool in the study of transportation systems. They rely on sophisticated modeling principles, enabling detailed analysis of behaviors and interactions within traffic systems. By considering each driver, pedestrian, or vehicle as an autonomous agent, these simulators offer an in-depth representation of traffic dynamics, thus facilitating the study of urban and interurban flows.

At the heart of these simulators is the concept of the agent, which, as previously mentioned, is an autonomous entity characterized by specific attributes, goals, and behaviors. This tailored approach is indispensable for accurately modeling the decisions and behaviors of different traffic elements, be it routine maneuvers or reactions to evolving traffic conditions. The interactions between agents, steered by algorithms, emulate a plethora of human behaviors and diverse traffic scenarios.

These interactions are essential for multi-agent simulation. They allow agents to react and adapt based on the actions of others and environmental changes, thus offering the possibility to reproduce a wide range of traffic phenomena. This adaptability is an asset for analyzing the effects of different traffic scenarios and for strategic planning.

The validation and calibration of simulators with real traffic data is a key step to ensure the accuracy of the modeling. This process aims to align simulations with empirical data to improve the reliability of the generated predictions. By adjusting the model parameters, researchers seek to ensure that the simulation

faithfully represents real traffic conditions.

Finally, flexibility and scalability are important characteristics of multi-agent traffic simulators. They can be adapted to different traffic scenarios and incorporate new forms of agents or behaviors, thus meeting the changing needs in the transportation field.

In conclusion, multi-agent traffic simulators offer a unique perspective on the dynamics of transportation systems. They combine advanced modeling methods with detailed analysis of human behavior, proving useful for studying, forecasting, and optimizing traffic flows in current and future urban environments. To illustrate the advancements and applications of multi-agent traffic simulators, we now turn our attention to some key examples in the field. These examples highlight the evolution and sophistication of traffic simulation tools over time, demonstrating their capabilities in modeling and analyzing traffic dynamics.

2.2.3 Key Examples

Multi-agent traffic simulators have evolved over time, gaining sophistication in modeling and analyzing traffic dynamics. Notable tools in this domain include SUMO, VISSIM, and MATSim. Each of these simulators has specific features and meets various needs in research and urban planning.

SUMO (Simulation of Urban MObility)

Developed by the German Aerospace Center (Deutsches Zentrum für Luft- und Raumfahrt, DLR), SUMO is an open-source multi-agent traffic simulator. Written in C++, it is characterized by its ability to model urban transport networks of various scales, from neighborhoods to entire cities, with particular attention to traffic flow details.

SUMO's modular architecture allows the integration of custom modules, offering users flexibility in creating and managing complex simulations. Its ability to simulate various types of traffic interactions, including driving behaviors and vehicle-pedestrian interactions, is a major asset.

However, SUMO also has its limitations. As an open-source simulator, its performance may vary depending on the quality and frequency of community contributions. Although capable of handling complex scenarios, managing large urban simulations can be demanding in terms of computational resources, particularly memory and processing time.

Moreover, SUMO requires specific technical expertise for its efficient configuration and use, which can pose a barrier for less experienced users. The reliance on precise and detailed traffic data for model calibration and validation can also be a challenge, especially in areas where such data are not readily available.

Ultimately, while SUMO stands out for its flexibility and accessibility as open-source software, facilitating adaptation and customization for various simulation scenarios, it presents certain limitations in terms of driver behavior modeling and detailed traffic visualization. These aspects are pivotal for conducting a thorough analysis of vehicle-to-vehicle and vehicle-environment interactions. Here, VISSIM comes into play, offering a more sophisticated approach with refined driver behavior models and superior visualization capabilities, thus addressing some of the needs unmet by SUMO.

VISSIM (Verkehr In Städten - SIMulationsmodell)

Developed by PTV Group, VISSIM is a multi-agent traffic simulator specialized in microscopic traffic modeling. PTV Group, a company known for its innovative solutions in traffic and transportation planning, has created VISSIM to simulate the individual behaviors of drivers and pedestrians in a variety of urban and in-

terurban environments. It is used for scenarios ranging from isolated streets to city-scale transport networks.

VISSIM stands out for its ability to model complex traffic interactions, including a variety of vehicles, cyclists, and pedestrians. It uses physical behavioral models and heuristic rules to simulate dynamic interactions between agents in a traffic environment.

However, VISSIM has its limitations. Its microscopic modeling nature, while beneficial for detail accuracy, can lead to increased complexity and computational time for large networks. Furthermore, the quality and accuracy of VISSIM simulations heavily depend on the availability and precision of input data, which can be an obstacle in contexts where traffic data are limited or outdated.

Another consideration is the cost of the license, potentially prohibitive for some users, especially in academic settings or for small municipalities. Additionally, although VISSIM offers a detailed graphical interface, its learning curve can be steep for new users, particularly those unfamiliar with advanced traffic simulation tools.

In summary, VISSIM, with its advanced simulation models and detailed visualizations, excels in accurately representing traffic flows and individual driver behaviors. However, when it comes to addressing the simulation of large networks over long periods or modeling the travel choices and traffic patterns of an entire population, VISSIM may encounter limits in terms of scale and data management complexity. At this juncture, MATSim proves to be an essential complement, as it is specifically designed to handle large-scale simulations with a focus on modeling travel behaviors and optimizing routes at the population level, thus offering a complementary perspective to that of VISSIM.

MATSim (Multi-Agent Transport Simulation)

Developed in Java, MATSim is a multi-agent traffic simulator, resulting from collaboration among several renowned research institutes, including ETH Zurich and the Technical University of Berlin. This simulator stands out for its ability to simulate individual mobility behaviors within extensive transport networks, with a particular focus on details and complex interactions.

MATSim's use of multithreading[30] allows it to handle large and complex traffic simulations, an asset for analyzing urban scenarios involving a significant number of agents. This functionality makes MATSim a relatively powerful tool for large-scale studies.

The modularity of MATSim is another strength, allowing for considerable adaptability for specific research and planning needs. Its ability to integrate and calibrate models based on real transport data improves the accuracy and relevance of simulations.

However, MATSim presents certain challenges. The complexity of its configuration and use is one such example, requiring in-depth technical expertise. Furthermore, the detailed simulations provided by MATSim require significant computational resources, potentially limiting its applicability in environments with constrained computing capabilities.

MATSim's reliance on precise and comprehensive traffic data for model validation is also a factor to consider. Where such data are not available, the reliability of simulation results could be impacted.

MATSim is an advanced multi-agent traffic simulator, designed for precise modeling of complex urban transport systems. Its advanced processing capabilities, flexibility, and integration of real data make it a valuable tool, despite the challenges related to its technical complexity and the need for significant computational resources. While these tools showcase the capabilities and advancements

in traffic simulation, they also highlight the inherent challenges and limitations of multi-agent traffic simulators.

The main features of SUMO, VISSIM, and MATSim are presented in Table 2.1.

Feature	**SUMO**	**VISSIM**	**MATSim**
Type of simulator	Multi-agent traffic sim	Multi-agent traffic sim	Multi-agent traffic sim
Prog. language	C++	C++	Java
Open source	Yes	No	Yes
Year of dev.	2001	1992	2006
Dev. institution	Deutsches Zentrum für Luft- und Raumfahrt (DLR)	PTV Group	ETH Zurich
Modeling capabilities	Vehicles, pedestrians, public transport	Vehicles, pedestrians, public transport	Vehicles, pedestrians, bicycles, public transport
Scenario types	Urban, interurban, pedestrian, public transport	Urban, interurban, pedestrian, public transport	Urban, interurban, regional, public transport

Table 2.1: Comparison of Multi-Agent Traffic Simulators

Feature	SUMO	VISSIM	MATSim
User community	Large, active	Commercial	Large, academic
Vis. tools	SUMO-GUI, SUMO-web	VISSIM-GUI	Via plug-ins (JOSM, QGIS)
Common applications	Urban planning, traffic management, public transport studies	Urban planning, traffic management, public transport studies	Urban planning, academic research, public transport studies
Integration support	Traci (API), OSM, NetEdit	API, OSM, integration with other PTV tools	Plug-ins, OSM, NetEdit
Doc. and support	Comprehensive and active doc.	Commercial support and doc.	Comprehensive and active doc.

2.3 Challenges and Limitations of Multi-Agent Simulators

Multi-agent traffic simulators, while essential in modeling complex transportation systems, face intrinsic constraints that can limit their efficiency and applicability. Major issues lie in performance challenges and the balance between complexity, realism, and computational capacity. Simulating realistic traffic environments on a large scale raises questions of computing power, marked by prolonged computation times and intensive use of resources. Furthermore, the aspiration for increased

realism in modeling the behaviors and interactions of agents highlights a dilemma between enhancing model sophistication and preserving acceptable computational performance. This introduction sets the stage for an in-depth exploration of these issues, shedding light on problems related to optimizing performance and managing the inherent complexity of multi-agent traffic simulators.

2.3.1 Performance Issues

Performance is a paramount issue in the field of multi-agent traffic simulators. These tools, designed to faithfully represent the complexity and dynamics of urban transport systems, face significant computational constraints that directly affect their efficiency and practical applicability. In this section, we explore various aspects of performance problems, examining the implications of large-scale traffic modeling and highlighting the difficulties related to managing processing capacities.

Simulating multi-agent traffic requires detailed modeling of thousands or even millions of individual agents, each with its characteristics, behaviors, and interactions. This microscopic approach, advantageous for capturing the complexity of traffic systems, results in a significant computational load. The computation time required for these simulations can be substantial, especially for large-scale scenarios or those requiring high resolution. Researchers and planners are often faced with a difficult choice: how to balance the need for detailed and realistic modeling with the constraints of available computational resources?

Moreover, performance issues are not limited to computation time. The efficiency of memory management, data storage, and communication between computing units are also critical factors. In a simulation environment, where each agent can interact with many other agents and elements, the volume of data to process and store can become considerable. This poses difficulties not only in terms

of computational efficiency but also in terms of the reliability and robustness of simulations.

Optimizing the performance of multi-agent traffic simulators thus becomes a multidimensional goal. It requires improvements both at the level of algorithms and data structures and in the software and hardware architecture used for simulations. Integrating concepts related to parallel and distributed computing, exploiting the capabilities of high-performance computing systems, and optimizing communication protocols between processes are among the strategies that can be adopted to meet these challenges.

These optimization strategies, while essential, often lead to a fundamental dilemma in the development of multi-agent traffic simulators: the balance between the complexity and realism of models on one hand, and their performance in terms of execution time on the other.

2.3.2 Complexity and Realism vs Computational Performance

In the development of multi-agent traffic simulators, a fundamental dilemma arises: the balance between the complexity and realism of models on one hand, and their performance in terms of execution time on the other. Here we examine this dilemma, assessing the repercussions of an increased quest for realism on the performance of simulators, as well as its impact on their practical utility. Multi-agent traffic models present inherent complexity at several levels. Behaviorally, the need to accurately model the decisions and interactions of each agent - drivers, pedestrians, or autonomous vehicles - requires advanced and nuanced algorithms. These algorithms must not only reflect individual behaviors but also contextualize these behaviors within the entire traffic system. The realism of simulations is also dependent on the ability to integrate a wide range of variables, including

environmental conditions, human factors, and unforeseen events. However, an increase in the complexity and realism of simulations can translate into a decrease in computational performance. Exhaustive models and dynamic traffic scenarios demand considerable computational resources, leading to prolonged computation times, intensive memory usage, and a strong need for storage capacity. This requirement becomes particularly problematic in the simulation of vast transport networks or over long periods, where the volume of data and the number of interactions to process increase exponentially. The main challenge, therefore, is to find an optimal balance between model complexity and computational performance. In this regard, the introduction of new algorithmic concepts, optimization of data structures, and improvement of simulator architecture becomes imperative. The goal is to optimize simulators capable of handling complexity without sacrificing performance, thus facilitating more realistic, accurate, and useful traffic simulations for understanding and improving transport systems.

To address these challenges and push the boundaries of what multi-agent traffic simulators can achieve, the integration of HPC concepts becomes essential. By leveraging HPC, we can enhance the processing capabilities of these simulators, enabling them to handle more complex scenarios and larger datasets with greater efficiency and accuracy. The following section delves into how HPC principles contribute to the advancement of multi-agent traffic simulators, highlighting their impact on simulation and analytical capabilities.

2.4 HPC-Oriented Design of Multi-Agent Traffic Simulator

2.4.1 HPC and its Impact on Traffic Simulation

The development of High Performance Computing represents a significant advancement in the processing and analysis of vast data sets, offering solutions to problems related to calculations of unprecedented complexity. This section provides an introduction to the fundamental principles of HPC, specifically exploring its impact on enhancing analytical and simulation capabilities in the context of multi-agent traffic simulators.

HPC involves not only the joint use of computing resources, typically beyond the capabilities of individual computers or standard servers, to provide significantly higher processing power but also the implementation of advanced algorithms optimized for parallel computation. This power is often achieved through clusters of processors, high-speed communication networks, and advanced storage systems, allowing for the parallel and efficient processing of large quantities of data.

In the realm of traffic simulators, HPC broadens the possibilities for simulations and facilitates the management of complex interactions among numerous agents and the modeling of extensive traffic scenarios. However, it is crucial to distinguish between two primary approaches: applying HPC techniques to existing simulators and designing new simulators specifically optimized for HPC.

- **Applying HPC to existing simulators:** This approach involves parallelizing existing traffic simulation models to leverage the computational power of HPC. By doing so, it addresses performance constraints, drastically reducing the time required for simulation calculations. This improvement is particularly relevant for traffic models demanding high computational power

to faithfully simulate interactions among hundreds of thousands or even millions of agents.

- **Designing HPC-optimized simulators:** In contrast, designing simulators specifically oriented HPC entails creating new algorithms and system architectures that are inherently optimized for parallel processing.

In the context of our research, we adopt a hybrid solution consisting of using a mix of these two approaches. Indeed, we apply a parallelization technique to MATSim, our case study of multi-agent simulators for road traffic. This parallelization is essentially applied to the MATSim core. However, our approach does not consist of simply parallelizing the existing program of this kernel but of modifying its design in order to improve not only its computational performance but also its algorithmic performance. For this, we apply the UC approach to MATSim to produce multiMATSim.

HPC-oriented design of multi-agent traffic simulators has multiple and significant implications. Firstly, HPC facilitates the processing and analysis of large quantities of traffic data, allowing for a deeper understanding and more accurate forecasts of traffic dynamics. Secondly, it enables the handling of more complex and realistic traffic scenarios, thus contributing to more effective planning and decision-making in urban transport. Consequently, HPC is not merely a response to existing computational challenges but also a driver of progress in traffic simulation, heralding a new era of traffic modeling and transport planning characterized by levels of precision, efficiency, and scope previously unattainable.

The following sections will examine in more detail how HPC principles have been integrated into traffic simulators, the performance improvements that have resulted, and the future prospects they open for the evolution of traffic modeling and transport management.

2.4.2 HPC Applied to Traffic Simulators

Optimization of Multi-Agent Traffic Simulators through Division and Distribution

A key strategy frequently cited in the enhancement of multi-agent traffic simulators is the efficient division and distribution of the network and agents, especially when incorporating HPC-related concepts. This approach is essential for managing the complexity and high computational load of simulations. It involves breaking down the transport network into smaller segments and distributing these segments, as well as the agents, across multiple computing units. This method is particularly relevant for simulations encompassing vast networks and a large number of agents, due to the high demand for computational resources they entail. The use of division and distribution leverages parallel computing to reduce computation times by simultaneously executing multiple operations. However, the efficiency of this approach depends on strategic implementation, taking into account the specific configuration of the transport network and traffic models. One of the challenges is to minimize interdependencies between the distributed sub-networks to limit the need for frequent and costly communications between computing units. The goal is to find a balance that promotes data localization and reduces the amount of communication, thus optimizing the overall performance of the simulation.

Various methods have been explored to further enhance the efficiency of these simulations by integrating advanced HPC concepts. Significant advancements have been made in this area, offering new perspectives for improving performance. Potuzak et al. introduced two communication protocols suitable for networks segmented into sub-networks[58]. The first protocol, SC-SV, semi-centralized, maintains data integrity without reducing inter-process communications. The second, named Long Step, effectively reduces the number of messages by spatially and

temporally combining data related to vehicles and roadways. Further studies by Potuzak et al. examine the advantages of traffic simulation on a multi-core computer cluster[57], demonstrating that the parallel/distributed version reduces inter-process communications and accelerates performance. Ramamohanarao et al. developed a spatial workload balancing strategy using the SMARTS simulator[62]. This method aims to decrease the number of communication channels between processes to reduce communication costs, while facilitating rapid and large-scale simulations. Xu et al. focused on temporal synchronization issues in parallel traffic simulations[74], proposing heuristics to increase lookahead values, thus reducing the time spent on synchronization and speeding up the pace of parallel simulations without sacrificing the statistical accuracy of results. Finally, the Neighbour-Restricting Graph-Growing (NRGG) algorithm developed by Xu and colleagues[73] optimizes the partitioning of networks in traffic simulations to minimize communications between logical processes (LP) of different sub-networks, providing an efficient solution to enhance the performance of parallel road traffic simulations.

The study by Whitanage et al.[70] presents a method for dividing road traffic networks into optimal connected sub-networks with uniform traffic flow patterns using taxi trip data to calculate edge weights. The method follows a multilevel scheme: graph coarsening with random matching, initial partitioning with recursive bisection, and refining with the KL algorithm. The algorithm runs 10 times, selecting the best division. Applied to Colombo city's road network (33,979 crossroads and 100,000 edges over 2,654 km^2), this method showed fewer border edges compared to the METIS partitioning algorithm[35] (a software package for partitioning large-scale graphs and optimizing fill-reducing orderings of sparse matrices). However, it does not consider load balancing among sub-networks, as it is not intended for parallel or distributed traffic simulation.

Anwar et al.[5] introduce a method for dynamically dividing road traffic networks to optimize traffic control by maintaining homogeneous sub-networks despite changing congestion levels. This approach was validated using Melbourne's traffic network, which includes 2,928 crossroads and 7,245 roads, and further tested on larger semi-synthetic networks with up to 28,465 crossroads and 53,494 roads.

Acosta et al.[2] leveraged the traffic simulator SUMO for a parallel/distributed approach to road traffic network division. Their work focuses on managing border edges to minimize their impact on vehicle movement, rather than detailing a specific division algorithm. This approach is demonstrated on a small regular grid with 4 nodes (crossroads) and 12 edges (roads).

Chang Liu et al.[41] propose a dynamic method for dividing road traffic networks to improve real-time management and control. The approach aims to create sub-networks with uniform density by clustering road segments with similar traffic characteristics. Starting with a static division to minimize cutting costs and enhance intra-subnetwork correlation, the method dynamically refines the structure through merging, cutting, and adjusting boundaries. This method was demonstrated on the road traffic network of Farmers Branch city, near Dallas, proving its effectiveness.

Xu et al.[75] present a method for dividing road traffic networks to enhance distributed simulation, utilizing a hypergraph representation and the hMETIS algorithm[36] (hMETIS is an extension of the METIS software package[35] specifically designed for hypergraph partitioning, which is useful for more complex scenarios where relationships between data points are not limited to pairs.). The focus of this approach is to reduce inter-process communication. The method is demonstrated on the Singapore road network, which comprises 10,702 crossroads, 21,333 roads (approximately 3,200 km in length), and about 950,000 vehicles.

Qiang Liu et al.[42] developed a method to enhance traffic management by

dividing road networks into sub-networks using an improved Newman Fast Clustering Algorithm (NFCA)[50]. This approach employs a weighted graph, with static weights representing distances between crossroads and dynamic weights indicating traffic flow. The method's efficiency, measured by modularity (the ratio of border roads to inner roads), was tested on a small network of 18 crossroads and 26 bidirectional roads. The improved NFCA showed better performance compared to the original algorithm.

Finally, Potuzak[56] conducted an extensive review of methods for dividing road traffic networks for distributed or parallel traffic simulations, identifying several key trends in the field. He highlighted the frequent use of real road networks and regular square grids for testing proposed methods. Many of these methods are compared to existing algorithms like METIS[35] to evaluate their relative performance. Most methods employ static load-balancing, though some use dynamic load-balancing to enhance real-time management. Additionally, many approaches leverage real traffic data, obtained from GPS, cell phones, or stationary sensors, to refine their models. Finally, Potuzak noted the common use of techniques such as K-means clustering, hierarchical clustering, and graph growing to effectively divide road networks.

Agent Distribution: An Alternative Perspective

In addition to conventional methods of dividing and distributing networks in multi-agent traffic simulators, an alternative perspective warrants attention: that of distributing the agents themselves. This approach, highlighted by Mastio et al.[43], delves into the mechanisms and implications of distributing agents within multi-agent traffic simulators. Mastio et al. explored two methods of distribution in a multi-agent environment, focusing on the balance and efficiency of simulations.

The first method, referred to as centralized synchronization, proposes divid-

ing the entire set of agents into distinct subsets, with each subset allocated to different servers. This strategy involves constant communication between servers to ensure each unit has access to up-to-date information on the overall network state. Although this method aims to minimize the necessary communication by keeping agents on their original server, it requires rigorous synchronization. This synchronization is crucial for reliably modeling the network dynamics, which are influenced by the movements of agents managed by different hosts. The primary focus is on maintaining a balance in computational load across servers while ensuring consistent network-wide information.

In contrast, the second method, known as environmental distribution, focuses on grouping agents on a server based on their geographic proximity within the network. Unlike the first method, which distributes agents somewhat independently of their physical locations, environmental distribution aligns the network's constituent elements, such as vertices and edges, with specific servers. Consequently, the agents located on these elements are grouped together, which facilitates more localized handling of interactions and movements. This method generates two main types of communications: the synchronization of information related to the weights of edges between servers and the transfer of agents from one server to another when they move to a vertex managed by a different server. The aim here is to reduce overall communication overhead by maximizing the local processing of interactions and minimizing cross-server data transfers, which is particularly beneficial in simulations where agent movements are frequent and geographically coherent.

Mastio et al.'s analysis concludes that, for their specific multi-agent mobility simulator model, the agent distribution method offers superior performance to environmental distribution. However, this study also highlights a critical issue: balancing the computational load between computing units. This finding

underscores the importance of considering dynamic load balancing mechanisms to further improve the performance of simulations. The figures 2.1 and 2.2 illustrate the differences in distribution. In the first, the color correspondences are shown on the agents (colored points), whereas in the second, the color correspondences with the nodes are depicted on the subnetworks (colored areas).

Another study conducted by Yadong Xu et al.[72] addresses city-scale nanoscopic traffic simulation to tackle increasing issues of congestion, collisions, and emissions. The architecture of the SEMSim traffic simulator is described, emphasizing the importance of parallelization and reducing dependencies between logical processes (LPs). Nano-agents, consisting of a driver and a vehicle, are modeled using behavioral and vehicle components. A discrete-event approach is favored to manage varied time scales effectively. A multi-objective optimization for the dynamic allocation of agents to clusters is proposed, incorporating knowledge of the road network topology. Future work will focus on implementing this simulation and developing dynamic methods to tackle this complex problem.

Impact of Inter-Node Communications and Future Research Directions

Efficient management of inter-node communications remains a major challenge in optimizing multi-agent traffic simulators. While division and distribution methods bring significant benefits in terms of complexity and computational load management, they often introduce an overhead related to the need to synchronize and communicate information across the computing network. The continuous data transmission between servers to maintain the integrity of simulations generates an overhead that can affect overall performance. Data exchanges, especially when frequent or voluminous, can cause significant latency and reduce the efficiency of parallel computing.

By analogy, we can consider these methods as Divide & Conquer techniques,

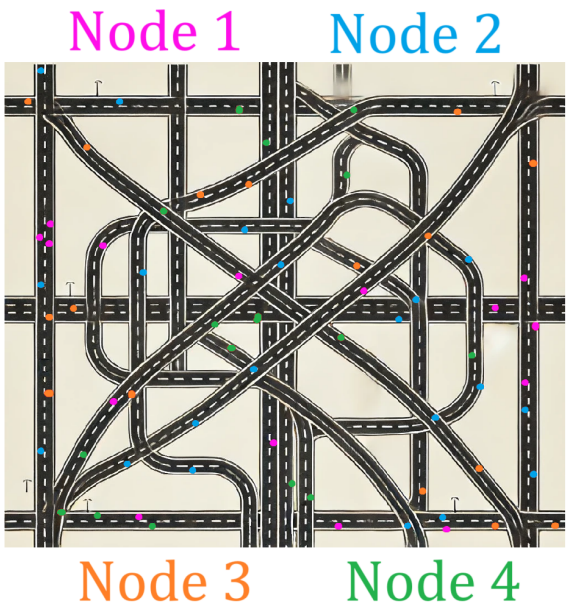

Figure 2.1: Distribution of agents

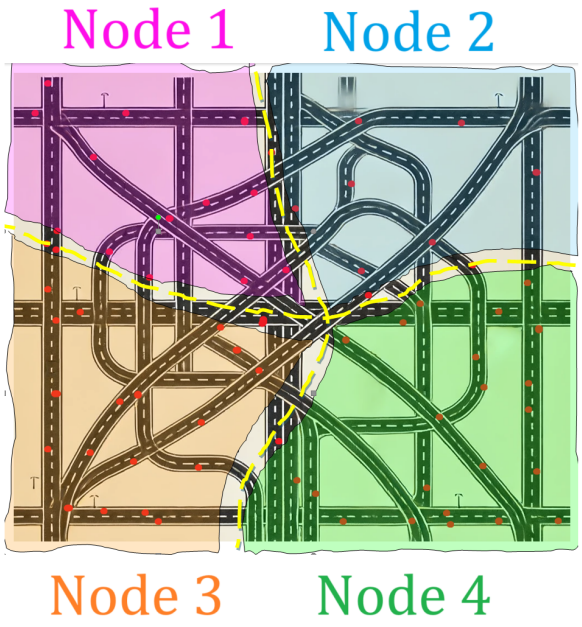

Figure 2.2: Distribution of subnetworks

which consist of dividing a problem that is difficult to solve into sub-problems that are easier to solve. In the next chapter, we propose an opposite technique to Divide & Conquer, called Unite & Conquer. This approach aims to improve the performance of these types of simulators by applying a unified strategy to address the challenges posed by complex simulations.

The challenge, therefore, lies in developing more efficient communication protocols and synchronization methods capable of minimizing communication costs while ensuring the accuracy and reliability of simulations. This issue represents a critical research domain for the future: developing solutions to mitigate the impact of inter-node communications. Advances in this area could lead to a new generation of multi-agent traffic simulators, capable of managing vast networks and a large number of agents with unprecedented efficiency and precision, thus transforming current constraints into opportunities for future breakthroughs.

Advantages of the Event-Driven Approach in Distributed Computing

Identifying the reduction of inter-node communications as a future research axis invites consideration of the event-driven approach as a potentially fruitful strategy. Adopting this event-based model marks a notable evolution in the design of multi-agent traffic simulators and offers an effective alternative to traditional models based on regular time intervals. Event-driven simulators operate by focusing on moments when significant changes occur in the system, rather than constantly updating the state of each agent at fixed intervals.

This specificity gives the event-driven approach remarkable potential to reduce unnecessary calculations and optimize the use of computing resources. In a distributed computing context, it minimizes the need for frequent and voluminous communications between computing nodes, as data exchanges occur in response to specific events, thus mitigating latency and communication costs.

Simulators such as MATSim[31] and SUMO[4] incorporate event-driven elements to efficiently manage various aspects of traffic simulations. These tools illustrate how reacting to events, rather than a fixed time sequence, can reduce interactions between nodes while providing accurate and detailed simulation of traffic dynamics.

However, it's important to recognize that the efficiency of the event-driven approach may vary, and depending on the context. In some scenarios, such as those with complex interactions or frequent and unpredictable state changes, this method can significantly reduce unnecessary calculations and communications. Conversely, in situations where high temporal precision is required, the time-based method may prove more adequate.

The choice between event-driven models and those based on time intervals should be guided by the specifics of the traffic scenario to be simulated and the objectives of the simulation. Incorporating the event-driven approach into multi-agent traffic simulators represents a notable advancement, particularly in distributed computing systems where minimizing inter-node communications is of paramount importance. Nonetheless, a careful evaluation of the benefits and limitations of each approach is necessary to optimize the performance of simulations and continue to innovate in the field of multi-agent traffic simulators.

GPU Computing in Multi-Agent Traffic Simulators

The GPU (Graphic Processing Unit), originally developed for graphics processing, has become a major component in the field of HPC due to its ability to execute a large number of threads simultaneously. This capability makes it particularly suited to parallelizable tasks, where a multitude of calculations can be performed in parallel. GPUs excel in parallel processing of simple repeated tasks across a large data set. This ability stems partly from their SIMD (Single Instruction, Multiple

Data) architecture, which allows a single instruction type to simultaneously process multiple data.

In the context of multi-agent traffic simulation, the SIMD approach integrated into GPUs finds an efficient application as it allows managing and executing calculations for many agents (vehicles, pedestrians, etc.) in parallel. Each agent can be considered as a set of data on which the same operations need to be performed (e.g., calculating position, speed, or detecting collisions). Therefore, the use of GPUs and the SIMD architecture significantly accelerates these simulations by reducing the time needed to process all agents in the simulated environment, making multi-agent road traffic simulations both faster and more realistic.

GPU Suitability for the Multi-Agent Paradigm As a reminder, in a multi-agent system, each agent can be considered an independent entity with its own rules and behaviors. The parallel processing capabilities of GPUs efficiently simulate many agents simultaneously, which is essential for reproducing the complexity and dynamics of real transport systems. GPUs facilitate the modeling of individual behaviors and interactions between agents on a scale and speed that would be difficult to achieve with conventional CPUs.

Studies such as those by Hermelin et al.[27] and Rajf and Potuzak[61] have explored the use of GPGPU (General-Purpose processing on Graphics Processing Units) in traffic simulators. Hermelin et al. highlight the benefits of hybrid solutions, which combine the modeling flexibility of agents and the efficiency of parallel GPU processing. Rajf and Potuzak, on the other hand, compared the performance of traffic simulations on GPUs and CPUs, demonstrating the significant acceleration potential offered by GPUs.

In the research conducted by David Strippgen and Kai Nagel[68], the authors delve into the efficiency of GPUs in accelerating road traffic simulation. They

implemented a specific data structure, known as the "circular buffer", for optimally storing data on GPU memory.

The "circular buffer" is a storage technique where data is organized in a fixed-size array, and when this array is filled, new data replaces the old in a cyclic manner. This approach avoids the overhead associated with dynamic memory allocation, facilitating efficient use of GPU memory.

By utilizing this optimized data structure, the authors fully leveraged the computing capabilities of GPUs to simulate complex traffic networks. The results demonstrated a significant improvement in performance compared to traditional methods, with a speed gain of up to 67 times compared to a highly optimized Java version for the MATSim traffic simulator.

These studies underscore that, while GPUs offer considerable parallel computing power, the performance gain varies depending on various factors, such as the traffic network structure and hardware specifics. Optimal management of these factors is key for maximizing the benefits of parallel GPU computing in multi-agent traffic simulations.

Another study conducted by Aleksandr et al. describes the development of GEMSim, a GPU-based mobility simulator designed for large-scale networks and generic population samples[64]. This simulator aims to efficiently exploit the massively parallel architecture of GPUs to accelerate mobility simulations. Adjustments to the simulation loop structure, organization of memory transactions on the GPU, and data structures are detailed to maximize performance.

Preliminary results for a large-scale scenario in Switzerland demonstrate significant acceleration of mobility simulations with GEMSim compared to MATSim, which is CPU-only. Indeed, GEMSim proves to be more than 12 times faster than MATSim in certain configurations, and up to 58 times faster for mobility simulations. This approach thus provides more accessible and faster traffic simulation

and forecasting tools for industry professionals.

Z. Chen et al.[67] developed a dynamic method for dividing road traffic networks, achieving a speedup of up to 110 times by leveraging GPU capabilities. This method, based on a parallel genetic algorithm, is designed for traffic management and microscopic simulations. It was tested on a grid with 25 crossroads and a small road network in Zhongguancun, Beijing, demonstrating significant performance improvements over the CPU version.

As we continue to explore the advancements in HPC applied to traffic simulators, it becomes evident that integrating cutting-edge technologies can significantly enhance simulation capabilities. Among these technologies, artificial intelligence is poised to play a pivotal role in the future of traffic modeling, offering unprecedented accuracy and predictive power.

2.5 AI and Road Traffic Simulation

Road traffic simulation is on the cusp of a radical transformation, driven by rapid advancements in AI, HPC, and digital technologies. This section explores the major innovations that are redefining traffic simulation, with a particular emphasis on distributed artificial intelligence, and examines how these technologies are shaping the future of urban transportation systems. Distributed AI plays a key role in the evolution of traffic simulations, enabling more complex analyses and more precise modeling of road user behaviors. Thanks to distributed AI, simulations can now manage complex multi-agent systems, where each agent (vehicle, pedestrian, cyclist) operates autonomously but is capable of interacting with and adapting to the actions of others, thus offering a dynamic and realistic representation of urban traffic.

2.5.1 Integration of Autonomous Vehicles

A specific application of distributed AI is the integration of autonomous vehicles into existing traffic. The advent of autonomous vehicles represents a gradual evolution of mobility technologies, marking a significant milestone in how transportation systems are conceptualized, developed, and managed. This transition towards greater vehicle autonomy holds the potential to transform traffic dynamics, offering new avenues for improving the safety, efficiency, and sustainability of urban transport networks while adding another element of complexity to the system.

Autonomous vehicles, through their ability to communicate with each other and with infrastructure via V2X technology, introduce an additional dimension to road traffic simulation. This interaction allows for more precise modeling of traffic flows, providing real-time data on traffic conditions, vehicle behaviors, and responses to various environmental stimuli.

At the heart of this evolution, the capability of autonomous vehicles to operate without direct human intervention relies on advanced systems of sensors, AI algorithms, and data processing technologies. These technologies enable vehicles to make safe and efficient navigation decisions, respond to unexpected changes in the traffic environment, and adapt to real-time traffic management strategies.

The integration of autonomous vehicles into road traffic simulations thus offers the possibility to explore new mobility scenarios, assess the impact of traffic interventions, and design more resilient transportation systems. For example, by simulating interactions between autonomous and non-autonomous vehicles, researchers can identify strategies to minimize congestion, reduce accident risks, and optimize travel times. V2X communication, which encompasses vehicle-to-vehicle (V2V), vehicle-to-infrastructure (V2I), vehicle-to-pedestrian (V2P), and vehicle-

to-network (V2N), represents a key technological advancement in the evolution of traffic simulation. This technology facilitates a continuous and multidirectional exchange of information, essential for improving traffic flow and safety. Its implementation opens up significant possibilities for more accurately modeling complex interactions within the transport system.

The integration of V2X communication into traffic simulation models implicitly relies on the use of AI and HPC. AI, with its advanced analytical and predictive capabilities, enables the interpretation of data exchanged via V2X to anticipate traffic conditions, detect potential incidents before they occur, and suggest real-time adjustments to optimize traffic flows. This analysis is made possible by the computing power provided by HPC, which efficiently processes the large volumes of data generated by V2X systems, ensuring a dynamic and responsive simulation of traffic scenarios.

The convergence of V2X communication with AI and HPC opens new horizons for road traffic simulation, enabling a more faithful and adaptive representation of urban dynamics. This integrated approach promises not only better traffic management but also a significant reduction in accident risks, an enhanced driving experience, and optimized urban planning strategies.

However, realizing this potential requires careful attention to challenges related to data security, privacy protection, and system reliability. The successful implementation of these advanced technologies in road traffic simulation will depend on the ability to navigate these ethical and technical considerations, highlighting the importance of ongoing collaboration among researchers, engineers, and policymakers.

To assess the effectiveness of traffic simulations using V2X (Vehicle-to-Everything) technology, Schünemann implemented his innovative infrastructure, the V2X Simulation Runtime Infrastructure (VSimRTI)[66]. In their study, they explored how

this infrastructure could be utilized to couple two well-known traffic simulators, VISSIM and SUMO. The goal was to understand how the use of V2X technology could influence the simulation process in terms of the total time required. They found that coupling these simulators effectively reduced the simulation time compared to using VISSIM alone. This reduction in simulation time is crucial in the context of V2X simulation, as it allows for accurate results within reasonable timeframes, which is critical for facilitating the analysis of intelligent transportation system performances. Thus, this study highlights the importance of V2X technology in optimizing traffic simulations, offering promising prospects for more effective and realistic applications in this rapidly expanding field.

Another study conducted by Lejun Jiang et al. [34] aimed to assess the effectiveness of V2X communications in predicting road traffic. Their findings indicated that even with a low penetration rate of connected vehicles, significant traffic predictions could be achieved. Utilizing a communication range of several hundred meters for roadside units, they observed a consistent coverage rate of about 90%. They also demonstrated that a strategic deployment of roadside units could maximize the efficiency of traffic predictions. Their work highlighted the potential of V2X communications to enhance road traffic prediction and, by extension, traffic management and road safety.

Lastly, Ali R. Abdellah et al. proposed the use of a unidirectional Long Short-Term Memory (LSTM) neural network model for traffic forecasting in Vehicle-to-Everything (V2X) networks[1]. This model improves prediction accuracy by leveraging past temporal data, facilitating more informed decision-making. The simulation results showed that the model achieved its highest level of accuracy when the transmission rate was 4 packets/s, thereby surpassing competing methods. However, performance declined as the transmission rate increased, with lower accuracy and longer processing times for models predicting at 14 packets/s.

These conclusions underscore the effectiveness of the proposed model for V2X traffic prediction and highlight the importance of judiciously choosing transmission parameters to optimize system performance.

2.5.2 Real-time Traffic Optimization

Beyond advancements related to autonomous vehicles and V2X communication, real-time traffic optimization relies on a wide range of technologies and strategies. These complementary approaches play a very important role in dynamic traffic management, enabling agile responses to fluctuating road conditions. This section explores how systems based on distributed artificial intelligence and other technologies contribute to more effective traffic management.

Predictive Analysis and Traffic Modeling

The use of advanced predictive models, powered by the analysis of historical and real-time data, allows traffic managers to anticipate flow variations and prepare appropriate responses. These models can predict congestion before it occurs, based on variables such as usual traffic volumes, special events, or seasonal changes. The Internet of Things (IoT), through the deployment of sensors and connected devices, provides continuous data on traffic, weather conditions, and various environmental factors influencing circulation. Once collected, this information feeds predictive models to anticipate congestions and other traffic variations. Edge Computing facilitates data processing close to its source, minimizing latency and speeding up the availability of analyses necessary for reactive traffic management. This proximity allows for quick updates and the application of predictive models, essential for adaptive and real-time traffic management.

Shilpa P. Khedkar and colleagues focused on real-time traffic prediction of IoT using machine learning, deep learning, and time-series forecasting methods

[37]. They compared several prediction models, including Long Short-Term Memory (LSTM), AutoRegressive Integrated Moving Average (ARIMA), and Vector AutoRegressive Moving-Average (VARMA), and Feedforward Neural Networks (FNN), across different traffic intervals and architectural parameters. Their experimental results showed that LSTM and FNN models provide more accurate real-time predictions compared to conventional models like VARMA and ARIMA, based on statistical parameters such as RMSE score, MAE score, and R-squared values. They also plan to design future algorithms capable of considering all dynamic parameters of the IoT environment to predict upcoming traffic in real-time with greater accuracy. Fei Dai and colleagues propose a deep learning spatio-temporal framework for traffic speed forecasting[17]. This framework combines two deep learning models, Convolutional Long Short-Term Memory (ConvLSTM) and Graph Convolutional Network (GCN), to extract the temporal and spatial features of traffic data. Specifically, the ConvLSTM model is used to capture the temporal dynamics of traffic speed data, while the GCN model is utilized to analyze the spatial complexity of the traffic network. Compared to three other deep learning approaches, the framework proposed by the authors demonstrates an ability to grasp the temporal dynamics and spatial complexity of traffic data, resulting in the best performance in terms of traffic speed forecasting.

Adaptive Traffic Management Systems

Adaptive traffic management systems adjust traffic control parameters, such as signal light durations, based on current traffic conditions. These systems rely on sensors and cameras to collect real-time data, thus enabling dynamic traffic regulation that reduces congestion and improves the smoothness of movements. P.G. Balaji and colleagues present a new approach to optimize traffic signal timing[8]. By leveraging the advantages of multi-agent systems and autonomous decision-

making, the authors examined two types of traffic signal control agent architectures implemented on a complex simulated traffic network. In these architectures, traffic signal control agents are software entities capable of making decisions autonomously to regulate traffic flow at intersections. They use real-time traffic data to adjust signal timing and optimize traffic flow. Simulation results showed that traffic signal controllers based on multi-agent systems outperformed currently used conventional signal control methods. This suggests that multi-agent approaches offer a promising solution for optimizing traffic signal timing in complex traffic environments. Damadam et al. explored the application of deep reinforcement learning combined with the IoT for traffic light management systems[18]. The system uses real-time data from sensors and cameras to efficiently manage traffic signals. Simulation results from Shiraz City showed significant improvements in traffic flow and reductions in vehicle queue lengths and waiting times compared to traditional fixed-time traffic signal systems.

Mobile Applications and Data Sharing Platforms

In developing strategies for real-time traffic optimization, navigation mobile applications and data-sharing platforms stand out for their ability to significantly improve the fluidity and safety of urban travel. Among these technologies, Waze and Google Maps emerge as leaders, thanks to their innovative approach to collecting and analyzing traffic data. These applications rely on user participation, who, through their smartphones, continuously feed the database with real-time information on traffic conditions[39][44]. The operation principle of these applications is based on collecting GPS data from user devices on the move. This massive data collection allows for precise mapping of travel speeds on roads, identifying congestion zones, and reporting various incidents such as accidents, roadworks, or road closures. Then, through advanced data processing algorithms and predic-

tive analysis, these applications assess the fastest and least congested routes, thus offering users alternatives to avoid congested areas. Waze, in particular, adopts a biomimicry-inspired approach, similar to the behavior of ants when exploring their environment and communicating with each other to find the most efficient path to a food source. In the context of Waze, each user acts as a "digital ant", sending signals about real-time traffic conditions, which are then analyzed by the application to guide other users. This crowdsourcing method not only allows for dynamic and real-time updates of traffic conditions but also encourages a form of cooperation and collective communication among road users, thus organically optimizing routes in an efficient manner. Thanks to this real-time collaboration and the application of collective intelligence principles, Waze and Google Maps facilitate more proactive traffic management, reducing travel times and contributing to smoother traffic flow. Additionally, these applications play a major role in promoting road safety by alerting drivers to incidents and providing updated information on road conditions. However, dependence on these technologies also raises questions about data privacy and information security. The need for constant internet connection and the accuracy of data in less densely populated areas represent other challenges to consider.

Data Capturing : Traffic Detector Technologies

Advanced detection technologies such as inductive loops, traffic radars, acoustic sensors, and under-pavement weigh-in-motion systems serve as effective tools for collecting accurate data essential for developing reliable and realistic simulation models. These devices are instrumental in capturing dynamic traffic parameters, such as vehicle speeds, traffic volumes, and densities, which feed the simulations to reflect current road conditions.

Inductive Loops Inductive loops measure vehicle presence and movement through changes in the magnetic field induced by the metallic mass of vehicles. This technology is particularly useful for assessing traffic flow and frequency of crossings at intersections, providing essential data for modeling traffic behavior at critical points. Bhaskar et al.[9] explored the use of inductive loops combined with microcontrollers to measure vehicle density and manage traffic signals. Their study demonstrates how data collected from inductive loops embedded in the road surface can be used to dynamically adjust traffic light durations based on real-time traffic conditions. This adaptive approach helps reduce congestion and improve the overall flow of traffic. Similarly, Lamas-Seco et al.[38] introduced a system called SiDIVS, which utilizes inductive loops to detect vehicle signatures through Time-Division Multiplexing, providing accurate vehicle identification and classification for traffic monitoring. These technologies exemplify the potential of inductive loops in enhancing intelligent transportation systems by providing crucial real-time traffic data for adaptive traffic management.

Traffic Radars While often associated with speed monitoring, traffic radars also collect information on vehicle movement. This data is indispensable for simulating average speeds and flow patterns on various road segments, helping to identify potential congestion points. The study by Chetouane et al.[12] explores the use of radar sensors in combination with other technologies to detect and manage traffic congestion. The research discusses how radar data aids in identifying traffic patterns and potential bottlenecks, enabling dynamic adjustments to traffic management strategies. This approach helps in real-time congestion detection and provides actionable insights for traffic planners to mitigate congestion effectively. The integration of radar data with other sensor data forms a robust framework for addressing the challenges of urban traffic congestion.

Under-pavement Weigh-in-motion Systems Under-pavement weigh-in-motion (WIM) systems, which measure the weight of moving vehicles, add an additional dimension to traffic simulations by allowing the assessment of the impact of heavy vehicles on traffic flows and infrastructure wear. This information is indispensable for modeling interactions between different vehicle types and for planning road maintenance. The use of these detection technologies enriches the database used to construct and refine road traffic simulation models, offering a more faithful and nuanced representation of actual traffic conditions. The accuracy and diversity of collected data enable simulations to more precisely simulate the traffic system's responses to various variables, thereby facilitating the search for solutions to improve traffic flow and reduce congestion. Alsaawy et al.[3] proposed an integrated framework for traffic congestion management, combining IoT, fog computing, cloud computing, and data analytics. WIM systems were used in the sensing layer of this framework to provide accurate measurements of traffic status. The data collected was processed to make efficient decisions and stored for further analysis, enhancing the overall effectiveness of traffic simulations and congestion management. Zadobrischi[76] explored the integration of various sensor networks, including WIM systems, for intelligent traffic monitoring. The study highlighted how real-time data from WIM systems can be used to model traffic flow, classify vehicles, and predict congestion. This data is vital for creating dynamic traffic simulation models that can respond to real-time changes in traffic conditions, thereby improving traffic flow and reducing congestion.

In the evolution of road traffic simulation, the adoption of technologies such as distributed artificial intelligence, autonomous vehicles, vehicle-to-everything (V2X) communication, the Internet of Things, and edge computing is essential. These innovations enable real-time data collection and analysis, thus improving the accuracy of traffic simulations and adaptive traffic management. Real-time mod-

eling, supported by data from sensors and road metrology, as well as collaborative navigation platforms like Waze, facilitates congestion prediction and optimizes traffic flow. However, implementing these advanced technologies requires careful management of data privacy and system security challenges.

In conclusion, while emerging technologies offer significant possibilities for improving traffic simulation and management, their effective integration into transportation systems requires ongoing attention to technical and regulatory implications. Continued research in this area is essential for fully realizing the potential of these innovations in creating more efficient and adaptive transport networks.

2.6 Conclusion

This study has undertaken a fairly comprehensive review of multi-agent traffic simulators, addressing their historical development, the challenges they face, and the technological advancements influencing them. It highlights the persistent difficulties and emerging opportunities at the intersection of technology and urban mobility.

Our study reveals that, despite notable progress, multi-agent traffic simulators continue to encounter significant obstacles, particularly in terms of computational efficiency and simulation accuracy. In response to these challenges, we have identified two potential avenues for improvement to enhance the performance and flexibility of these simulators.

Firstly, the HPC-oriented design of existing simulators is considered. This design can operate at the level of parallel implementation or at the simulator modeling level, thus defining an intrinsic parallelism in the algorithm resulting from the model. This approach underscores HPC's capability to address the performance challenges associated with complex simulations.

Secondly, using AI modelling is suggested to enrich the simulators' capacity to model and analyze traffic behaviors. While these avenues are still under development, they present promising prospects for faster, more accurate, and adaptive simulations, meeting the evolving demands of urban transport systems.

In summary, this state-of-the-art review provides an overview of the current field and lays the groundwork for future research. It indicates directions for innovation and development likely to positively influence not only multi-agent traffic simulators but also urban mobility planning and management. Our contribution, by leveraging HPC and AI, aims to participate in the evolution of this rapidly transforming field.

Having explored the current state-of-the-art in traffic simulation, we now turn to our first contribution: the application of the Unite and Conquer method to the MATSim simulator. UC, a technique used in high-performance linear algebra, offering intrinsic multi-level parallelism, heterogeneity, and fault tolerance. The following chapter details how this method can be adapted to enhance the speed and efficiency of traffic simulations, particularly for extensive urban networks.

Chapter 3

Unite and Conquer Approach

3.1 Contextualization and Justification of the Approach

Faced with the challenges posed by multi-agent traffic simulation, particularly due to the considerable volumes of data to be processed and emerging computer architectures, it becomes essential to examine algorithmic concepts adapted to these complex environments. To this end, we are interested in the Unite and Conquer approach[22]; used in high-performance linear algebra to accelerate the convergence of iterative methods. Studies carried out on the application of this approach to the resolution of large linear systems and large eigenvalue problems show[71] great efficiency, particularly when the convergence conditions of classical methods are difficult. This approach is based on the integration and reorganization of traditional iterative numerical algorithms so that they can, in addition to accelerating global convergence, fully exploit the multilevel concurrency, hierarchical memory structures and heterogeneous processing units characteristic of modern computing platforms.

At the heart of the UC approach is the idea of bringing together multiple iterative methods, called co-methods, to accelerate overall convergence toward the solution. Unlike classic iterative methods, which aim to improve the starting condition of an iteration based on the result of the previous iteration, UC calculates this condition in "function" of the results of the previous iteration of all co-methods participating in this collaboration. The definition of this function is a key element of UC methods and allows defining many new methods based on existing iterative ones.

The collaborative strategy between co-methods contrasts with traditional iterative methods that operate in isolation. The communications in (a)synchronous-mode between co-methods orchestrate a joint effort of several algorithms reflecting a synergy that allows more efficient use of available computing resources and offering increased adaptability to the complexities of today's computing systems, whether parallel, distributed or heterogeneous.

Meanwhile, the well-known Divide and Conquer (DC) method is based on a different principle: it divides a problem into smaller sub-problems, solves these sub-problems independently, and then combines their solutions to obtain the overall solution. Although DC is effective in reducing the complexity of problems by fragmenting them, it does not implement active collaboration between the different methods or algorithms working on the sub-problems.

Unlike DC, the UC approach does not simply seek to divide the initial problem into more manageable parts but aims to create a dynamic of collaboration between different co-methods participating in this "competition". In inter-co-method communications, each co-method shares and benefits from the progress made by the others, thus continually improving the quality of the overall solution. This continuous interaction between co-methods not only accelerates convergence but also increases the robustness of the solution in the face of variability and complexity of

the data or models involved. In summary, while Divide and Conquer focuses on structural simplification of problems, Unite and Conquer harnesses the power of interaction and collaboration between different co-methods to optimize the solution process. This fundamental distinction between the two approaches highlights the potential of unified communications to take advantage of advanced IT architectures and address the challenges posed by big technologies.

3.2 Fundamental Principles

As previously mentioned, the fundamental principles of the UC approach revolve around the collaboration between several iterative methods, referred to as co-methods, to solve linear systems of equations and eigenvalue problems in the context of large-scale simulations. This strategy is based on key concepts definining this collaboration that allow for significant optimization of the resolution process, adapted to today's complex computational architectures.

3.2.1 Optimization of Communications

One of the cornerstones of the UC approach from the point of view of the efficiency of parallel calculations; is the optimization of communications between co-methods. In parallel and distributed computing architectures, operations requiring (a)synchronous communications between a large number of cores or nodes can become a significant bottleneck. The UC approach minimizes these operations by favoring, whenever possible, asynchronous communications that can overlap with calculations. This method reduces latency and improves the utilization of computing resources.

3.2.2 Fault Tolerance

The UC brings together several iterative co-methods in order to improve the behavior of the overall collaborative method. The number of these co-methods can be ℓ (with ℓ a number greater than or equal to 1). So, if for any reason one or more of the co-methods disappears, as long as one of the co-methods is active the overall method will work. This important characteristic indicates the fault tolerance of the UC methods.

3.2.3 Diversity of Parallelism and Load Balancing

The co-methods of a UC method can present an intrinsic parallelism of a different nature compared to each other. Thus, a parallel programming model well suited to one does not necessarily suit others. We are therefore in the presence of heterogeneous parallelism between the high granularity tasks corresponding to the co-methods. It is clear that this parallelism added to that intra-co-method indicates that in UC methods a minimum of two levels of parallelism is always present.

Furthermore, a special case of UC methods is when the co-methods are instances of the same iterative method. In other words, co-methods are examples of the same method parameterized differently. In this case, the associated UC method is called MultipleX where X designates the iterative method used. The heterogeneity of CPU in this particular case is essentially expressed by the need for calculation/memory resources of each of the instances requiring more or less powerful nodes/processors.

The diversity of parallelism inherent in UC methods and their adaptability to heterogeneous architectures impose load balancing on the components of these architectures. That is, during implementation, co-methods must be assigned to

the nodes of the parallel machine based on their intrinsic parallelism and their weight. Intra-co-method load balancing with finer-grained parallelism can often be performed dynamically, thus achieving high operational efficiency on heterogeneous systems.

3.2.4 Collaboration and Dynamic Selection

The core of the UC approach lies in the collaboration between co-methods which, at the end of each restart cycle, exchange information to select the best one among those available. This selection is based on predefined criteria, such as minimizing the residue or optimizing convergence. This collaborative strategy allows a significant reduction in the number of cycles needed.

Finally, the fundamental principles of the UC approach represent an interesting advancement in solving large-scale problems on modern computing architectures. By focusing on optimizing asynchronous communications, fault tolerance, load balancing, and efficient collaboration between co-methods, UC methods open new perspectives for enhancing the performance and efficiency of complex numerical simulations, such as those required in large-scale multi-agent traffic simulation.

To conclude, let us present the pseudo-algorithm of the UC method: assume that we need to solve a large linear algebra problem P, such as an eigenvalue problem or a linear system.

Let $L_1, L_2, \ldots, L_l$ be a set of iterative methods, each capable of solving problem P. Let I_i^k be the initial condition (when $k = 0$) and the restart condition (when $k > 0$) for L_i and let S_i^k be the approximate solution obtained by L_i at the end of its k-th iteration/cycle with the initial condition I_i^k (for $i = 1, \ldots, l$). The main steps of this approach to solving P are presented in Algorithm 1.

The restart strategy of the UC algorithm is a key component of the overall algorithm. It involves updating the restart condition I_k^i based on the results obtained

Algorithm 1 Unite and Conquer algorithm

1: **Start**. Choose a starting matrix $[I_1^0, \ldots, I_\ell^0]$, let $k = 0$.
2: **while** for $i = 1, \ldots, \ell$ **do in parallel**
3: Compute S_i^k by L_i with initial condition I_i^k.
4: **if** accuracy of S_i^k is good enough **then**
5: **STOP**
6: **end if**
7: **Share** S_i^k information with all other processes j ($j = 1, \ldots, \ell$ and $j \neq i$).
8: **Restart**. Update starting condition $[I_1^{k+1}, \ldots, I_\ell^{k+1}]$ for restarting by $f(S_1^k, \ldots, S_\ell^k)$ and go to 2.
9: **end while**

$S_k^1, \ldots, S_k^l$ (step 8 of Algorithm 1). When the l processes in step 2 of Algorithm 1 run asynchronously, step 7 (Share) will be part of "Iterate" (step 2) in the same way as step 8 (Restart). In this case, the restart condition for each process is a function of the most recent results available from all the l processes. The success of the approach strongly depends on the quality of the restart information at the beginning of each new process cycle. This information is a combination of the results obtained by all processes in their previous cycle. In other words, defining the function f, which describes this combination at step 8, is of utmost importance for the rapid convergence of the UC algorithm. When the co-methods are instances of the same iterative method, the corresponding UC method is also called $multipleX$, where X is the name of the co-method. Remember, an instance of an iterative method represents the method with a given set of parameters (inputs).

3.3 Examples of UC methods

The case study on the application of the UC approach to solving linear algebra problems illustrates its effectiveness and relevance in the context of large-scale computations. This section focuses on the application of the UC approach to the hybrid LS-Arnoldi/GMRES method, chosen as a representative case study to

demonstrate U&C's ability to improve convergence and computation efficiency on parallel and distributed architectures.

Linear algebra problems, such as solving systems of linear equations and determining eigenvalues, are fundamental in many scientific and engineering fields. The challenge lies in the ability to solve these problems efficiently on modern computing architectures, which are characterized by their large scale and heterogeneity.

3.3.1 Hybrid LS-Arnoldi/GMRES Method

We can consider the LS-Arnoldi/GMRES method[71] representing the most general case of UC methods. This is a method for solving a very large linear system using the least squares (LS), Arnoldi and GMRES co-methods. The role of LS and Arnoldi methods is to help improve the starting conditions of the co-main method GMRES for the resolution of the targeted linear system. We can see from the results presented in[71] how the collaboration between these three co-methods allows good convergence and improve the results of GMRES if it were executed alone. We thus note the effectiveness of the UC approach and its relevance in the context of large-scale calculations.

3.3.2 Multiple Explicitly Arnoldi Method (MERAM)

The MERAM method[23] represents the use of the UC approach for solving a large eigenproblem. It consists of collaboration of several instances of the same ERAM co-method. As in the case of the previous example by consulting [23], we can see the effectiveness of this method in terms of convergence acceleration as well as in terms of calculation performance.

3.4 Unite and Conquer Application to Traffic Simulation

For the simulation of multi-agent traffic, we draw inspiration from the particular case of the UC approach, namely where the co-methods are instances of the same method. Rather than mobilizing several different co-methods treating the same data, we opted for the use of a single method (*multipleX*, as previously named), applied to different datasets. This approach was explored through its applicability to different multi-agent traffic simulators, focusing particularly on MATSim for its process of converging towards an optimal solution, notoriously demanding in terms of computational resources.

Unlike the traditional "Divide and Conquer" method, which segments a problem into independent sub-problems to then consolidate the solutions, UC envisions a dynamic collaboration between different instances of the same algorithm or between different algorithms, enriched by distinct data. This symbiosis allows not only for an acceleration of convergence but also for a more judicious exploitation of computational capacities, suited to the complex structures of contemporary computer systems.

Our contribution thus relies on the application of the UC approach to traffic simulation, by examining how this strategy can be integrated within MATSim. By differentiating the data processed by the same method, we define instances of it constituting our co-methods. This approach aims to explore new ways to optimize the convergence process. Consequently, it reduces the overall computational cost while preserving or enhancing the accuracy of the simulations. This focus underscores the importance of designing traffic simulators that not only address computational complexity but also flexibly adapt to the requirements of cutting-edge computing architectures. The application of UC is concretized here through

the study of MATSim, while emphasizing that this strategy is inherently generalist and therefore applicable to other traffic simulators. The goal is to demonstrate that, although MATSim serves as the primary framework for our exploration, the principles and techniques developed can be transposed and adapted to a variety of simulation platforms.

Having established the benefits of the UC approach in optimizing traffic simulations, we now turn our attention to the broader context of HPC architectures that support such advanced techniques.

Chapter 4

High-Performance Computing for Multi-Agent Simulation

In the era of Big Data and AI, high-performance architectures have become essential components of the technological landscape in research and industry. They provide the computational power necessary to process massive volumes of data, solve complex simulation and analysis problems, and accelerate discovery and innovation.

In scientific research, supercomputers and HPC clusters enable detailed simulations necessary to advance in fields such as particle physics, genomics, weather forecasting, and astrophysics. These architectures facilitate computationally intensive simulations that, would otherwise be prohibitively time-consuming or simply impossible with traditional computers.

In industry, high-performance architectures play a vital role in product design, engineering, and strategic decision-making. For example, in the automotive sector, they are used for conducting virtual crash tests, optimizing aerodynamic designs, and improving vehicles' energy efficiency. In the financial sector, they enable modeling complex economic scenarios, analyzing risks in real time, and performing

high-frequency trading.

The rise of information and communication technologies has also amplified the need for platforms capable of handling continuous data streams. High-performance architectures underpin the data centers that support the Internet's infrastructure, offering services such as cloud computing, web hosting, and video streaming platforms.

The convergence of these architectures with advanced technologies like AI and machine learning opens new possibilities. They not only enhance existing algorithms but also enable the development of new models capable of learning and adapting by exploiting these systems' parallel processing capabilities.

Thus, high-performance architectures are both a driver and a catalyst of technical progress, enabling researchers and professionals to push the boundaries of knowledge and human capacity. They have become indispensable for addressing current and future challenges and will undoubtedly play an even more decisive role in the years to come as data and computation requirements continue to grow.

This chapter aims to explore the foundations of high-performance hardware and software architectures that underlie supercomputers and advanced computing systems.

Effective exploitation of the parallelism of multi-agent simulators requires a good knowledge of the targeted hardware architectures as well as parallel programming models. In this chapter, we first present some key processor kind constituting parallel systems. Namely, three of the main types of common processors are the CPU and the hardware accelerators GPU and FPGA. Some representative parallel systems, constituted by these elements or a combination of these elements, are then presented. These are the Fugaku supercomputer on which we carried out experiments, the Pegasus and Cygnus supercomputers as well as the French Ruche cluster which was also used for our experiments as part of this thesis. We

then present the main basic programming models for the parallel implementation of multi-agent simulators. Indeed, the algorithms resulting from the modeling of these simulators can be seen as a set of autonomous and heterogeneous interacting objects presenting multi-level parallelism. The distribution of these objects to the nodes of parallel machines or to the components of a distributed system is a complex task and often requires assistance in programming these systems. This help is provided by software platforms like YML and Pegasus. Although these platforms are very well suited to the MATSim simulator case study, we did not use them as part of this thesis. This is essentially due to the complexity of intervention for the reorganization of components in the coding of MATSim type software written in Java. However, the description of parallel and/or distributed programming environments of multi-agent simulators for road traffic would be incomplete without this description.

4.1 Parallel Architectural Fundamentals

HPC computing systems are fundamentally constitued by some type of key processors such as Central Processing Units (CPU), Graphics Processing Units (GPU), and Field-Programmable Gate Arrays (FPGA). Each of these components offers specific parallel processing capabilities and contributes to the advancement of research and industry. In this section, we will examine the fundamental characteristics of these processors, starting with CPUs.

Central Processing Units

The CPU is the heart of a computer, responsible for executing programs and managing computing tasks. It is designed to handle a wide range of computational tasks, from simple arithmetic operations to complex decision-making processes.

A CPU's architecture is typically characterized by its ability to perform sequential instruction operations, with each CPU core capable of executing a series of instructions using a set of registers and an Arithmetic Logic Unit (ALU).

However, modern CPUs have evolved to include multiple cores, thereby allowing parallel processing within the same chip. This multicore design enables CPUs to handle multiple processes simultaneously, significantly increasing overall computing performance. Moreover, with technologies such as hyper-threading, a single CPU core can execute multiple computation threads, further enhancing parallel processing efficiency.

CPUs can process a variety of complex instructions, and they can also communicate effectively with other system components, such as memory and input/output devices. Additionally, optimizing algorithms to take advantage of CPUs' multicore architecture has become common practice in high-performance applications, allowing scientists and engineers to achieve unprecedented computational speeds.

In the following sections, we will continue our exploration of architectural fundamentals by examining GPUs and FPGAs, analyzing how these technologies complement CPUs to form the backbone of today's HPC systems.

Graphics Processing Units

GPUs are specialized in fast parallel processing of large amounts of data, making them particularly suited to applications requiring intensive graphical processing and large-scale scientific computations. Unlike CPUs, which are designed for a wide variety of computing tasks, GPUs are optimized for calculations where a large set of operations can be performed in parallel on different data.

A GPU's design is characterized by a SIMD approach, meaning a single instruction can be executed simultaneously on multiple pieces of data. This translates to the ability to perform hundreds of thousands of parallel computations, making

GPUs extremely efficient for graphical rendering applications, numerical simulations, and more recently, deep learning and data analytics.

GPUs also have their own dedicated memory, usually GDDR (Graphics Double Data Rate), designed to support very high data transfer rates between the memory and the processing cores. This high-bandwidth memory is highly pertinent for tasks requiring rapid and repeated data access, such as image processing or matrix calculations.

The use of GPUs has extended beyond graphical applications thanks to programming environments such as CUDA and OpenCL, which allow developers to leverage the computational power of GPUs for general-purpose parallel computing tasks. As a result, GPUs play an increasingly central role in HPC systems, especially in areas where parallelism can be extensively exploited.

In summary, GPUs significantly contribute to the expansion of HPC capabilities, offering a specialized and complementary alternative to CPUs for applications requiring parallel data processing.

Field-Programmable Gate Arrays

Field-Programmable Gate Arrays, or FPGAs, are semiconductor devices based on a matrix of configurable and reprogrammable logic blocks. Unlike CPUs and GPUs, which have fixed architectures and predetermined instruction sets, FPGAs can be programmed to perform specific tasks by reconfiguring their internal structure.

The reprogrammability of FPGAs allows them to adapt to a variety of different computing functions, from simple digital logic to complex signal processing algorithms. This gives them a significant advantage in terms of flexibility, as they can be optimized for maximum performance in specific applications.

FPGAs are also known for their energy efficiency, particularly in cases where

customized parallel processing is required. Since their architecture can be adapted to minimize unnecessary operations, they can often perform tasks with less power than CPUs or GPUs.

In terms of parallel processing, FPGAs offer a fine granularity level that allows developers to create custom parallel architectures. This can be particularly beneficial in applications where latency time is critical, as circuits can be designed to execute tasks in parallel without relying on thread management by an operating system.

A potential downside of FPGAs is that they require specialized expertise in circuit design and hardware description languages such as VHDL or Verilog. Moreover, the development phase can be longer and more costly compared to using CPUs or GPUs, due to the custom nature of FPGA-based solutions.

Despite these challenges, interest in FPGAs as a component of HPC systems continues to grow, especially for workloads that can benefit from dedicated hardware acceleration. By combining the reconfigurability of FPGAs with the processing capabilities of CPUs and GPUs, it is possible to create powerful and versatile systems capable of tackling a wide range of intensive computational tasks.

In conclusion, the selection between CPUs, GPUs, and FPGAs for HPC applications is contingent upon the specific computational requirements and objectives of a given task. Each processor type offers distinct advantages that cater to different aspects of computational workloads. CPUs, with their general-purpose flexibility and capability for complex decision-making tasks, remain the backbone of computing systems for a wide array of applications. GPUs, through their massively parallel architecture, provide an efficient solution for data-intensive tasks that can benefit from parallel processing. FPGAs, offering customizable hardware acceleration, present a unique option for optimizing specific computational algorithms with unparalleled efficiency and adaptability. The strategic integration of these

processing units within computational systems can lead to significant advancements in computing performance, efficiency, and flexibility. As such, the future of HPC will likely see an increased emphasis on hybrid systems that leverage the complementary strengths of CPUs, GPUs, and FPGAs to meet the ever-growing demands of scientific research, data analysis, and technological innovation.

Transition to High-Performance Architectures

The evolving landscape of computational demands, particularly in the domains of scientific research, big data analytics, and AI, necessitates a continuous reevaluation of processing architectures to meet these challenges effectively. The comparative analysis of CPUs, GPUs, and FPGAs underscores the importance of leveraging specific strengths of each processor type to optimize computational performance. This understanding facilitates a natural transition towards high-performance architectures that are not only more capable but also more efficient and adaptable to the complexities of modern computational tasks.

As we pivot towards high-performance architectures, it becomes evident that the future of computing is intrinsically linked to the harmonious integration of these diverse processing units. This integration is driven by the need to accommodate a broad spectrum of computational workloads, from those requiring the flexible, general-purpose computing power of CPUs to tasks that benefit from the parallel processing capabilities of GPUs, and the customizable, hardware-optimized efficiency of FPGAs.

HPC architectures are increasingly characterized by their heterogeneity, incorporating a mix of processing units to harness the optimal balance of speed, efficiency, and precision. Such systems are designed to dynamically allocate tasks to the most suitable processor type, thereby maximizing computational throughput and minimizing energy consumption. This approach not only enhances the

performance of specific applications but also broadens the scope of computational problems that can be tackled effectively.

Furthermore, the advancement of software tools and programming models plays a pivotal role in the accessibility and utilization of high-performance architectures. These tools enable the seamless orchestration of tasks across CPUs, GPUs, and FPGAs, abstracting the complexity of the underlying hardware and allowing researchers and developers to focus on solving computational challenges rather than on hardware intricacies.

In essence, the transition towards high-performance architectures signifies a paradigm shift in computing, moving from a one-size-fits-all approach to a more nuanced, application-specific strategy. This shift is essential for pushing the boundaries of what is computationally feasible, enabling unprecedented advancements in scientific discovery, technological innovation, and the analysis of complex data. As we continue to explore the synergies between CPUs, GPUs, and FPGAs within high-performance architectures, we pave the way for a future where computational limitations are continually redefined and overcome.

4.2 High Performance Architectures

In this section, we present the Fugaku, Pegasus, Cygnus and Ruche systems to illustrate some representative supercomputers of those that can be used as support for effective multi-agent simulations. Note that the Fugaku supercomputer and the Ruche cluster served as support for all our experiments in the context of this thesis.

4.2.1 Fugaku Supercomputer

The Fugaku supercomputer[13], named after an alternative name for Mount Fuji, represents a monumental achievement in the field of HPC. Developed through a

collaborative effort between RIKEN and Fujitsu, Fugaku is situated at the RIKEN Center for Computational Science (R-CCS) in Kobe, Japan. This supercomputer has been meticulously designed to address a wide array of complex scientific, industrial, and societal challenges, marking a significant milestone in computing technology. This text aims to provide a comprehensive overview of Fugaku, detailing its architecture, capabilities, and the implications of its deployment in various research fields.

Architecture and Design

Figure 4.1[25] illustrates the intricate node structure of the Fugaku supercomputer. Each node is composed of multiple Core Memory Groups (CMG), with each CMG directly connected to a segment of High Bandwidth Memory (HBM2). The CMGs are depicted as yellow squares containing a 4x3 grid of smaller squares, each representing an individual core (C) within the processor. These cores are the fundamental units of computation, and their arrangement within the CMG suggests a focus on parallel processing capabilities.

Adjacent to each CMG is a green rectangle labeled HBM2, signifying the high-speed, high-capacity memory designed to keep pace with the computational demands of the cores. The diagram shows a PCIe Controller centrally located between the CMGs, acting as a bridge for peripheral communication. On one side of the PCIe Controller, the Tofu Interface is shown, indicating its role in node-to-node communication within the supercomputer's network.

This node diagram underscores Fugaku's architectural commitment to providing ample memory bandwidth to each core group via HBM2, ensuring rapid data transfer rates that are essential for HPC tasks. The interplay between the CMGs, HBM2, PCIe Controller, and Tofu Interface is carefully orchestrated to deliver a harmonious balance of processing power, memory performance, and network

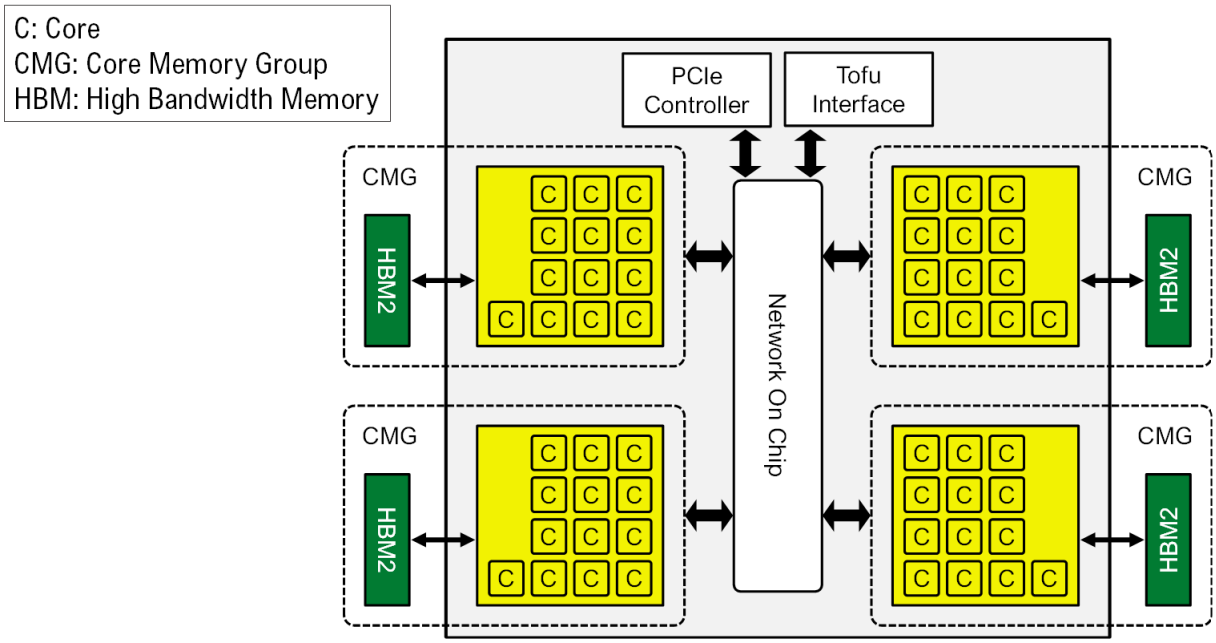

Figure 4.1: Node Configuration of the Fugaku Supercomputer Highlighting Core Memory Groups, High Bandwidth Memory, and Interconnectivity[25]

efficiency.

Integrating this description with the preceding text offers a comprehensive view of Fugaku's architectural prowess. The supercomputer leverages the A64FX CPU's 48-core design and 512-bit vector operations to perform high-throughput computational tasks effectively. The Tofu interconnect D network enhances this capability by providing the necessary high-bandwidth, low-latency communication pathways between over 150,000 nodes. This cohesive design strategy ensures that Fugaku remains adept at handling complex scientific simulations, AI computations, and large-scale data analytics, solidifying its position as a powerhouse in the realm of HPC[65].

Performance and Capabilities

As of June 2024, the Fugaku supercomputer, located at the RIKEN Center for Computational Science in Kobe, Japan, remains one of the most powerful systems globally, ranked fourth on the TOP500 list. It has a High Performance Linpack (HPL) benchmark score of 442.01 petaflops (Rmax) and a Theoretical Peak (Rpeak) of 537.21 petaflops, highlighting its computational speed and efficiency. Fugaku is equipped with 7,630,848 cores, demonstrating its extensive parallel processing capabilities. Initially, Fugaku held the top position from June 2020 to November 2021, indicating its sustained performance over the years. In June 2024, Fugaku's configuration achieved the first rank in the HPCG benchmark with a performance of 16,004.5 teraflops, reaffirming its status as a leading supercomputer[69].

Applications and Impact

Fugaku's deployment is anticipated to revolutionize various research fields, including climate science, disaster modeling, healthcare, and energy. Its computational power enables researchers to perform simulations and analyses at unprecedented scales and resolutions, providing insights that were previously beyond reach. For instance, in healthcare, Fugaku is instrumental in drug discovery and understanding complex biological processes, such as protein folding and virus transmission dynamics.

Moreover, Fugaku plays a pivotal role in disaster prevention and mitigation, utilizing its computational might to simulate natural disasters like earthquakes and tsunamis with high accuracy. This capability is imperative for developing more effective disaster response strategies and minimizing potential impacts on society.

4.2.2 Cygnus Supercomputer

The Cygnus supercomputer at the University of Tsukuba's Center for Computational Sciences, which preceded the Pegasus system, marks a significant advancement in HPC. Officially launched in May 2019, Cygnus was designed to support a wide range of scientific applications, including astrophysics, particle physics, material science, and artificial intelligence. It is notable as the world's first multihybrid accelerated cluster with GPU and FPGA coupling[10].

Technological Framework of Cygnus

Cygnus distinguishes itself through its unique hybrid architecture, combining CPUs, GPUs, and FPGAs. This configuration allows Cygnus to leverage the strengths of each component for different types of computational tasks. It utilizes Mellanox HDR InfiniBand technology to connect these components, ensuring high-speed communication and data transfer across its nodes.

Each node in Cygnus includes high-performance CPUs and GPUs, specifically Nvidia V100 GPUs, and some nodes are also equipped with Intel Stratix 10 FPGAs. This setup enables the system to achieve 30 teraflops of double-precision floating-point performance per node, with a total system performance of 2.4 petaflops for FP64 calculations and 5.12 petaflops for FP32 calculations.

Hybrid Architecture and Performance Optimization

Cygnus employs a hybrid architecture that excels in both coarse-grained and fine-grained parallelism. GPUs in Cygnus provide exceptional performance for SIMD operations, which are common in many scientific simulations. In contrast, FPGAs offer fine-grained parallelism, making them suitable for non-SIMD algorithms such as those used in climate simulations, bioinformatics, and molecular dynamics.

One notable application running on Cygnus is the ARGOT simulation, used for modeling radiation transfer in the early universe. This application benefits from the combined use of GPUs and FPGAs, with each type of processor handling different parts of the simulation based on their computational strengths.

Advanced Networking and Storage Solutions

Cygnus utilizes a high-bandwidth HDR InfiniBand network, with four 100Gbps channels per node configured in a full-bisection fat tree topology. This network architecture minimizes latency and maximizes data throughput, which is crucial for the performance of distributed scientific applications. Additionally, Cygnus is connected to a 2.5 PB Lustre file system for efficient data storage and retrieval, further enhancing its capability to handle large-scale simulations and data-intensive tasks.

Strategic Orientation and Future Implications

Cygnus is strategically designed to support a diverse range of scientific research projects. Its multi-hybrid architecture is a significant step forward from its predecessor, COMA, which relied solely on CPU and GPU technology. By incorporating FPGAs, Cygnus opens new possibilities for accelerating specific algorithms that are not well-suited to traditional CPU or GPU processing.

The innovative use of HDR InfiniBand and the combination of different processing units position Cygnus as a versatile and powerful tool for researchers. This flexibility allows scientists to optimize their applications, whether they require the massive parallelism of GPUs or the specialized capabilities of FPGAs.

4.2.3 Pegasus Supercomputer

The Pegasus supercomputer[11] at the University of Tsukuba's Center for Computational Sciences represents a significant stride in the integration of HPC and AI, encapsulating the cutting-edge in technological synergy. Officially commencing operations on April 1st, 2023, Pegasus, also known as PACS-XI, is architected as a 'Big Memory Supercomputer,' uniquely tailored to address the expansive computational demands of modern scientific research.

Technological Framework of Pegasus

Pegasus distinguishes itself through its world-first combination of Nvidia's H100 Tensor Core GPUs and Intel's 4th Generation Xeon Scalable processors, known as Sapphire Rapids, complemented by the latest Intel Optane persistent memory (PMEM) and Nvidia's NDR200 InfiniBand networking. This fusion of components delivers an unprecedented 6.5 petaflops of theoretical double-precision performance across its 120 compute nodes, each node being a confluence of high-speed processing and expansive memory capacity, required for large-scale simulations and data-intensive AI workloads.

Each node is a marvel of engineering, with an Intel Sapphire Rapids CPU featuring 48 cores at 2.1 GHz, coupled with 256 GB of PMEM and 16 GB of DDR5 memory across eight modules, resulting in 26 TFLOPS of FP64 and 51 TFLOPS of FP64-Tensor performance. The nodes are further empowered by 80 GB of HBM3 memory, which provides a staggering 2 TB/s bandwidth. This robust configuration is designed to facilitate an ad hoc parallel file system, leveraging the node-local storage to circumvent the performance gap typically observed between CPU/GPU processing and storage systems.

Ad hoc Parallel File System and PMEM Utilization

In an innovative approach to file management, Pegasus employs the CHFS, which eschews conventional metadata servers to avoid sequential processing limitations, thus enhancing performance and scalability. Furthermore, Pegasus leverages PMEM to serve as a 'fourth cache' layer behind the DDR memory, harmonizing capacity with speed and obviating the need for extensive reprogramming – a simplicity afforded by mere job script declarations.

HPC and AI Convergence

The advent of Pegasus embodies the convergence of HPC and AI – a bidirectional enrichment where HPC advances AI capabilities ("HPC for AI"), and AI, in turn, streamlines HPC ("AI for HPC"). This symbiosis is meaningful in addressing the growing discrepancy between computational performance and memory capacity. By utilizing PMEM for both extensive memory needs and as part of a high-performance, ad hoc file system, Pegasus facilitates efficient in-situ processing, a method increasingly vital in the era of Big Data.

Strategic Orientation and Future Implications

Pegasus' strategic orientation diverges from its predecessor Cygnus, which employed a combination of GPU and FPGA technology. While Cygnus was focused on performance through the coupling of GPU and FPGA, Pegasus is designed with a different perspective, aiming to expand HPC and AI applications without the strict necessity of MPI parallelism, as evidenced by its generous allocation of 2 terabytes of PMEM per node.

Preliminary testing with an astrophysical simulation code named ARGOT has demonstrated Pegasus' GPU to run 1.86 times faster than Cygnus' V100 GPU,

highlighting its potential to enable much larger simulations in disciplines such as astrophysics, climate science, bioscience, and in the application of AI for drug discovery.

While the foundational hardware capabilities of systems like Pegasus are undeniably critical, it is the accompanying software that truly unlocks and orchestrates these capabilities. Effective software support is essential for harnessing the full potential of high-performance architectures, facilitating groundbreaking research and applications across various scientific domains. The development and optimization of such software form a cornerstone in the advancement of computational sciences.

4.2.4 Ruche HPC Cluster

The Ruche HPC cluster, part of the Fusion/Ruche platform, is a high-performance computing system hosted at the IDRIS site of CNRS. This platform is the result of a collaborative project involving the computing centers of CentraleSupélec, the École Normale Supérieure Paris-Saclay, and the University of Paris-Saclay. It serves a diverse range of scientific and educational purposes within the University of Paris-Saclay community.

Technological Framework of Ruche

Ruche's technological framework includes a variety of high-performance components to support its extensive computing capabilities. The system is built around nodes equipped with Intel Xeon Gold CPUs and multiple types of GPUs, including Nvidia HGX A100, Tesla V100, and Tesla K80. Specifically, it features:

- CPU Xeon Gold 6230 20C @ 2.1GHz
- CPU Xeon Gold 6148 20C @ 2.4GHz

The platform supports parallel code development using MPI, OpenMP, OpenACC, CUDA, and Kokkos, making it ideal for preparing and testing codes before transitioning to larger supercomputers. It also facilitates moderate-sized physical simulations, visualization (using tools like Python and Paraview), continuous integration, and serves as a computational resource for educational purposes.

Advanced Networking and Storage Solutions

Ruche is connected through a high-speed OPA network operating at 100 Gbits/s. The storage technology employed is the Spectrum Scale GPFS parallel file system, offering 380 TiB of usable space and an IO rate of 9 GB/s. Additional technologies used within the platform include Ceph for distributed storage and OpenStack for cloud computing capabilities.

Strategic Orientation and Future Implications

Ruche's infrastructure is strategically designed to meet the evolving computational needs of the scientific community. It supports a wide array of applications and provides redundancy between two main sites: IDRIS, which hosts Fusion/Ruche and LabIA, and Building 206, which hosts the VirtualData cloud. This setup ensures reliable data storage and offers robust computational resources to researchers and educators.

Key Objectives and Benefits

The primary objectives of the Ruche platform include:

- Supporting the development and testing of parallel codes before deployment on larger systems.
- Facilitating moderate-scale physical simulations and data visualization.

- Providing a computational resource for educational purposes and continuous integration.
- Enhancing the capabilities of the University to contribute to national and international collaborative projects.
- Promoting eco-responsible practices through shared infrastructure and optimized resource use.

4.3 Parallel Programming Models and Software Support Frameworks

In this section, we present parallel programming models and some tools to help with parallel and/or distributed programming. However, in simulation algorithms, both models are intrinsically present, and we therefore need to apply a combination of these models. Additionally, these models or their combinations can exist at multiple levels, making the programming of these simulators quite complex. To enable the programmer to focus on their application and offload some aspects of this complex programming, software platforms for parallel programming and execution assistance are available. Among these, we present YML and Pegasus.

4.3.1 Pegasus

The Pegasus Project[19] [21] offers a comprehensive framework for executing workflow-based applications across a range of computing environments, including desktops, campus clusters, computational grids, and cloud infrastructures. Scientific workflows streamline the execution of multi-step computational tasks, such as data retrieval, reformatting, and analysis, and are represented as Directed Acyclic Graphs

(DAGs), where nodes symbolize individual tasks and edges indicate task dependencies. This structure efficiently manages the data flow through tasks that range from simple serial processes to complex parallel computations, supplemented by smaller serial tasks for data preprocessing and analysis.

The Pegasus Workflow Management Service translates high-level workflow descriptions into detailed execution plans on distributed resources. This involves determining the required input data, computing resources, and data transfer needs for workflow execution. This abstraction allows researchers to design workflows conceptually without focusing on the specific details of the execution environment or middleware requirements. Pegasus facilitates integration with existing cyberinfrastructure by efficiently coordinating the use of distributed computing resources.

Pegasus employs several strategies to manage and mitigate errors, including task retries, re-execution of workflows, workflow-level checkpointing, remapping segments of the workflow, sourcing alternative data for staging, and generating rescue workflows to outline pending tasks. Additionally, Pegasus effectively manages storage resources to execute data-intensive workflows on systems with limited storage capacity. It also maintains comprehensive logs of the execution process, including data locations, data utilized and produced, and the specifics of the software and parameters used.

The Pegasus system architecture, as depicted in Figure 4.2[20], consists of several key components and interfaces that interact to facilitate the execution of complex scientific workflows across a variety of distributed computing resources:

- **User Interfaces:**
  - Users interact with Pegasus through various programming languages, including Python, Java, and Perl.
  - Workflow composition tools like Hubzero and Wings offer flexibility in

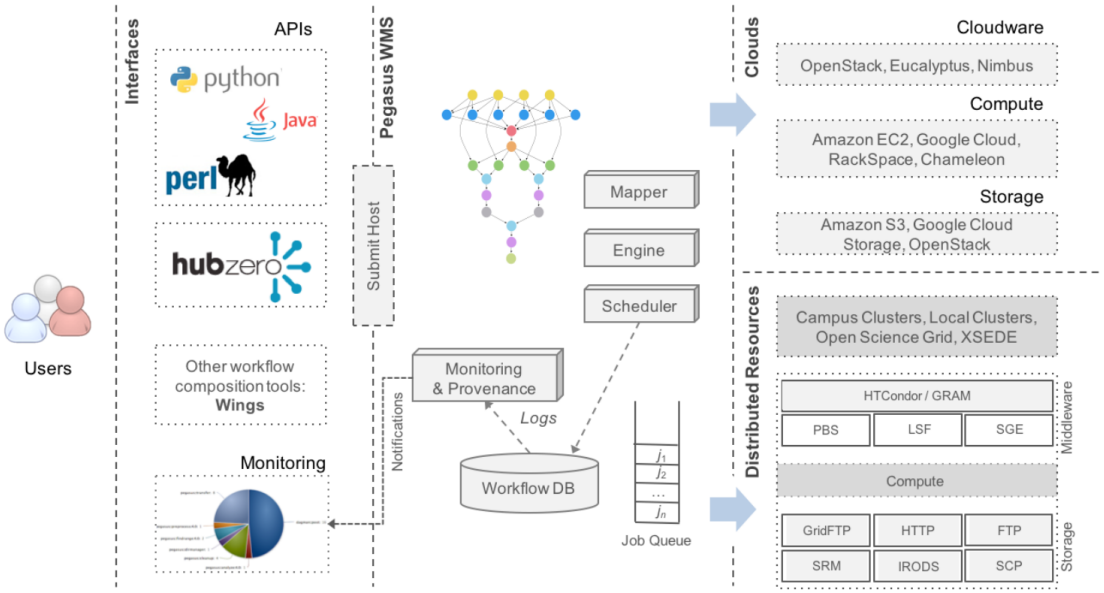

Figure 4.2: Pegasus system architecture[20]

workflow creation.

- **Submission Server (Submit Host):**
  - The submission server is where users prepare and initiate their workflows.
- **Core Components of Pegasus:**
  - **Mapper:** Converts an abstract workflow description into an executable workflow, optimizing by identifying appropriate resources and planning task execution.
  - **Engine:** Orchestrates task execution according to workflow dependencies.
  - **Scheduler:** Schedules tasks on computational resources.
- **Monitoring and Provenance:**

  - The monitoring system collects logs during workflow execution.
  - Collected data is stored in a workflow database to support traceability and post-execution analysis.

- **Clouds:**
  - Pegasus operates on cloud infrastructures such as OpenStack, Eucalyptus, Nimbus, Amazon EC2, Google Cloud, RackSpace, and Chameleon.

- **Storage:**
  - Manages storage across platforms like Amazon S3, Google Cloud Storage, and OpenStack.

- **Distributed Resources:**
  - Utilizes distributed computing resources, including campus clusters, local clusters, Open Science Grid, XSEDE, and job management systems like HTCondor/GRAM, PBS, LSF, and SGE.

- **Middleware:**
  - Employs protocols and distributed file systems for communication and data management, such as GridFTP, HTTP, FTP, SRM, iRODS, and SCP.

In essence, Pegasus provides a platform for executing workflows in varied environments without requiring modifications to the workflow itself. It supports performance, scalability, data management, reliability, and error recovery, serving as a comprehensive solution for managing complex scientific workflows.

4.3.2 YML: A Framework for Global Computing Environments

As part of the broader category of software support for operating high-performance architecture, the YML framework[59] extends the capabilities for creating and executing parallel applications on supercomputers and/or networks of heterogeneous and remote machines. YML is based on a dedicated workflow language known as YvetteML, specifically designed to manage the complexities inherent in large-scale middleware applications. This section delves into YML, outlining its architecture, objectives, and the unique features of the YvetteML language.

Architecture and Design

YML's design abstracts the complexities of underlying middleware, presenting a two-layered approach for application development. The essence of YML's architectural philosophy is encapsulated in its dedicated programming language, YvetteML, which separates the application description into a component definition language and a graph description language. This delineation facilitates the specification of computational tasks and their integration within a complex workflow.

YvetteML employs XML for defining components, providing a standardized mechanism for detailing computational tasks. However, the construction of the workflow graph, a key element for linking components, diverges from pure XML, indicating a novel approach in workflow description within YML.

The YML framework is designed as a comprehensive environment for developing and executing parallel applications on HPC architectures and the networks of heterogeneous and remote machine middleware. The architecture, illustrated on Figure 4.3[52] , is comprised of several components, each serving a distinct purpose within the system.

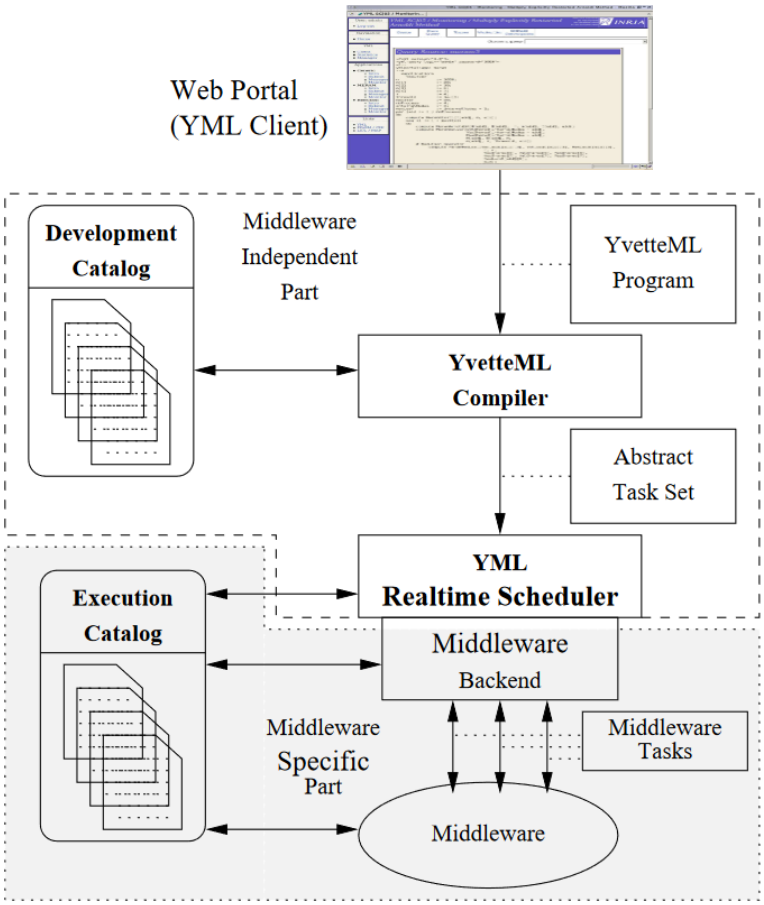

Figure 4.3: YML Framework Architecture[52]

At the user level, the framework is accessed via the Web Portal (YML Client), which provides an interface for interaction with the underlying components. Central to the architecture is the division into middleware-independent and middleware-specific parts, ensuring flexibility and extensibility across different middleware platforms.

The Middleware Independent Part features the Development Catalog, which houses the abstract descriptions of computational components and the graph components. These descriptions are written in the YvetteML language, a dedicated workflow language that enables the design of complex application structures. The

YvetteML Compiler processes YvetteML programs, translating them into an Abstract Task Set ready for scheduling and execution.

On the Middleware Specific Part, the Execution Catalog contains the concrete implementations of the components, tailored to the execution environment. The Realtime Scheduler is responsible for orchestrating the execution flow of the tasks, informed by the compiled abstract set.

Communication between the Middleware Independent Part and the Middleware Specific Part occurs through well-defined interfaces, allowing the YvetteML Compiler to submit tasks to the Realtime Scheduler, which in turn interacts with the Middleware Backend. This Backend is tasked with managing middleware tasks, interfacing directly with the middleware itself to handle execution.

The architecture highlights the separation of concerns, where the Development Catalog and the YvetteML Compiler operate independently of the middleware, while the Execution Catalog and the Realtime Scheduler adapt to the specific requirements and functionalities of the underlying middleware.

Middleware Abstraction

A key goal of YML is to offer a middleware-agnostic framework, enabling the development and execution of applications without dependency on the runtime environment. This objective is realized through a component model that differentiates between generic information applicable across all middleware and middleware-specific details. The development catalog contains universally applicable information, aiding in the broad description of components and their graphical interactions. Concurrently, an execution catalog, tailored to each runtime environment, stores specific component implementations, ensuring versatility across different platforms.

Back-end Integration

Incorporating back-ends specific to each middleware, YML's architecture effectively hides the unique programming and job scheduling nuances from the end-user. This strategy highlights YML's commitment to middleware independence, permitting the seamless transition between runtime environments without the need for application modifications.

Enhancements and Experiments

Acknowledging the necessity for continued development and experimentation, YML identifies areas such as graph compilation and workflow engine scheduling as opportunities for improvement. The full expansion of graphs by the YvetteML compiler prior to execution is noted for its potential to optimize and validate application graphs. Moreover, the framework's adaptability to new scheduling policies signifies an important direction for enhancing YML's scheduling capabilities, with a focus on developing a more sophisticated scheduling infrastructure.

YML offers a structured yet adaptable framework for addressing the diverse requirements of computational tasks in modern global computing environments. Through the YvetteML language, YML provides a versatile platform for the development and execution of parallel applications, free from middleware constraints. As YML advances, it underscores the continuous effort to simplify and enhance the utility of HPC machines and distributed system middleware for the computational community.

Conclusion

Pegasus and YML frameworks provide quite similar functionalities for the management and execution of complex workflows. Pegasus offers a workflow management

system that maps abstract workflow descriptions onto distributed resources, facilitating the execution of multi-step computations across various scientific domains. It emphasizes error recovery and provenance tracking, ensuring that workflows are executed reliably.

YML, on the other hand, abstracts the middleware layer, offering a framework that allows users to construct and execute applications over large-scale distributed systems without the need for in-depth knowledge of the underlying infrastructure. Its dedicated language, YvetteML, supports the definition and integration of computational components within a workflow.

Both frameworks play a role in the evolving ecosystem of software support for HPC architectures. They address the challenges of resource management, scalability, and the efficient execution of data-intensive tasks. By providing tools that abstract lower-level complexities, Pegasus and YML enable researchers and developers to focus on the computational aspects of their work, rather than the intricacies of the execution environment.

As the demands on HPC continue to grow, the importance of frameworks such as Pegasus and YML becomes more pronounced. They contribute to the development of flexible, robust, and scalable solutions necessary for advancing computational research and applications.

4.4 Convergence of HPC and AI

The intersection of HPC and AI represents a pivotal moment in the evolution of computational architectures, heralding a new era of synergistic innovation. This fusion is reshaping the foundational principles of computing systems, fostering the emergence of architectures that are intrinsically tailored to meet the complex demands of AI-driven applications. The ramifications of this convergence extend

beyond mere technical enhancements, signifying a profound shift in the approach to computational challenges and the exploration of scientific inquiries.

AI's integration into HPC environments necessitates a reevaluation of conventional architectural paradigms. Traditional HPC systems, designed for highly structured, numerical computations, are being reimagined to accommodate the stochastic and data-intensive nature of AI algorithms. This transformation is characterized by a pronounced emphasis on parallel processing capabilities, facilitated by the adoption of GPUs and Tensor Processing Units (TPUs). These specialized accelerators excel in executing the parallelizable tasks inherent in machine learning and deep learning processes, offering substantial improvements in computational throughput and efficiency.

Furthermore, the dynamic and often unpredictable computational patterns of AI workloads compel a reconfiguration of HPC architectures towards greater flexibility and adaptability. The advent of heterogeneous computing environments, comprising a mix of CPUs, GPUs, and other accelerators, reflects this shift. These architectures are designed to dynamically allocate resources based on the specific requirements of each task, ensuring optimal performance across a wide spectrum of AI and HPC applications. This adaptability is crucial for supporting the iterative development cycles of AI models, where computational needs can vary significantly over time. Systems like Frontier[53], the world's first exascale computer[69], leverage GPUs alongside CPUs to meet the demands of AI and HPC workloads, demonstrating the critical role of specialized accelerators in enhancing computational throughput. Similarly, the Aurora supercomputer[6] is designed with AI integration in mind, featuring a deep integration of CPUs and GPUs to facilitate a wide range of AI-driven scientific research.

The deep integration of AI into HPC also necessitates advancements in memory and storage technologies. AI applications frequently require rapid access to vast

datasets, prompting innovations in high-bandwidth memory (HBM) and storage solutions that can be closely integrated with processing units. This architectural evolution aims to reduce the latency and bandwidth bottlenecks that can impede the performance of data-intensive AI algorithms, facilitating more efficient data processing and analysis.

On the software front, AI's ascendancy is catalyzing the development of a new generation of programming models, libraries, and frameworks optimized for AI-centric architectures. This software ecosystem is pivotal for unlocking the potential of AI-enhanced HPC systems, providing the tools and abstractions needed to efficiently map AI algorithms to underlying hardware. Moreover, these software advancements are democratizing access to HPC-powered AI, enabling a broader community of researchers and practitioners to leverage these powerful computational resources for groundbreaking work in fields ranging from genomics to climate science.

Energy efficiency has also emerged as a critical consideration in the design of AI-optimized HPC architectures. The intensive computational demands of training and deploying AI models necessitate innovative approaches to power management and cooling. Techniques such as dynamic voltage and frequency scaling (DVFS), alongside more radical approaches like approximate computing, are being explored to mitigate the energy footprint of these systems without compromising computational integrity. The Fugaku supercomputer, powered by ARM technology, exemplifies the potential for less energy-intensive computing without sacrificing performance. ARM's architecture, known for its energy efficiency, offers a sustainable path forward for HPC systems, aligning with the growing concern over the environmental impact of computing operations.

Finally, AI itself is becoming an instrumental tool in optimizing and managing HPC infrastructures. Machine learning models are increasingly deployed to pre-

dict system performance, optimize resource allocation, and enhance the reliability and efficiency of HPC operations. This recursive application of AI underscores the transformative potential of the HPC-AI nexus, not only as a facilitator of advanced computational capabilities but also as a self-enhancing mechanism for the continuous improvement of computing systems.

In essence, the confluence of HPC and AI is engendering a transformative shift in computational architectures, driven by the need to accommodate the unique and evolving demands of AI algorithms. This shift is not merely technical but conceptual, heralding a new paradigm in which computing systems are not only tools for calculation but also platforms for learning and adaptation. As this convergence progresses, it promises to unlock new frontiers in scientific exploration and technological innovation, redefining what is possible in the realm of computation.

4.5 Conclusion

In concluding this section, we have traversed the foundational principles of modern supercomputing, outlined the advanced architectures used in HPC, scrutinized the associated software ecosystems, and discussed the profound implications of integrating AI with HPC. This strategic amalgamation of HPC and AI has catalyzed a notable evolution in both hardware and software design, leading to significant innovations. Systems like Fugaku, Cygnus, and Ruche, with their adoption of cutting-edge technologies, exemplify the shift towards faster, more efficient, and more capable systems to meet contemporary computational demands. Fugaku, in particular, highlights a push towards energy efficiency, demonstrating that peak performance can also be environmentally conscious.

Furthermore, this section underscored the critical role of a robust software ecosystem in leveraging the capabilities of underlying architectures. The sym-

biosis between hardware and software is pivotal for maximizing performance and efficiency in executing complex computational tasks.

Ultimately, the convergence of HPC and AI marks a major milestone in computing, not only pushing the boundaries of hardware capabilities but also redefining approaches to scientific and technical problem-solving. This evolution promises to enable unprecedented scientific breakthroughs and support technological innovations that will have a lasting impact across various fields. In summary, the integration of HPC and AI opens an exciting chapter in the history of technology, heralding advancements that were once deemed beyond reach.

Having reviewed the state-of-the-art in traffic simulation, examined the algorithmic approach, and explored emerging high-performance computing architectures, we now turn to our specific contributions. The following chapter will discuss our efforts to enhance multi-agent traffic simulators, detailing the methods and techniques we have employed.

Chapter 5

Contribution to the Modeling of Multi-agent Traffic Simulators

In this chapter of the thesis, we highlight the specific contributions of our research. Our investigation stands out for two principal contributions, each born from the meticulous exploration and tailored adjustment of methods devised to meet some challenges of the field.

The first contribution concerns the application of the UC approach to algorithms producing multi-agent simulators for road traffic. We will see that the technique resulting from this algorithmic adaptation represents a new variant of the targeted simulator algorithm and provides results that not only enrich understanding, but also strengthen the ability to effectively solve these problems. Such results underline the adequacy and relevance of the proposed methodology in the broader context of simulators having the same characteristics as the one which is the subject of our experiments.

Our second contribution draws upon the strategic deployment of AI techniques to discern patterns from the inputs and outputs of specific simulators. The objective here is to elucidate the behaviors encapsulated by these simulators, thereby

enabling us to replicate their simulations with greater efficiency and less dependency on their traditionally time-intensive operations. This initiative adopts a pragmatic stance, intending to harness the potential of AI to streamline the simulation process, thus sidestepping a commitment to overhaul the foundational use of AI in the domain. The core advantage of this approach lies in its capacity to refine and expedite existing simulation methodologies, paving the way for swifter and more resource-efficient processes.

It is pertinent to mention that for our explorations, we have elected to initially focus on MATSim as our primary target. This choice is informed by several considerations. Firstly, our approaches are designed to be generic; however, MATSim was selected due to its prominent stature within the academic literature, attesting to its widespread acceptance and utility in research. Additionally, as an open-source simulator, MATSim offers unparalleled accessibility and adaptability, enabling researchers to modify and extend its capabilities to suit their specific investigative needs. This openness not only fosters a collaborative and innovative research environment but also ensures that our methodologies, although experimented on MATSim, offer extensibility to many other multi-agent simulators.

5.1 MATSim as a Case Study

As previously articulated, the methodologies developed within the scope of this research are not confined to the specificities of any single simulator. Instead, they are crafted with a versatility that permits their application across various multi-agent road traffic simulator architectures. The adaptability of our approaches underscores a foundational principle of this work: to devise strategies that are as universally applicable as possible, subject only to minor modifications to align with the distinct architectures of different simulators.

In this context, the selection of MATSim as our primary focus serves not only as a practical application of our methods but also as an exemplar through which we can demonstrate their broader applicability. MATSim, with its robust presence in the scientific literature and open-source ethos, represents an ideal platform for this demonstration. It is imperative, therefore, to delve deeper into the functioning of MATSim to elucidate the manner in which our methodologies have been applied and to furnish a comprehensive understanding of the contributions made.

The following, titled "MATSim Overall Functioning" aims to provide a thorough exposition of MATSim's operational framework. Understanding MATSim's comprehensive functionality is crucial for several reasons. First, it allows us to illustrate the specific adjustments and implementations of our proposed methods within the MATSim environment. Second, it facilitates a deeper understanding of how these methods can be extrapolated and adapted to other simulators, highlighting the generalist potential of our contributions. Lastly, this detailed exploration into MATSim will serve to reinforce the relevance of our research, placing them in the broader context of contributing to the advancement of multi-agent road traffic simulation technologies.

By presenting MATSim in this detailed manner, we not only pay homage to its significance as a tool in our research but also lay the groundwork for a discussion on the potential for these methodologies to be adapted and applied in alternative simulation contexts. This exploration into MATSim's overall functioning is thus an essential precursor to a full appreciation of the methods introduced and their impact on the field of road traffic simulation.

5.1.1 MATSim Overall Functioning

MATSim, an open-source framework developed in Java, stands at the forefront of large-scale, agent-based transport simulation. Within this framework, each

agent represents an individual equipped with the capacity to perceive, think, and act within a complex transportation environment. Initially, agents are assigned plans derived from various data sources, including census information, enabling the creation of a synthetic population that accurately mirrors the demographic and behavioral characteristics of real-world populations in the simulated areas.

A distinctive feature of MATSim is its cyclic process: at the end of each simulated 24-hour day, the plans of all agents are meticulously evaluated and scored based on several criteria, such as travel duration, mode of transport selection, and activity sequences. This evaluation reflects how well the plan adheres to the agent's preferences and constraints. Figure 5.1 illustrates this iterative process.

Following this evaluation, a selected group of agents enters the replanning phase[32]. Equipped with insights gained from previous iterations, these agents adjust or revise their initial plans, thereby enriching the spectrum of simulated behaviors. This dynamic ensures continuous improvement and diversification of the strategies employed by the agents.

The framework operates in repetitive cycles, with each iteration representing a 24-hour period. Through this iterative mechanism and the adaptive replanning phase, MATSim gradually progresses towards outcomes that increasingly align with desired objectives. The primary goal is often to reach a state of user equilibrium, where agents can no longer improve their situation by changing their behavior. This state of equilibrium is achieved when the system converges, meaning that the variations in the agents' plans and scores stabilize. Although the number of iterations, defined by the user-set variable max, is at the user's discretion, it is common practice to set max = 300 in many MATSim scenarios. Stop criteria based on convergence measures are suggested in the literature to guide the duration of simulations.

MATSim typically does not include an explicit "stop criterion" for several rea-

sons. The continuous nature of the simulation, designed to simulate the evolution of traffic patterns over time, allows the system to reach an equilibrium state where agents' behavior stabilizes. The number of iterations needed to reach this equilibrium can vary greatly depending on the complexity of the scenario and initial conditions.

Moreover, different scenarios and studies may have varying requirements for when to stop the simulation. By not enforcing a strict stop criterion, MATSim provides researchers with the flexibility to define their own stopping conditions based on their specific study objectives and metrics. Although MATSim does not include an integrated stop criterion, users can implement their own criteria based on metrics such as travel time convergence, iteration stability, or satisfaction of specific performance indicators. This flexibility is crucial for exploratory applications, allowing researchers to understand the system's evolution over extended periods and gain insights into the long-term impacts of various transport policies and changes.

5.1.2 Operational Modules

With a foundational understanding of MATSim's general operation established, it becomes crucial to delve deeper into the framework's core components. MATSim operates around five foundational elements: Input, Execution (Mobsim), Scoring, Replanning, and Output[31]. These components interact in a synergistic cycle, driving the simulation process from initialization to conclusion. Let's explore each module in detail, along with their interactions:

- **Input Module:** This module processes the essential inputs for the simulation, which include the initial plan of each agent—detailing their intended daily activities and travel—and the network file, which outlines the road and public transport networks specific to the simulated area. These inputs are

pivotal in setting the stage for the simulation, providing the framework with a detailed snapshot of the environment and agent intentions.

- **Execution Module (Mobsim):** The heart of the simulation, Mobsim, brings the agents' daily plans to life by simulating their movements and activities within the virtual environment. This module captures the dynamics of travel and activity patterns, modeling how agents interact with the transportation network and each other. The outcomes of this module, including the locations, times, and modes of travel, feed directly into the Scoring Module.

- **Scoring Module:** Following execution, each agent's daily plan is assessed by the Scoring Module. Utilizing a utility function, this module evaluates the performance of agents' plans based on criteria such as movement efficiency and satisfaction derived from activities. The scoring process considers both the cost of travel (e.g., time, discomfort) and the benefits of completed activities, providing a comprehensive measure of plan viability.

- **Replanning Module:** Armed with feedback from the Scoring Module, the Replanning Module offers agents the opportunity to adapt their plans for subsequent iterations. A proportion of agents, typically around 10% (although the user can select the desired rate), are randomly chosen for this replanning process. These agents may alter aspects of their plans, such as departure times, routes, or modes of transportation, in response to previous experiences and traffic conditions. This iterative adjustment process is critical for evolving the simulation towards more efficient and realistic outcomes.

- **Output Module:** The final stage in the cycle, the Output Module, consolidates and presents the results of the simulation. Outputs include detailed plan scores, revised plans for agents undergoing replanning, logs of daily

events, and comprehensive statistics. These outputs are available in both textual and graphical formats, facilitating analysis and interpretation of the simulation results.

An illustrative sequence of MATSim's core modules in operation is presented in Algorithm 2, offering a visual guide to their integrated function.

With the operational elements of MATSim outlined, the focus now shifts to the pivotal Replanning Module. This module's role in shaping the dynamics and outcomes of simulations cannot be understated. Through iterative feedback and adaptation, it enables the simulation to capture the complexity and variability of real-world transport systems. In the following section, we will unpack the intricacies and strategic importance of the Replanning process, highlighting how it facilitates the continuous improvement and realism of the simulated transport scenarios.

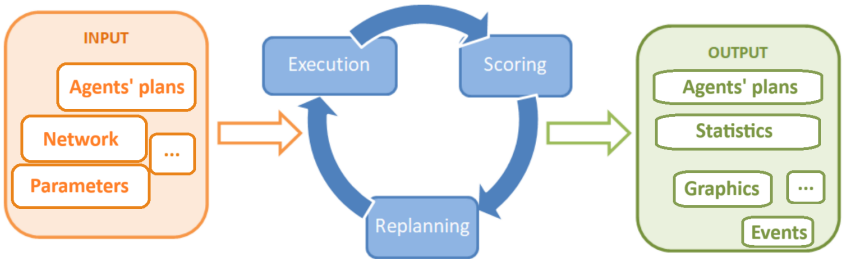

Figure 5.1: MATSim Iterative Loop

5.1.3 Replanning Module in MATSim

The Replanning phase in MATSim plays a pivotal role in the iterative process of transport simulation, offering a mechanism for agents to refine their strategies based on prior experiences. During this phase, a predetermined fraction of the agents, denoted as $R = 0.1M$ (where M represents the total number of agents

within a scenario), are selected for the opportunity to modify their existing plans. This selection ratio is not fixed and can be adjusted by the user to suit specific research needs or simulation objectives.

Each agent is constrained by a maximum number of plans they can hold, indicated by nb_plans. Commonly set to $nb\_plans = 5$, this parameter limits the diversity of strategies an agent can explore but can be tailored to extend or restrict the decision-making horizon of the agents.

As agents proceed through the simulation iterations, plans yielding lower scores are progressively replaced. This ensures that each agent's repertoire of plans evolves, ideally increasing the overall utility of the agent within the simulation environment. New plans are then subjected to the execution and scoring phases, facilitating a dynamic feedback loop that incrementally drives the system towards equilibrium. The goal is to iterate until the agents' plan scores stabilize, indicating a state of system equilibrium where further adjustments cease to yield significant improvements.

MATSim introduces several strategies within the Replanning phase to enable this evolutionary process:

- **Plan Selector Strategy:** *ChangeExpBeta* is a notable strategy that selects plans based on a probabilistic model, specifically $e^{\Delta_{score}}$, where Δ_{score} represents the difference in scores between plans. This method facilitates a preference towards plans that have demonstrated higher utility. For example, if Plan A has a score of 10 and Plan B has a score of 8, the difference in scores Δ_{score} is 2. The probability of selecting Plan A over Plan B is proportional to e^2, which is significantly higher than e^0 (since $e^0 = 1$) for Plan B. This means that the strategy naturally tends to favor the better-performing Plan A, but still allows for the possibility of selecting Plan B, ensuring a balance between exploration and exploitation.

- **Route Innovation Strategy:** This strategy allows for the adjustment of routes within a randomly selected plan based on past traffic conditions. It employs various routing algorithms, such as Dijkstra and A*, to explore alternative paths that might reduce travel time or cost.

- **Time Innovation Strategy:** Adjusting the timing of activities within a plan can significantly impact an agent's daily schedule. This strategy shifts activity end times earlier or later, aiming to find more efficient or preferable schedules.

- **Mode Innovation Strategy:** Exploring different modes of transportation for a given journey can lead to substantial benefits. This strategy changes the transportation mode within a plan, testing the effectiveness of alternatives.

Strategy Weights and Implementation

Each strategy is associated with a relative weight, denoted as $\rho_{PlanSelector}$, $\rho_{reroute}$, ρ_{time}, and ρ_{mode}, which influences the likelihood of its selection during the replanning phase. These weights are crucial for guiding the simulation towards specific objectives, such as minimizing overall travel time or enhancing the use of public transportation.

MATSim normalizes the strategy weights if their sum does not equal one, ensuring that the selection process is balanced. This normalization process is an essential step in maintaining the integrity of the simulation's evolutionary dynamics.

Consequently, starting from a certain distribution of weights among the different replanning strategies, MATSim operates iteratively until it reaches an equilibrium. As previously mentioned, this equilibrium is represented by the convergence of the average performance scores of the daily plans for each agent. This iterative

process ensures that the simulated system evolves towards a stable state where agents' plans consistently meet their objectives and constraints.

The articulation of strategy weights forms the basis for the multiMATSim framework, an extension designed to simulate multiple, interconnected transportation systems. A detailed discussion on multiMATSim and its implementation will be presented subsequently, highlighting its capabilities and the strategic importance of replanning within this broader context.

It is important to note that different distributions of strategy weights can lead to varied results and allow for a broader exploration of solutions. We will now delve into the specific distinctions and implications of these different distributions in the implementations of MATSim.

Having delineated the weight distribution of strategies within the Replanning module, it is imperative to emphasize that this distribution serves as the foundational framework for the implementation of multiMATSim. A detailed exposition of this implementation will be presented in the subsequent section.

5.2 multiMATSim : A Unite and Conquer-based Approach

5.2.1 Description

Addressing the convergence challenges observed in MATSim within complex transportation systems—characterized by extended simulation durations and increased computational demands—this study introduces an approach inspired by the Unite and Conquer methodology, aimed at improving the efficiency of MATSim's convergence process. This approach, termed multiMATSim, constitutes an original contribution of this research.

Algorithm 2 MATSim Algorithm

Initialize: Set max, the total number of iterations, and nb_plans, the maximum number of plans per agent, and M, the total number of agents in the scenario. Initialize strategy probability weights $\rho_{PlanSelector}$, $\rho_{reroute}$, ρ_{time}, ρ_{mode}, ensuring $\sum \rho = 1$. Prepare the simulation environment with agents, the transportation network, and initial plans. Define the fraction R for agent selection in replanning.

Iterate: For iteration 1 to max and for Agent a_k with $k = 1$ to M **do**:

1. Mobsim (Mobility Simulation): Execute the simulation of daily routines for agent a_k, capturing its interactions with the transportation network and the other agents.

2. Scoring: Evaluate the utility of each plan based on their daily activities and mobility, assigning scores.

3. Replanning:

if agent a_k is part of the fraction R selected for replanning **then**

Route: Change their routes.

Time: Alter their departure times.

Mode: Switch transport modes.

PlanSelector: Adopt new strategies for choosing plans.

The changes are influenced by the weighted probabilities ρ, with each strategy impacting plan utility based on previous iterations' experiences.

end if

end For

4. Select New R: At the end of each iteration, a new subset of agents equivalent to the fraction R is randomly chosen for the Replanning phase in the next iteration, ensuring varied and dynamic adaptation across the agent population.

As previously mentioned, convergence in MATSim is achieved when the plan scores for each agent stabilize, indicating a harmonious state in the selection of travel plans. This signifies that agents have identified their optimal paths, adjusted according to their interactions within the transportation system.

The essence of our approach involves running several instances of the core MATSim algorithm simultaneously for each iteration and each agent. The core is considered as a function composed of Mobsim, Scoring, and Replanning. Although MATSim exhibits some intrinsic fine-grained parallelism (akin to multi-threading), our method introduces an additional level of natural parallelism that can be leveraged in conjunction with the existing parallelism in MATSim. These concurrent instances, while aligned in scenarios and parameters, differ in the weights assigned to their replanning strategies. Consequently, starting from identical initial plans, the variation in weights across instances promotes the generation of diverse plans during each replanning phase.

This differential weighting strategy allows for an extensive exploration of possible solutions, capturing a wide array of plan variations. An integral feature of our approach is the periodic communication established between instances at every *step* iterations, facilitating an exchange of plans. Thus, at the end of an iteration, an instance i (where $i \in 1, \ldots, N$ and N is the total number of instances) will be informed of the performance scores from other instances for a given agent a_k (with $k \in 1, \ldots, M$ and M being the total number of agents).

This setup allows instance i to ascertain which instance harbors the plan with the highest score for agent a_k. It then faces the decision to adopt this plan based on a meticulously defined criterion. This criterion examines the performance of two instances, α and β (with α, $\beta \in [1; N]$), for agent a_k. Should the score S^k_α of instance α be less than that of β, S^k_β, and S^k_β surpasses S^k_i for all $i \in [1; N]$, instance α might opt to adopt β's plan for the next iteration.

However, if each instance were to consistently select the highest scoring plan for each agent, it could lead to a scenario where all instances converge on identical plans for every agent. To circumvent this, a selection criterion based on the score dispersion among instances for each agent is implemented. This approach limits plan exchanges to instances that have demonstrated suboptimal performance, thus ensuring a diverse evolutionary trajectory for travel plans.

The dispersion metric indirectly evaluates the efficacy of plans, guiding the strategic exchange of plans. By employing this criterion, the methodology leverages plan performance as a validation tool for plan exchanges, favoring the adoption of superior plans while maintaining a threshold on exchanges to predominantly occur between instances showcasing lower performance outcomes.

Utilizing this selection criterion enables the exploitation of plan performance as an indirect validation mechanism for plan exchanges. It facilitates the adoption of more favorable plans while establishing a threshold for exchanges, ensuring they primarily occur between instances exhibiting lower performance levels.

This selection mechanism operates as follows: consider the vector V_{S^k} of size N, encapsulating the scores from all instances for the agent a_k.

Should the absolute difference between the score S^k_α from instance α and S^k_β exceed the standard deviation of V_{S^k}, the exchange of plans is deemed advantageous, and instance α will incorporate the plan associated with the score S^k_β from instance β.

This initial criterion, focusing on the disparity in scoring between two plans, evaluates suitability based on proximity to the optimal plan. Future research is anticipated to refine this criterion, with the potential integration of machine learning techniques. Drawing inspiration from the Monte Carlo method, subsequent iterations within MATSim will prioritize plans with higher scores, aiming to enhance plan quality. This method represents an initial step, with forthcoming efforts di-

rected towards developing advanced techniques for more effective plan assessment and selection.

The key parameters integral to this approach are summarized as follows:

- N: Determine the number of MATSim instances to run in parallel.
- max: MATSim concludes after a predefined number of iterations, irrespective of convergence achievement.
- M: The number of agents in the current MATSim scenario.
- R: The subset of agents undergoing the Replanning module.
- $step$: Set the iteration interval for inter-instance communication and plan exchange.
- $iter\_processing = \frac{max}{step}$, indicating the frequency of processing and communications.
- Assign varied weights for probabilities associated with Replanning strategies, ensuring $\rho_{PlanSelector} + \rho_{reroute} + \rho_{time} + \rho_{transportmode} = 1$. Each instance i has a distinct distribution of weights.
- Set $nb\_plans = 1$, focusing each instance on optimizing a singular plan.

This framework underpins the multiMATSim initiative, marking a significant advancement in the simulation's strategy for optimizing transportation plans through parallel processing and strategic plan exchanges.

Consider a scenario with just two MATSim instances ($N = 2$) and a single agent ($M = 1$), as depicted in Figure 5.2. Two MATSim instances, labeled i and j, operate on separate computational nodes (x and y respectively). Despite sharing the same scenario, geographic area, and simulated agent, they generate

Algorithm 3 multiMATSim Algorithm

Initialize: Set max, N, M, R, $step$, and strategy probability weights for all N instances, ensuring $\sum_{j=1}^{N} \rho_j = 1$ for each strategy in each instance. Initialize the simulation environment for each instance with agents, transportation network, and initial plans.
Iterate: For each iteration $l = 1$ to max and for Agent a_k with $k = 1$ to M
for each instance $i = 1$ to N **in parallel do**
    Mobsim: Simulate a_k daily activities and movements.
    Scoring: Evaluate a_k's plan.
    if $a_k \in R$ **then**
        Replanning: Adapt a_k's plan using strategies with weights ρ_i.
    end if
end for
if $l \mod step = 0$ **then** ▷ Communication step at defined intervals
    for each instance $i = 1$ to N **do**
        Exchange scores and plans with all other $N - 1$ instances.
        Compute the best score for each agent from among those calculated by the N instances.
        Calculate the standard deviation std_k of the scores obtained for each agent a_k across the N instances.
        for each agent a_k **do**
            Identify the instance β with the best score S_β^k.
            if $|S_\beta^k - S_i^k| > std_k$ **then**
                Instance i adopts the plan associated with S_β^k for agent a_k.
            end if
        end for
    end for
end if

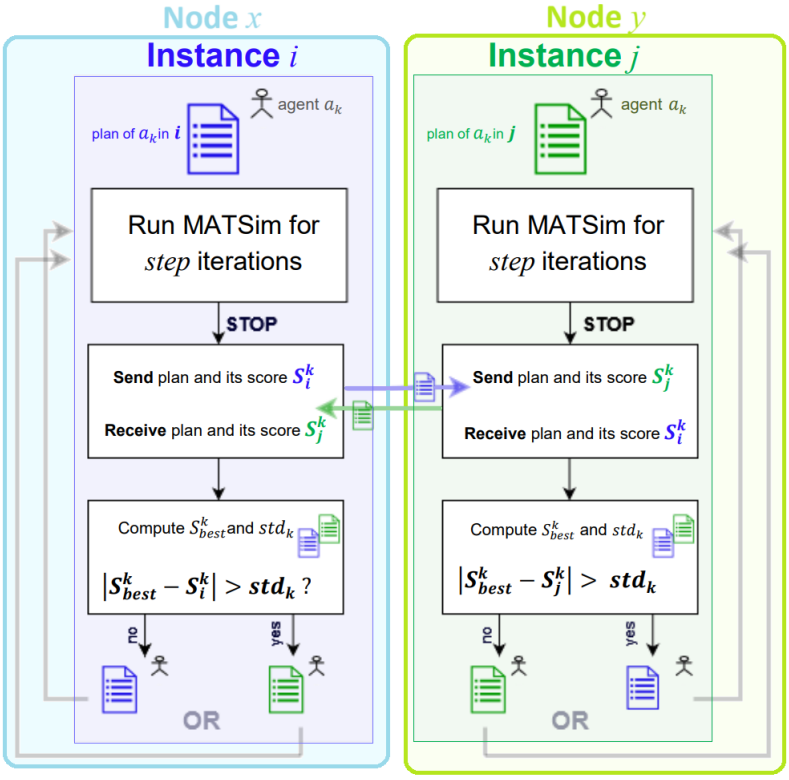

Figure 5.2: Illustration of the multiMATSim algorithm with two ranks / instances and one agent

different plans due to distinct replanning strategies. The elements of Figure 2 are explained as follows:

- **Run MATSim for *step* iterations:** Each instance executes MATSim for a defined number of iterations (*step*), which includes a 24-hour simulation, plan scoring, and replanning to generate optimized plans for the agent.
- **STOP:** Upon completing these iterations, instances i and j temporarily halt their executions.
- **Send and receive plans:** The instances exchange the executed plans and their associated scores for agent a_k. Consequently, i sends its plan and score S_i^k to j, and reciprocally, j sends its plan and score S_j^k to i.

- **Compute S^k_{best} and std_k:** Having received each other's data, instances i and j compute which plan achieved the best score, denoted S^k_{best}, and determine the standard deviation std_k among the scores for agent a_k.

- **Decision Making:** The instances make a decision based on a comparison of the best score S^k_{best} to their own score S^k_i, factoring in the standard deviation std_k. If the absolute difference between their score and the best score is greater than std_k, they adopt the plan associated with S^k_{best}. Otherwise, they retain their original plan.

- These steps are iteratively repeated for a predefined number of MATSim iterations (*step*), with the aim of optimizing the agent's travel plans.

While this example simplifies the scenario to a single agent and two instances, it serves as a foundational model for understanding the process, which can be scaled up to more complex simulations with a larger number of agents and instances.

Finally, the various instances of MATSim in multiMATSim run in parallel, communicating intermediate results until the fixed number of iterations, max, is reached. We did not include a convergence stop criterion initially. At first, we relied on a "visual" inspection of the curve showing the average scores of the daily plans for all agents as a function of the number of iterations.

For reference, MATSim does not have a predefined convergence formula. Several criteria are proposed in the literature [28] [29]. Drawing from these works, we adopted the following criterion: $\forall p > Q, p \leq Q + r, \Delta_p < \epsilon$. This criterion formalizes what we observe visually during simulations: the changes become increasingly minimal with each iteration. In other words, we noticed that the changes in simulation results gradually diminish until they reach an insignificant threshold, and we translated this observation into a mathematical formula.

We also compared the values of ϵ against a MATSim baseline to ensure the

relevance and reliability of our convergence criterion. This formula states that for any iteration p greater than Q, and for a set of successive iterations r such that $p \leq Q + r$, the difference in average agents' plan scores Δ_p between each iteration and the previous one remains less than an arbitrary threshold ϵ.

In other words, if during r successive iterations the value of Δ_p always remains below this threshold, we consider there is convergence in MATSim.

5.2.2 Methodology

In the present study, an examination of multiMATSim is conducted using the Los Angeles 0.1% scenario[15] as the initial framework. The primary aim is to evaluate scalability through the augmentation of instances and, by extension, the number of computational nodes. This evaluation will involve conducting experiments with both $N = 4$ and $N = 8$.

MATSim operates with different datasets, referred to as scenarios, for each city. These scenarios include essential data such as infrastructure, population, and transportation networks. Scenarios for various cities are available on GitHub and can be easily accessed and utilized for simulations.

Several cities have their scenarios prepared for MATSim, including Berlin[14], Zurich[14], New York[16], and Los Angeles. Each scenario contains detailed information about the respective city's transportation infrastructure and population data. These datasets allow for realistic and comprehensive simulations of urban transportation systems.

For this study, we have selected the city of Los Angeles. The Los Angeles scenario is available in different proportions to cater to various levels of simulation complexity and computational demands. The proportions include 0.1%, 1%, 10%, and 100% of the city's population and infrastructure data. Each proportion provides a different level of detail and computational load, making it suitable for

scalability testing and performance evaluation.

In our experiments, we use the Los Angeles 0.1% scenario as the starting point. This choice is strategic for initial scalability evaluations as it offers a manageable dataset size while still providing a realistic simulation environment. The smaller proportion allows for quicker iterations and adjustments, which are crucial in the preliminary phases of the study. As the experiments progress, larger proportions such as 1% or 10% can be utilized to further test and validate the scalability of multiMATSim across an increased number of computational nodes.

By using these different proportions, we aim to systematically evaluate the performance and scalability of multiMATSim under varying computational loads. This approach will help us understand how the system behaves with different dataset sizes and identify potential bottlenecks or areas for optimization. Ultimately, the goal is to ensure that multiMATSim can efficiently handle large-scale simulations, providing valuable insights for transportation planning and policy analysis in urban environments.

Although the current experiments employing 4 and 8 nodes may not yield an exhaustive assessment of scalability, they represent a preliminary exploration and contribute insights that lay the groundwork for a more detailed analysis in subsequent stages of the research.

Furthermore, this study introduces an additional scenario, which also focuses on Los Angeles but incorporates a higher agent count: Los Angeles 1%. This scenario involves 191,649 agents, marking a significant increase, approximately tenfold, from the initial scenario. The objective is to assess the scalability of the approach as the data volume per node increases, ensuring that no unintended artifacts are introduced. This thorough investigation is anticipated to shed light on the performance of multiMATSim under scenarios that present varying computational loads.

Table 5.1: Specifications for a Single Node in Each Supercomputer

Specification	**Ruche**	**Fugaku**
CPU Reference	Intel Xeon Gold 6230	Fujitsu A64FX
CPU Architecture	x86 (Cascade Lake)	Armv8.2-A SVE 512 bit
Total Cores per Node	40 (2 CPUs, 20 cores/CPU)	48 cores (compute) + 2/4 (OS)
Processor Base Frequency	2.10 GHz	Normal: 2 GHz, Boost: 2.2 GHz
Cache	L1: 32 KB per core (instruction) + 32 KB per core (data), L2: 1 MB per core, L3: Up to 27.5 MB (shared)	L1 : 64 KB per core (instruction) + 64 KB per core (data), L2 : 32 MB (8MB per 12-core group), L3: -
SIMD Extensions	AVX-512	SVE (Scalable Vector Extension)

Technical Specifications

Thanks to the Franco-Japanese collaborations established for many years between the Center for Computational Science in Kobe (Japan) and my host team at the Maison de la Simulation (University of Paris Saclay), the experiments were able to be carried out on the Fugaku supercomputer. As a reminder, according to the latest TOP500 ranking in June 2024[69], Fugaku is currently in first position worldwide with the Benchmark HPCG and in 4th position with the Benchmark Linpack. Another hardware support used for our experiments is the Ruche computing cluster[45] at the Paris Saclay mesocenter. Ruche platform is equipped with 232 CPU nodes (9K cores), 18 large memory nodes and 68 GPUs. For more detailed information on Fugaku and Ruche, please refer to Chapter 4.

Please see Table 5.1 for the node-level specifications of each system, including both the cluster and the supercomputer.

Experimental Protocol

This section outlines the experimental protocol utilized in this study, with each simulation instance operating on a dedicated computing node employing the multiMATSim methodology. It is noteworthy to mention that, although multithreading is an intrinsic feature of MATSim, the system was specifically configured to utilize 20 threads, an increase from the standard default of 8 threads. This configuration choice was made notwithstanding the availability of 40 or more cores on each node. The decision to employ only 20 threads was informed by empirical observations indicating enhanced performance on both computational platforms being investigated.

Several theoretical rationales can be proposed to explain this phenomenon. Initially, the reduction in the number of threads might diminish resource conflicts, thereby optimizing the utilization of computational resources. Additionally, this setup may lead to improved cache memory usage, as fewer threads could potentially reduce cache contention among the processing units. Furthermore, a decrease in thread management overhead might also play a crucial role in the improved performance observed, as managing a lesser number of threads demands reduced computational resources. Moreover, reducing the number of threads can also lessen context-switching overhead, thereby decreasing latency and increasing overall system performance. Additionally, fewer threads can lead to more efficient use of memory bandwidth, as there is less competition for memory access, resulting in lower latency and higher throughput. This setup can also minimize the thermal load on the processors, leading to more stable and consistent performance over extended periods. Moreover, by reducing the number of threads, the system may experience fewer synchronization issues, which can often be a source of significant performance bottlenecks in highly parallelized environments. Furthermore, this configuration can enhance the predictability of task execution times, which

is particularly beneficial in real-time computing environments where deterministic performance is crucial. By balancing the number of threads with the hardware's capabilities, the system can better leverage the strengths of modern processors, such as advanced branch prediction and out-of-order execution, leading to more efficient and faster processing. Taken together, these considerations suggest that the selected configuration successfully achieves a balance between task parallelism and the native capabilities of the hardware, illustrating a sophisticated trade-off strategy for maximizing computational efficiency and system performance.

The implementation of a multi-level parallelism strategy, combining distributed instances and multithreading, highlights the methodological sophistication of this research. This approach aims to improve the efficiency and scalability of the simulations by leveraging both task and data parallelism, thus addressing complex computational challenges. Communications between the different computing nodes, each running an instance, are asynchronous, and overlapping these communications with computations further optimizes performance in terms of execution time.

Additionally, to enhance the system's robustness, fault tolerance mechanisms have been introduced. These ensure that if one instance crashes or fails, the others can continue operating normally. That said, there might seem to be a contradiction between fault tolerance and synchronous communications, since synchronous communications are inherently blocking, which can limit flexibility in the event of a failure. However, we have overcome this contradiction through specific implementation strategies that allow fault tolerance to be maintained even with synchronous communications. These strategies ensure that blocking communications do not impair the system's ability to handle failures.

Although the number of instances is currently limited, this fault-tolerant design is crucial for maintaining the system's overall stability and reliability, balancing

the need for synchronous coordination with resilience to failures.

Regarding the operational parameters for multiMATSim, the variables were established as $step = 50$ and $max = 300$. This means that the simulation will undergo a total of 300 iterations, with exchanges taking place every 50 iterations. This systematic methodology not only enables a thorough exploration of the simulation landscape but also ensures a structured approach to data collection and analysis, thereby providing a solid foundation for the empirical investigations conducted in this study.

5.3 AI-based Approach

In this section, we present our AI-based approach to improve multi-agent traffic simulations. This approach constitutes our second major contribution, and we have again chosen to use MATSim due to its flexibility and its capability to simulate complex traffic behaviors at an urban scale. MATSim is also one of the most cited multi-agent traffic simulators in the literature and one of the most used in research, making it an ideal tool for our study.

Our primary objective is to develop a neural network model capable of generating outputs from MATSim inputs. Given the complexity of this task and the variability of MATSim inputs and outputs depending on different geographical areas, we started with a proof of concept. MATSim produces many discrete and continuous variables as outputs, and for the initial phase of our project, we focused on generating a discrete variable.

MATSim uses data specific to each geographical area, making it difficult to generalize models. To manage this complexity, we focused our research on a specific area, Los Angeles, and limited the simulation to 1000 agents. This allowed us to simplify the analysis while maintaining the relevance of the results.

To enhance multi-agent traffic simulations, we explored the application of artificial intelligence techniques to provide a faster and more efficient simulation alternative. The complexity and computational demands of running MATSim are well-documented. In response to these challenges, our study proposes a neural network-based approach to accelerate simulations and obtain results more quickly.

Our goal is to create a predictive model capable of replicating the results of MATSim simulations in a fraction of the time required. We chose to focus on predicting discrete variables used by agents, which are essential for understanding travel dynamics. By using features such as the distance traveled and the occurrences of different transport modes, our model aims to provide accurate and rapid predictions.

We opted for the use of multilayer perceptron (MLP) neural networks due to their capacity to model non-linear relationships and their efficiency in handling high-dimensional input data. This approach allows for the modeling of complex agent interactions while maintaining computational efficiency.

To optimize the performance of our model, we conducted a series of experiments to fine-tune hyperparameters, such as the number of MLP layers, the size of each layer, dropout rates, and loss functions. We compared different loss functions, including Mean Squared Error and Huber loss, to determine the most suitable for our specific data.

Our contribution demonstrates the feasibility of using MLP-based neural networks to accelerate multi-agent traffic simulations. By creating a model capable of quickly predicting one or more discrete variables used by agents, this approach is intended to be generic and applicable to other multi-agent transport simulators, offering a flexible and versatile solution for various traffic simulation applications. This method is a first step and is designed to be extended to predict other variables, both discrete and continuous, and ideally all outputs of MATSim in this

case, as well as for other multi-agent traffic simulators.

5.3.1 Data Preparation

Data Cleaning

We conducted 7,000 MATSim simulations, each involving only 1,000 agents, to obtain the outputs associated with specific inputs for adequately training our neural network. This provided us with 7,000 input/output pairs from simulations with different agents to train and test our neural network. These agents were extracted from the Los Angeles scenario, which comprises less than 200,000 agents, representing 1% of the total population.

In MATSim, input data generally include information such as age, gender, household income, type of housing, and details on daily activities and modes of transport used by the agents. However, given the limited number of data points available for training and our initial focus on a discrete variable (the modes of transport used), we retained only the relevant elements to reduce the complexity of the neural network model.

As previously described, each agent's plan is an input in MATSim, which is adjusted iteratively through replanning to reach an equilibrium among all agents in the scenario. This initial plan already contains different modes of transport used by each agent during the day and the geographical coordinates of each of their activities for the day. Our ultimate goal is to obtain this adjusted plan, as generated by MATSim, and here we start by obtaining the adjusted transport modes vector as the first discrete variable. Therefore, we retained these elements, namely the different modes of transport and the number of times each was used. We calculated the Euclidean distance traveled by the agent during the day using the geographical coordinates of each of their activities, as distance can be a

significant variable for mode choice.

Thus, we retained the following columns: *Distance*, *car*, *pt*, *bike*, *walk*, *ride*, *ride_taxi*, and *ride_school_bus*. From the geographical coordinates of the various activities planned for an agent throughout the day, we calculated a total distance and subsequently normalized it.

In MATSim, an agent can use several modes of transport throughout the day for different activities. For example, an agent might ride a bike to the station to take public transport to work and return the same way. Once back from work, the agent might drive a car to go shopping. This means the agent used public transport twice, rode a bike twice, and drove a car twice. If we represent this information in a vector with the format `[car, pt, bike, walk, ride, ride_taxi, ride_school_bus]`, the vector would be `[2, 2, 2, 0, 0, 0, 0]`.

It is this vector, showing the choice of transport modes for each agent during the simulation, that we aim to predict using the neural network in the initial phase, just as an execution of MATSim would provide this information.

Imagine an agent has this vector in their initial plan. After running MATSim, meaning the execution of their daily plan together with several other hundreds/thousands of agents, it might turn out that, due to very smooth traffic and crowded public transport causing delays (as very used by other agents in the scenario, for instance), the agent ends up using only the car to go to work and does their shopping directly on the way back. We would then have a different vector, which would be `[3, 0, 0, 0, 0, 0, 0]`. This change is what our neural network should be able to capture.

$$\begin{bmatrix} 2 & 2 & 2 & 0 & 0 & 0 & 0 \end{bmatrix} \xrightarrow{\text{MATSim/NN}} \begin{bmatrix} 3 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Initial Plan Adjusted Plan

Data Normalization and Training Unit Preparation

After the data cleaning process, we proceeded to normalize and structure the data for input into the neural network. Given that MATSim simulations involve multiple agents (in this case, 1,000 agents per scenario), it was essential to appropriately prepare these data to accurately reflect the interactions and dependencies among the agents within each scenario.

We began by normalizing the distance data, which represents the total distance traveled by each agent during the day, calculated from the geographic coordinates of their planned activities. This normalization was performed to ensure that all input features are on a comparable scale, which is crucial for the effective training of neural networks.

Next, we organized the data for each scenario, consisting of 1,000 agents and their respective data, into individual training units. This approach enables the neural network to capture the relationships and patterns within the same scenario. By structuring the data in this way, the model can learn from the collective behavior of the agents, which is crucial for accurately predicting the transport modes used by each agent throughout the day.

Since we ran MATSim 7,000 times, we obtained 7,000 sets of input/output data to train and test our neural network. This large dataset helps ensure that the model generalizes well and accurately replicates the behavior observed in the simulations.

The preparation of the data involved not only cleaning and normalizing the input features but also carefully organizing the data into training units that reflect the real-world dynamics of the simulated scenarios. This structured approach is fundamental for training a neural network that can effectively replicate the outputs of MATSim simulations.

5.3.2 Model Architecture

Here, we detail the architecture of the neural network model used to predict the modes of transport for agents in the MATSim simulations. The choice of the model and its configuration are crucial for ensuring optimal performance and good generalization of predictions. We opted for a model based on multilayer perceptrons due to their ability to capture complex relationships in data, particularly the interactions between agents within a scenario. The specific configuration of the model, including layers, units, dropout rates, and hyperparameters such as learning rate and batch size, will also be discussed in detail.

Choice of MLP Model

The MLP model was chosen for this study due to its capacity to model non-linear relationships and complex interactions between agents. Although our input data does not possess a temporal dimension, it consists of interactions between agents within a scenario. The primary reasons for choosing the MLP model in this context are:

- **Capturing Complex Dependencies:** MLP networks are adept at learning intricate dependencies between data points. In our case, the interactions and interdependencies between agents within the same scenario are significant. The MLP model can effectively capture these relationships, which is crucial for accurately predicting transport modes. For example, an agent using a bike to reach the train station and then taking public transport could influence the transport choices of other agents in the same scenario.
- **Flexible Structure:** MLPs offer a flexible and scalable structure that allows the modeling of complex interactions among agents, without the overhead of sequence-based models. This flexibility is useful for capturing the various

dynamics that occur within a 24-hour scenario, where agents evolve together and influence each other's transport choices.

- **Handling Fixed and Variable Input Sizes:** In our study, the data consists of a fixed number of 1,000 agents per scenario. The MLP model is particularly suitable for handling fixed input sizes but can also be adapted to scenarios of different sizes in future studies. This adaptability enhances the robustness of the model when dealing with different datasets or scenarios.

- **Efficient and Compact Architecture:** By using MLPs, we can construct efficient models without the need for excessive parameters, as is often required in complex fully connected networks. MLPs allow us to achieve a balance between model complexity and performance, reducing the risk of overfitting while maintaining accuracy.

- **Comparison with Other Neural Network Types:** While convolutional neural networks (CNNs) are highly effective for structured data like images, they are less suited to modeling interactions between agents in our context. MLPs, on the other hand, are well-equipped to handle the kind of multi-dimensional data present in agent-based scenarios, offering better performance in capturing agent interactions compared to simpler models like logistic regression or more specialized models like CNNs.

- **Robustness to Noise and Data Variations:** MLPs have shown robustness to noisy data and variations, which is crucial in the context of agent behavior, where interactions can be unpredictable and highly variable. This robustness contributes to the reliability of the model when dealing with real-world data.

The choice of the MLP model for this study is motivated by its ability to cap-

ture complex relationships and interactions among agents within a scenario. This approach improves the accuracy of transport mode predictions by considering the mutual influences of agents within the same simulated environment. MLPs offer significant advantages over other types of neural networks in terms of flexibility, robustness, and the ability to model complex interactions effectively.

5.3.3 Model Configuration

Description of the Architecture

The MLP model architecture used in this study consists of multiple layers designed to capture the complex interactions among agents within a scenario. The details of the architecture are as follows:

- **MLP Layers**: The model comprises two fully connected layers, each with 256 neurons. MLPs are well-suited for learning non-linear relationships in high-dimensional data. These layers allow the model to capture complex interactions between agents in the scenario.

- **Dropout**: To prevent overfitting and improve the generalization of the model, a dropout rate of 0.1 is applied after each fully connected layer. Dropout is a regularization technique that randomly disables a fraction of the neurons during training. This helps the model become less reliant on specific neurons, improving its generalization capabilities.

- **Output Layer**: The final layer is a dense layer with a linear activation function. This layer produces the final predictions in the form of continuous vectors, corresponding to the transport modes used by the agents.

The complete model architecture is as follows:

1. First fully connected layer with 256 neurons
2. Dropout with a rate of 0.1
3. Second fully connected layer with 256 neurons
4. Dropout with a rate of 0.1
5. Dense layer with linear activation to predict the output values

Hyperparameters The selected hyperparameters are essential for ensuring optimal convergence and high-quality prediction performance. The key hyperparameters used are:

- **Learning Rate**: A learning rate of 0.001 was chosen with the Adam optimizer. This learning rate facilitates effective weight updates while maintaining stability in the convergence process. Adam was selected for its adaptive learning rate feature, improving the model's convergence speed and stability.
  - **Adam (Adaptive Moment Estimation)**: Adam is a gradient-based optimization algorithm that adapts the learning rate for each parameter by utilizing estimates of the first and second moments of gradients. This leads to faster and more stable convergence, particularly in cases where gradients are sparse or noisy.
- **Batch Size**: A batch size of 64 was chosen, which determines the number of samples used for a single weight update. This size strikes a good balance between training speed and gradient stability. Larger batch sizes could accelerate training but might also increase gradient variability.
- **Loss Function**: We selected the Mean Squared Error (MSE) loss function to measure the error between predicted and actual values, and we also compared the results with the Huber loss function.

  - **Mean Squared Error (MSE)**: MSE calculates the average of the squared differences between predicted and actual values. It penalizes larger errors more heavily, making it useful when the goal is to minimize significant deviations in predictions. However, MSE is sensitive to outliers, as larger errors disproportionately affect the loss value.
  - **Huber Loss**: The Huber loss combines the strengths of both MSE and Mean Absolute Error (MAE). It behaves quadratically for small errors and linearly for larger ones, reducing sensitivity to outliers while maintaining robustness for smaller errors. This makes it particularly useful in datasets with outliers, as it reduces their impact on the model's training process.

  Our experiments showed that the overall performance using MSE and Huber loss was comparable, with similar values for MAE and validation accuracy. Both loss functions produced reliable results, with the choice depending largely on the specific characteristics of the dataset and the need to handle outliers.

- **Callbacks**: Callbacks are used to monitor the training process and adjust certain aspects dynamically.

  - **ReduceLROnPlateau**: This callback monitors the validation loss and reduces the learning rate by a factor of 0.1 if no improvement is observed for 5 consecutive epochs. This ensures that the model can continue fine-tuning even when improvements become infrequent.
  - **EarlyStopping**: EarlyStopping halts training when validation loss fails to improve for 10 consecutive epochs, preventing overfitting. It also ensures the best model performance is saved without excessive training.

Model Training The training process for the MLP model involves several critical steps:

- **Data Preparation**: After cleaning and normalizing the data, agent interaction data is created for each scenario. Each scenario, consisting of 1,000 agents, is treated as a distinct input for the MLP model.

- **Normalization**: Input data, particularly the distances traveled by agents, is normalized to improve model training stability. The mode count data remains unnormalized, as these are discrete values representing the number of times each mode of transport is used.

- **Data Splitting**: The data is split into training and validation sets in an 80/20 ratio, respectively. This split allows for evaluating the model's generalization ability on unseen data. Due to the limited size of our dataset, no separate test set was created.

- **Training**: The model is trained over 50 epochs with a batch size of 64. The ReduceLROnPlateau and EarlyStopping callbacks dynamically adjust the learning rate and stop training when the model reaches its best performance.

By combining this architecture and these hyperparameters, the MLP model is designed to effectively capture the dynamics and interactions among agents within the same scenario to accurately predict the modes of transport used. This configuration enables the model to learn complex relationships while maintaining robustness against overfitting and training data variations.

5.3.4 Loss Functions and Metrics

Use of the MSE Loss Function

Throughout our study, we primarily employed the Mean Squared Error (MSE) loss function to train our model. MSE is a standard choice for regression tasks, as it calculates the average of the squares of the differences between predicted and actual values. This choice was motivated by MSE's ability to penalize larger errors more heavily, which is often advantageous for minimizing significant deviations in predictions.

The formula for MSE is:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{5.1}$$

where y_i represents the actual values, $\hat{y}_i$ the predicted values, and n the total number of predictions.

While MSE provided reliable performance overall, we observed that it is sensitive to outliers. This sensitivity can lead to overfitting, particularly when dealing with counting vectors as outputs, which may contain occasional extreme values.

Comparison with the Huber Loss Function

To address this sensitivity to outliers and evaluate potential improvements in robustness, we also compared the performance of the model using the Huber loss function. The Huber loss function offers a balanced approach by combining the properties of MSE and MAE.

The Huber loss is defined as:

$$L_\delta(y, \hat{y}) = \begin{cases} \frac{1}{2}(y - \hat{y})^2 & \text{for } |y - \hat{y}| \leq \delta, \\ \delta|y - \hat{y}| - \frac{1}{2}\delta^2 & \text{for } |y - \hat{y}| > \delta, \end{cases} \tag{5.2}$$

where δ is a threshold that controls the transition between the quadratic behavior (similar to MSE) for small errors, and the linear behavior (similar to MAE) for larger errors.

The main advantage of the Huber loss is its reduced sensitivity to outliers. For small errors, it behaves like MSE, allowing the model to make precise predictions. For larger errors, it behaves like MAE, mitigating the impact of extreme values.

Comparison Results After applying the Huber loss function, we found that the overall performance metrics were comparable to those achieved with MSE. The choice of loss function ultimately depends on the specific characteristics of the dataset and the importance of handling outliers. For our dataset, where outliers were present but not dominant, both MSE and Huber loss produced similar levels of accuracy, with Huber loss offering slightly better robustness in scenarios involving extreme values.

Other Metrics Used

In addition to the Huber loss function, chosen for its robustness in handling data with outliers, several metrics were employed to evaluate the performance of the model.

Mean Absolute Error (MAE) The MAE is calculated for each mode of transport as well as globally. The MAE measures the average of the absolute differences between the predicted values and the actual values, providing a direct and interpretable measure of the average error.

Mean Squared Error (MSE) The MSE, unlike the MAE, penalizes larger errors by squaring them. Although the Huber loss function is used during training for its robustness against outliers, the MSE remains a useful metric for evaluating the average quadratic error of the predictions.

Coefficient of Determination (R^2) The coefficient of determination assesses the proportion of variance in the data explained by the model. A high R^2 indicates that the model captures the variability of the data well. The formula for R^2 is:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{5.3}$$

where $\bar{y}$ is the mean of the actual values.

Cosine Similarity To evaluate the overall similarity between the prediction vectors and the actual vectors, we used *cosine similarity*. This measure is particularly useful for comparing vectors in high-dimensional spaces, such as the vectors of transport modes. The formula for cosine similarity is:

$$\text{Cosine Similarity} = \frac{\sum_{i=1}^{n} y_i \hat{y}_i}{\sqrt{\sum_{i=1}^{n} y_i^2}\sqrt{\sum_{i=1}^{n} \hat{y}_i^2}} \tag{5.4}$$

This measure ranges from -1 to 1, where 1 indicates that the vectors are perfectly aligned, 0 that they are orthogonal, and -1 that they are opposed.

Using these different metrics allows us to have a comprehensive and nuanced evaluation of our model's performance. By considering average errors, explained variances, and vector similarities, we can identify the strengths and weaknesses of the model from different perspectives, helping us improve our modeling approach.

5.3.5 Optimization and Adjustments

Hyperparameter Tuning

To optimize the performance of our neural network, we conducted a systematic search for hyperparameters. This step is crucial for determining the optimal configuration that will allow the model to generalize well to unseen data. We used the following method for hyperparameter search:

Search Method : Grid Search This method involves systematically testing all possible combinations of predefined hyperparameters. Although this approach is exhaustive and computationally expensive, it allows for a thorough exploration of the hyperparameter space to find the optimal configuration.

Tested Parameters and Combinations We tested various combinations of the following hyperparameters:

Hyperparameter	**Tested Values**
Learning Rate	0.01, 0.015, 0.001, 0.0001
Batch Size	16, 32, 64
Number of MLP Layers	1, 2, 3, 4
Neurons per Layer	128, 256
Dropout Rate	0.1, 0.2, 0.3, 0.4

Table 5.2: Hyperparameters and their tested values

Learning Rate We tested several learning rates to identify the one that allows the model to learn efficiently without diverging or converging too slowly. The values of 0.01 and 0.001 were particularly scrutinized for their impact on model performance.

Batch Size The batch size influences both the stability and speed of convergence. We tested batch sizes of 16, 32, and 64 to evaluate their impact on the model's performance and training stability.

Number of MLP Layers We experimented with architectures ranging from one to four MLP layers to assess their ability to model the complex interactions between agents in the scenarios.

Neurons per Layer The number of neurons in each MLP layer affects the model's capacity to learn complex patterns. We tested configurations with 128 and 256 neurons per layer to evaluate the balance between model complexity and performance.

Dropout Rate For each configuration, we applied a dropout rate to prevent overfitting. We tested rates of 0.1, 0.2, 0.3, and 0.4 to find the best trade-off between reducing overfitting and maintaining useful information for the model.

These various combinations allowed us to identify the optimal model configuration based on performance criteria such as MAE and Cosine Similarity.

The optimal configuration, as described in the previous section on hyperparameters, included a learning rate of 0.001, a batch size of 64, two MLP layers with 256 neurons each, and a dropout rate of 0.1.

Conclusion

In conclusion, the AI-based approach for modeling multi-agent traffic simulations described in this section outlines a methodology utilizing a multilayer perceptron neural network to predict an initial discrete variable, specifically transport modes used by agents. This method aims to reduce computational time and resource usage compared to traditional simulations with MATSim. The architecture of our

MLP neural network, along with its optimized hyperparameters, has been carefully designed to meet the specific requirements of our case study. By focusing on simplified input data and a discrete variable, we have minimized the model's complexity, making this approach accessible and potentially applicable to other multi-agent traffic simulators. This contribution lays the groundwork for future research to further enhance the accuracy and scalability of multi-agent traffic simulations.

While promising, this approach also presents certain challenges and limitations, which will be detailed in Chapter 7, where the results are presented.

Overall, our contribution proposes a pathway to accelerate multi-agent traffic simulations by combining simplicity and efficiency, paving the way for further advancements and complementary research.

Chapter 6

High Performance multiMATSim

In this chapter, we describe in more detail parallel programming models pertinent to the multiMATSim framework. Efficient simulation of such complex systems necessitates leveraging parallel computing paradigms to distribute computational tasks across multiple processors,to reduce execution times and improve the scalability of simulations.

6.1 Parallel Programming Model for multiMATSim

The current implementation of multiMATSim employs a parallel programming model that intricately combines inherent multithreading within MATSim with an additional layer of parallelism through distributed computing. This section elaborates on the multiMATSim architecture, emphasizing the conceptual framework of the model. The focus will be on the principles of synchronous communication via MPI for inter-node coordination and the potential for adopting asynchronous communication to accommodate an increased number of MATSim instances.

Figure 6.1 illustrates the multiMATSim architecture with six computation

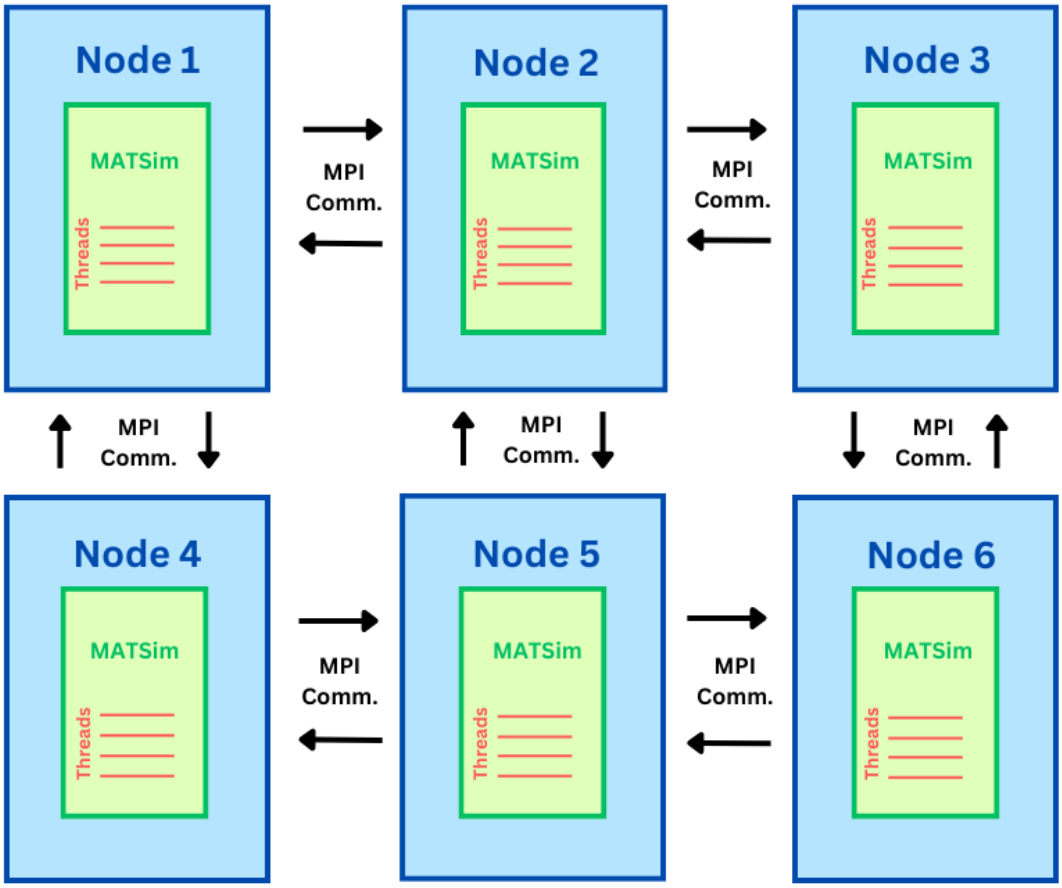

Figure 6.1: Overview of multiMATSim Architecture: Distributed Nodes with Multithreading

nodes, each running an instance of MATSim. Each node is labeled (Node 1 to Node 6) and contains a MATSim instance, which itself utilizes multithreading for various simulation tasks. The arrows between the nodes represent MPI communication for synchronizing the simulation state across all nodes.

6.2 Parallel Implementation of multiMATSim

6.2.1 Shared Memory Computing

QSim, the core mobility simulation engine within MATSim, leverages multithreading to simulate the movements and interactions of agents in parallel. This is crucial for handling large-scale simulations with thousands of agents.

Principle: QSim processes the activities and movements of agents concurrently, utilizing multiple threads to execute these tasks simultaneously. This reduces the overall simulation time by distributing the workload among several threads.

Implementation: Within each MATSim instance (as shown in the diagram), QSim uses threads to manage different aspects of the simulation, such as agent movement, event handling, and interaction processing. Each thread can handle a subset of agents or specific types of events, ensuring efficient utilization of CPU resources.

Additionally, the replanning module in MATSim benefits from multithreading. Replanning involves adjusting agents' plans based on simulation feedback to optimize their travel behavior.

Principle: Replanning applies various strategies to agents' plans in parallel. This includes route replanning, activity rescheduling, and mode choice adjustments.

Implementation: During the replanning phase, each thread can process a separate batch of agents. This parallel processing ensures that multiple replanning strategies are applied simultaneously, significantly speeding up the overall replanning process. In the diagram, this is represented by the threads within each MATSim instance, which handle different replanning tasks concurrently.

The use of shared memory computing can be further enhanced by leveraging multi-core CPUs or GPUs, allowing for more efficient parallel processing and better resource utilization. GPUs, in particular, can execute thousands of threads simultaneously, providing a significant boost to performance for tasks that are highly parallelizable.

6.2.2 Distributed Computing

The MPI communication layer coordinates the simulation across multiple nodes, ensuring that each instance of MATSim operates in sync with the others.

Synchronous Communication: Currently, multiMATSim uses synchronous MPI communication to exchange data between nodes. This ensures that all nodes progress through the simulation steps together, maintaining a consistent state across the distributed system. Given the current setup with a relatively small number of computation nodes, we have chosen to rely on synchronous communications. This approach ensures that we maximize the amount of data available at each synchronization point, allowing us to fully evaluate the impact of the task distribution mechanism and explore the broadest possible range of solutions.

Future Potential: As we plan to scale up and add more instances of MATSim, there is significant potential to switch to asynchronous communication mechanisms. Some instances are more time-consuming than others, leading to idle times while waiting for synchronization. Asynchronous communication would allow nodes to continue processing independently without waiting for slower nodes to catch up, potentially increasing the overall efficiency of the simulation. This shift could result in substantial time savings and improved performance as the system scales.

The distributed computing model in multiMATSim divides the overall simulation workload across multiple computation nodes. Each node runs an independent

instance of MATSim, and these instances work together to simulate complex transportation networks. This division of labor allows multiMATSim to handle larger and more detailed simulations than would be possible on a single node. By distributing the workload, we can leverage the computational power of multiple nodes to achieve high efficiency and scalability in our simulations.

6.2.3 Communications

In distributed simulations like multiMATSim, communication between nodes is critical for maintaining the coherence and accuracy of the simulation. However, it is important to note that data transfer between nodes is often more time-consuming than computational operations themselves. Efficient communication strategies are essential to minimize the overhead and ensure that the simulation runs smoothly.

Importance of Communication: Communication between nodes involves the exchange of critical simulation data, such as the state of agents, transportation plans, and environmental conditions. This data exchange must be timely and accurate to maintain the integrity of the simulation.

Cost of Data Transfer: The cost of transferring data between nodes can be significant. It often outweighs the computational cost of processing the data. Therefore, optimizing communication strategies is essential. Techniques such as minimizing data transfer, compressing data before transmission, and overlapping communication with computation can help mitigate the impact of communication overhead.

In the state of the art, it has been shown that communication in such multi-agent simulators is a significant challenge and constitutes a substantial area of research. Several strategies have been developed to address this problem. For example, asynchronous communication mechanisms can be employed to reduce idle times and improve overall system throughput. Additionally, data compression

techniques can be used to reduce the volume of data transmitted between nodes, thereby decreasing communication time and bandwidth usage.

Furthermore, overlapping communication with computation is another strategy that can be employed. This technique involves scheduling data transfers to occur concurrently with computational tasks, thereby utilizing system resources more efficiently and reducing the overall simulation time.

Research and Strategies: The field of multi-agent simulation has recognized communication as a crucial bottleneck and has seen significant research efforts aimed at improving communication efficiency. Strategies such as adaptive communication protocols, which dynamically adjust the frequency and volume of data exchange based on the simulation state, have been explored. Additionally, hierarchical communication models, where nodes communicate within subgroups before synchronizing with the larger network, have shown promise in reducing communication overhead.

By combining shared memory computing within each node and distributed computing across multiple nodes, multiMATSim achieves a highly efficient and scalable simulation environment. This approach allows for the simulation of complex transportation networks and large numbers of agents, providing valuable insights for transportation planning and policy analysis. Efficient communication strategies ensure that despite the challenges posed by data transfer, the system remains robust and capable of handling extensive simulations.

6.2.4 Transition to Other Potential Models

Exploring alternative parallel programming models presents an opportunity to further optimize multiMATSim simulations. This section introduces potential models that could address existing challenges and enhance computational efficiency.

Spatial Decomposition

Principle Spatial Decomposition involves segmenting the simulated geographic area into smaller sub-regions that can be processed in parallel. This method leverages the inherent spatial locality of transportation simulations, where interactions are often localized within specific areas. By dividing the simulation space, computational tasks become more manageable and can be distributed across multiple processing units, significantly enhancing simulation scalability and efficiency.

Application In the context of MATSim, each computing node is responsible for simulating the movements and interactions of agents within its assigned sub-region. This localized approach allows for more focused and efficient computation, as each node deals with a fraction of the total simulation load. A crucial component of this model is the mechanism for handling agents as they move across the boundaries of sub-regions. This requires sophisticated coordination between nodes to ensure that agent data is accurately transferred and that the simulation remains consistent across the entire geographic space. By implementing spatial decomposition, MATSim can more effectively simulate large-scale transportation networks, making it possible to tackle more complex scenarios with higher fidelity and in less time.

Agent Parallelization

Principle Agent Parallelization involves the distribution of simulation agents across different threads or processes, allowing their simulations to be processed in parallel. This approach leverages the autonomous nature of agents in transportation simulations, where each agent's decisions and movements could potentially be computed independently of others. Parallelizing the processing of agents could thus spread the computational workload across available hardware resources, po-

tentially reducing the time required for complex simulations.

Application Applying this model within multiMATSim could involve utilizing parallel programming models such as OpenMP to facilitate the parallel processing of agents within a single computing node. This would mean structuring the agent processing loop in a way that allows for the concurrent execution of computations related to each agent in separate threads. OpenMP, in particular, offers a convenient and efficient mechanism for achieving this, providing the capability for dynamic allocation of threads based on the workload and the available hardware capacities.

Such an approach would not only aim to speed up the simulation by taking advantage of multi-core processors within a computing node but also enhance the scalability of the simulations with additional hardware resources. Moreover, parallelizing agents could potentially improve the simulation's responsiveness to dynamic changes, as updates to individual agents or small groups of agents could be processed more quickly.

GPU Utilization for Specific Computations

Principle The utilization of GPUs for specific computational tasks capitalizes on the GPUs' ability to execute thousands of threads in parallel. This principle is particularly advantageous for tasks that are highly parallelizable, where the execution workload can be distributed across the vast array of processing cores within a GPU. Such a strategy can significantly accelerate computations, reducing the overall simulation time for complex models.

Application In a hypothetical enhancement of MATSim, leveraging GPUs could be considered for accelerating certain computationally intensive parts of the simulation, such as distance calculations or specific replanning steps. Utilizing pro-

gramming frameworks like CUDA (for NVIDIA GPUs) or OpenCL (for a wide range of GPU architectures) would enable this acceleration.

For instance, the calculation of shortest paths for numerous agents or the evaluation of complex replanning algorithms could benefit from the parallel processing capabilities of GPUs. These tasks, when executed on traditional CPU setups, might constitute bottlenecks due to their computational intensity and the linear increase in processing time with the number of agents. However, by offloading such tasks to GPUs, MATSim could potentially see a reduction in simulation times, allowing for more detailed scenarios or larger-scale simulations to be run within practical time frames.

Tools for Implementing GPU Utilization

Integrating GPU acceleration into MATSim, a Java-based application, presents certain challenges. Direct utilization of CUDA or OpenCL would require interfacing Java with these lower-level programming frameworks, possibly through JNI (Java Native Interface) or leveraging existing libraries designed to bridge Java with GPU computing capabilities. The feasibility and efficiency of such integration would need careful consideration and testing to ensure that the expected performance gains justify the additional complexity in the simulation framework.

Optimization of Data Structures for Parallelism

Principle The principle of optimizing data structures for parallelism involves selecting or designing data structures and algorithms that are inherently suited for parallel computation. This approach aims to minimize the need for locking mechanisms and waiting periods, which can become significant bottlenecks in multi-threaded or distributed environments. Optimized data structures enhance the efficiency and scalability of simulations by allowing more seamless concurrent

access and manipulation of data.

Application In the context of enhancing multiMATSim, this principle could be applied through the implementation of concurrent versions of critical data structures such as agent lists, spatial grids, and transport network graphs. By adopting or developing data structures that inherently support concurrent operations, multiMATSim could achieve significant performance improvements, particularly in areas where high levels of parallel access and updates are required.

- **Agent Lists**: For the dynamic management of agents within the simulation, utilizing concurrent lists or arrays could facilitate efficient, lock-free access and updates to agent states, improving the performance of simulations with a high number of agents.
- **Spatial Grids**: Spatial grids are pivotal for numerous queries and operations within transport simulations, including location-based searches and agent interactions. Concurrent spatial grids, possibly based on quadtree or other spatial partitioning techniques, would allow for more efficient querying and updating of spatial information in a parallelized manner.
- **Transport Network Graphs**: The transport network, represented as a graph, is central to routing and simulation of movements. Employing concurrent graph data structures could enable faster computation of routes and updates to the network state, especially beneficial for simulations involving dynamic changes to the network.

Tools for Implementing Data Structure Optimization

The transition to such optimized data structures may require a careful assessment of the specific needs and bottlenecks within multiMATSim and might involve the

exploration of existing libraries that offer concurrent data structures or the development of custom solutions tailored to the simulation's requirements. Implementing these optimizations would likely necessitate adjustments to the existing architecture and algorithms of multiMATSim, with a focus on maintaining or improving the accuracy and reliability of the simulation outcomes. However, the potential for enhanced performance and scalability presents a compelling case for considering these improvements as part of multiMATSim's ongoing development.

Chapter 7

Experimental Results and Performance Analysis

7.1 multiMATSim

This section delineates a detailed analysis of the performance outcomes derived from the utilization of multiMATSim across two distinct scenarios: a 0.1% and a 1% agent representation of Los Angeles. In an effort to comprehensively evaluate the scalability and efficiency of the multiMATSim approach, both scenarios were subjected to tests utilizing 4 and 8 parallel instances. The ensuing results and performance metrics were primarily run on the Ruche computing cluster, offering insights into the operational efficacy of the multiMATSim methodology under varying computational loads. Some results obtained on Fugaku will be presented and discussed in a subsequent section.

The empirical observations from this investigation are graphically represented in Figures 7.1, 7.2, and 7.6 for the 0.1% scenario, and Figure 7.3 and 7.4 for the 1% scenario. These figures illustrate the evolution of average plan scores executed by all agents within each scenario. The graphical representation utilizes

a black curve to delineate the progression in the MATSim 0.1% and 1% scenarios (serving as the baseline). This black curve specifically represents the performance of MATSim running independently, without any influence from multiMATSim or other configurations. In contrast, the colored curves denote the performance across various instances of multiMATSim.

It is imperative to acknowledge that the graphical illustrations in Figures 7.1, 7.3, 7.6, 7.2, 7.5, and 7.4 reveal square peaks, which are attributed to the stop-and-restart procedures inherent in multiMATSim. These peaks emerge after 80% of the iterations defined by the *step* parameter, symbolizing the average of the historically best scores attained during the preceding *step* iterations for each agent.

It is also important to note that a similar peak or plateau is present on the black curve representing MATSim alone, appearing around the 240th iteration (which corresponds to 80% of the total iterations). This peak, however, does not occur earlier in the process, as MATSim alone lacks the stop-and-restart mechanism employed by multiMATSim. The occurrence of this peak in MATSim alone reflects the natural convergence of the simulation as it approaches its optimal solution.

Despite the presence of these artifacts, the critical focus of the analysis is directed towards the overarching trend in the evolution of average scores. Such a focus provides a more accurate reflection of the continuous improvement and optimization facilitated by the multiMATSim approach, thereby emphasizing the system's adaptability and the effectiveness of the simulation framework in enhancing urban transportation planning and analysis.

The stop-and-restart mechanism in multiMATSim is designed to periodically pause the simulation, allowing for a re-evaluation and exchange of agent behaviors based on the best historical scores. During these pauses, instances may either adopt plans from other instances or continue refining their own plans. This mechanism helps prevent the simulation from getting stuck in local optima and promotes

further exploration of the solution space. As a result, the performance curves of multiMATSim display more frequent adjustments and recalibrations compared to MATSim running independently.

By concentrating on the overall trends rather than the individual peaks, we gain a deeper understanding of the long-term benefits of the multiMATSim framework. The continuous upward trajectory of the average scores highlights the system's capacity to adapt and optimize over time, ultimately leading to more effective and efficient solutions for urban transportation challenges.

7.1.1 Results: Scalability

Los Angeles 0.1% Scenario

The scalability of the multiMATSim approach is assessed through its application to the Los Angeles 0.1% scenario, focusing on the total execution time, speedup, and a comparative evaluation of the 4 and 8 instances configurations.

- **Total Execution Time**: Observations indicate that the total time required to complete the full suite of 300 iterations exhibited negligible differences across the tested configurations. Specifically, the standard MATSim process concluded within 8 hours, whereas the multiMATSim configuration with 4 instances (7.1) required 8 hours and 40 minutes, and the 8 instances configuration (7.2) completed in approximately 8 hours and 45 minutes. Remarkably, the configuration utilizing 8 instances demonstrated a rapid achievement of the convergence score, noticeably soon after the initial data exchange. It is imperative to note that the primary objective is not the reduction of the total execution time per se, but rather the acceleration of convergence time, indicating a more efficient simulation process.

- **Speedup**: Preliminary results had shown that the multiMATSim configuration with 4 instances facilitated a significant speedup in achieving convergence. Specifically, a convergence score of 105 was attained merely 1.5 hours into the simulation (by iteration 52, immediately following the first data exchange), presenting a stark contrast to the standard MATSim, which required 6 hours to reach the same score (yielding a speedup factor of 4.0). The anticipation surrounding the 8 instances configuration's performance was met with an interesting outcome, as the speedup observed was akin to that achieved with the 4 instances setup.

- **Comparison Between 4 and 8 Instances**: Reference to Figure 7.2 reveals a notable surge in the performance of multiMATSim with 8 instances immediately following the first data exchange at iteration 50. Previous investigations highlighted the similar outcomes with the 4 instances configuration. The present analysis substantiates that while the performance with 4 instances remains exemplary, simply doubling the number of instances to 8 does not categorically surpass the former setup in terms of speedup. Nonetheless, the significant performance improvement observed after the initial data exchange at iteration 50 with 8 instances suggests that an earlier data exchange might have further enhanced the speedup, potentially rendering the 8 instances configuration more advantageous than the 4 instances setup.

Los Angeles 1% Scenario

This section presents the scalability assessment of the multiMATSim methodology when applied to the Los Angeles 1% scenario, focusing on total execution time, speedup, the comparative performance of the 4 and 8 instances configurations, and system stability.

Figure 7.1: Average scores of executed plans of multiMATSim LA 0.1% across iterations (with number of instances $N = 4$ and $step = 50$)

- **Total Execution Time**: The inherently more data-intensive 1% scenario exhibited prolonged completion times. Specifically, the baseline MATSim required a substantial 36 hours to finalize 300 iterations. In contrast, the multiMATSim configurations, both with 4 and 8 instances, recorded completion times around 50 hours. Noteworthily, MATSim achieved the convergence score within 29 hours, whereas multiMATSim, under both the 4 and 8 instances configurations, reached this critical benchmark in approximately 6 hours, underscoring the method's efficacy in rapid convergence achievement.

- **Speedup**: In the context of the 1% scenario, a speedup factor of 4.8 was observed, marginally exceeding the 4.0 speedup noted in the 0.1% scenario. This increment underscores the enhanced efficiency of the multiMATSim approach with increasing data density, illustrating its scalability and robustness

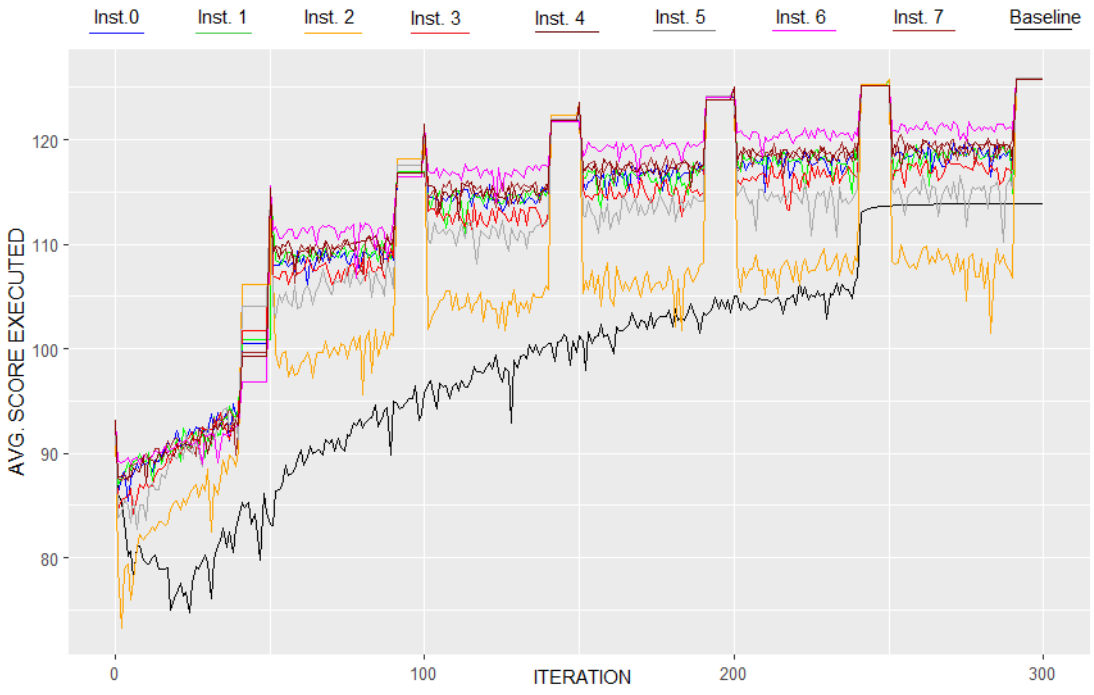

Figure 7.2: Average scores of executed plans of multiMATSim LA 0.1% across iterations (with number of instances $N = 8$ $step = 50$)

in data-intensive environments.

- **Comparison Between 4 and 8 Instances**: Similar to the 0.1% scenario, the performance comparison between the 4 and 8 instances setups in the 1% scenario, as depicted in Figure 7.3 and 7.4, did not reveal a definitive speedup advantage upon scaling from 4 to 8 instances. Nonetheless, a recurring observation was the effect of the data exchange at the 50th iteration, suggesting that an earlier exchange might have fostered faster convergence, particularly with the 8 instances configuration (7.4). This conjecture is supported by the behavior of instance 3, which demonstrated a propensity for swifter convergence, echoing the findings from the 0.1% scenario.

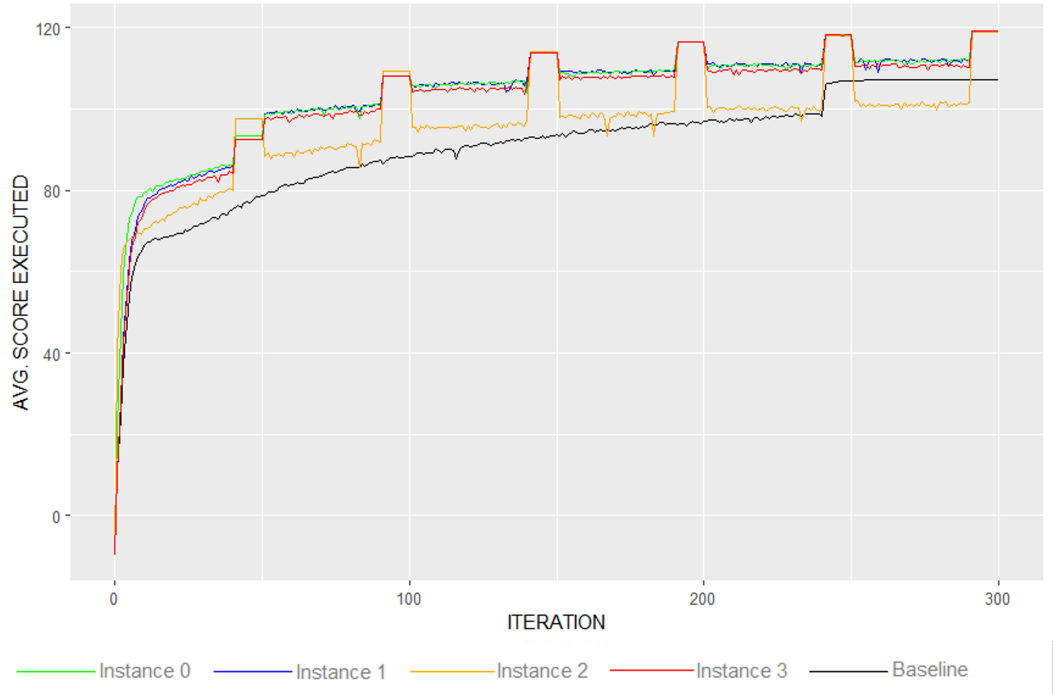

Figure 7.3: Average scores of executed plans of multiMATSim LA 1% across iterations (with number of instances $N = 4$ and $step = 50$)

- **System Stability**: An intriguing observation in the 1% scenario was the noticeable reduction in score oscillations compared to the 0.1% scenario. This trend suggests an enhanced system stability with the increased agent count, pointing towards the multiMATSim method's adaptability and resilience in managing higher volumes of data without compromising on performance integrity.

7.1.2 Discussion: Scalability and Performance Insights

The empirical findings from the application of multiMATSim necessitate a nuanced evaluation of its performance relative to the conventional MATSim framework.

- **Effect of Communication**: A pivotal element of multiMATSim's efficacy is

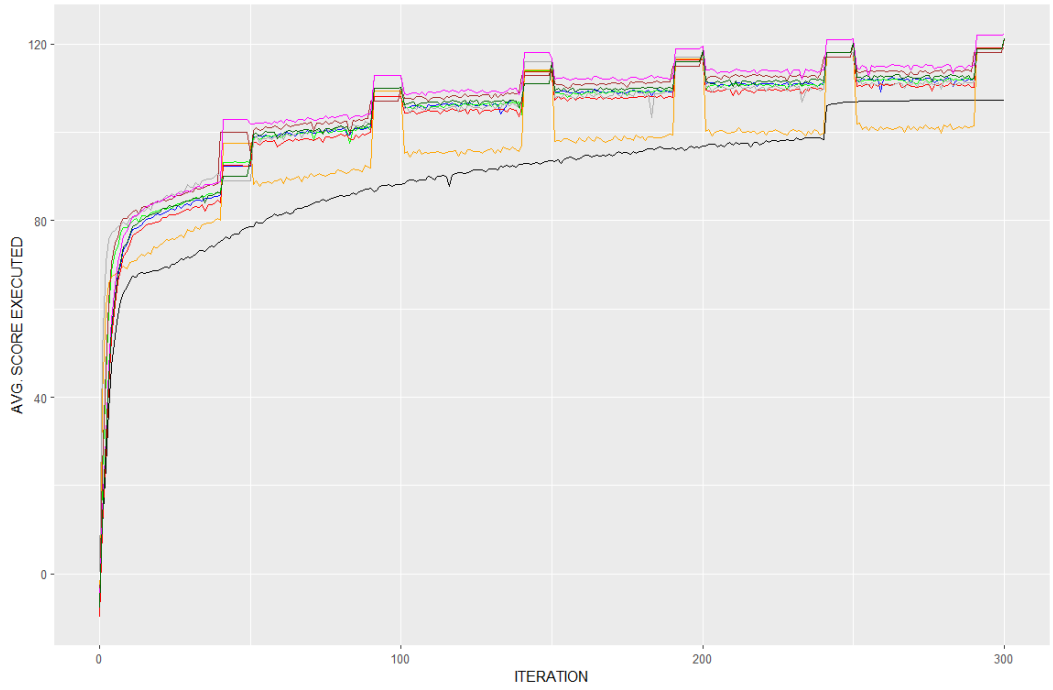

Figure 7.4: Average scores of executed plans of multiMATSim LA 1% across iterations (with number of instances $N = 8$ and $step = 50$)

the strategic dissemination of information amongst instances. The beneficial impact is readily apparent with 4 instances; however, the deployment of 8 instances, while not conclusively outperforming the former in this study, undeniably demonstrates potential. This is particularly evident in the significant score improvements from the initial data exchange. An earlier occurrence of this exchange might have facilitated enhanced scalability with 8 instances, attributable to a more extensive and prompt exploration of the solution space. Future endeavors could prioritize implementing asynchronous communication to reduce idle times and enhance overall system efficiency. This could be complemented by optimizing the granularity, frequency, and timing of information exchanges, possibly through adaptive strategies tailored to the observed system dynamics or targeted agent-specific communications.

- **Trade-off Between Duration and Convergence**: Despite a potential increase in total execution time, a notably faster convergence rate is observed. The ability to process a larger data volume within a single node, as evidenced in the 1% scenario, underscores the scalability of the approach. The proportional increase in resources, from 4 to 8 nodes, alongside the corresponding instances, does not compromise performance, suggesting a promising scope for further scalability enhancement with fine-tuned configurations, particularly for scenarios with a $step < 50$. This hypothesis has led to additional experimentation adjusting the *step* value, with detailed outcomes presented in Section C.

- **Influence of Strategy Parametrization on Execution and Simulation Realism**: The parametrization of replanning strategies significantly affects execution durations and the realism of the simulation. Notably, Instance 2 experienced an execution time up to 1.5 times longer than its counterparts. In environments characterized by synchronous communications, such disparities can impact the overall execution efficiency of multiMATSim.

  In MATSim, up to *nb_plans* (generally 5) are generated for each agent over iterations using replanning strategies and are selected or discarded through a specific mechanism. However, in multiMATSim, we have set *nb_plans* to 1. With this configuration, the *planselector* strategy, which selects the best plan from those available for each agent, has a neutral impact as it only selects one plan without performing complex comparisons between multiple plans.

  In this context, Instance 2 assigns the lowest weight to the *planselector* strategy, allowing more room for other replanning strategies such as *reroute*, *time allocation*, and *mode choice*. As a result, these strategies take on a more

prominent role, which can lead to extended execution times due to the increased complexity of the necessary computations.

Nevertheless, the fact that Instance 2 surpasses sequential execution in terms of convergence, despite a more favorable configuration for execution speed in the latter, demonstrates that the multiMATSim strategy outperforms MATSim even under conditions where MATSim would theoretically be advantaged. This observation indicates that it is possible to optimize overall performance by strategically reconfiguring the weights assigned to different strategies. Moreover, a balanced distribution of weights among replanning strategies contributes to increased realism in the simulation by allowing for a diversity of plans, adequate responsiveness to changes, and better consideration of individual agent preferences.

- **Stark Difference in Total Execution Times**: The observed discrepancies in total execution times between the MATSim and multiMATSim configurations, both for the 0.1% and 1% scenarios, warrant further examination. These disparities could stem from the influence of replanning strategy weights, the increased agent count, and the intensive processing of XML files in the 1% scenario. A recalibration of data exchange frequencies might yield improvements in data-dense scenarios, optimizing the overall computational efficiency and effectiveness of the multiMATSim framework.

  Furthermore, as previously mentioned, implementing asynchronous communications should address this issue by reducing idle times and enhancing overall system performance.

7.1.3 Results and Influence of *step* Value Variation

The initial implementation of multiMATSim with a *step* value of 50, for both $N = 4$ and $N = 8$, yielded promising outcomes. However, the enhancements observed with $N = 8$ were relatively modest when juxtaposed against the results for $N = 4$, despite a doubling of the computational resources. This discrepancy prompted a reevaluation of the *step* value, with the hypothesis that more frequent data exchanges could potentially amplify the benefits of utilizing a greater number of instances.

Subsequent experiments were therefore conducted with a reduced *step* value of 25, while employing $N = 8$, to facilitate data exchanges every 25 iterations among the 8 instances of MATSim. Figure 7.5 illustrates the average plan scores achieved in the multiMATSim framework for the Los Angeles 0.1% scenario with $N = 8$ and $step = 25$. Figure 7.6 presents analogous results but for $N = 4$. Although the graphical outcomes for the Los Angeles 1% scenario are omitted here, similar patterns were observed across both scenarios.

Configuration: $step = 25$**; Instances:** $N = 8$ (Figure 7.5):

- Certain exchanges, particularly those originating from Instance 6, not only achieved but, in some instances, exceeded the benchmark convergence score of 105 immediately following the first exchange of information.
- Several instances necessitated additional iterations beyond the first exchange to meet or surpass this benchmark score.
- The convergence score of 105 was consistently reached within approximately 45 minutes.

Encouraged by the outcomes of this configuration, an evaluation of the impact of a *step* value of 25 while maintaining $N = 4$ was undertaken.

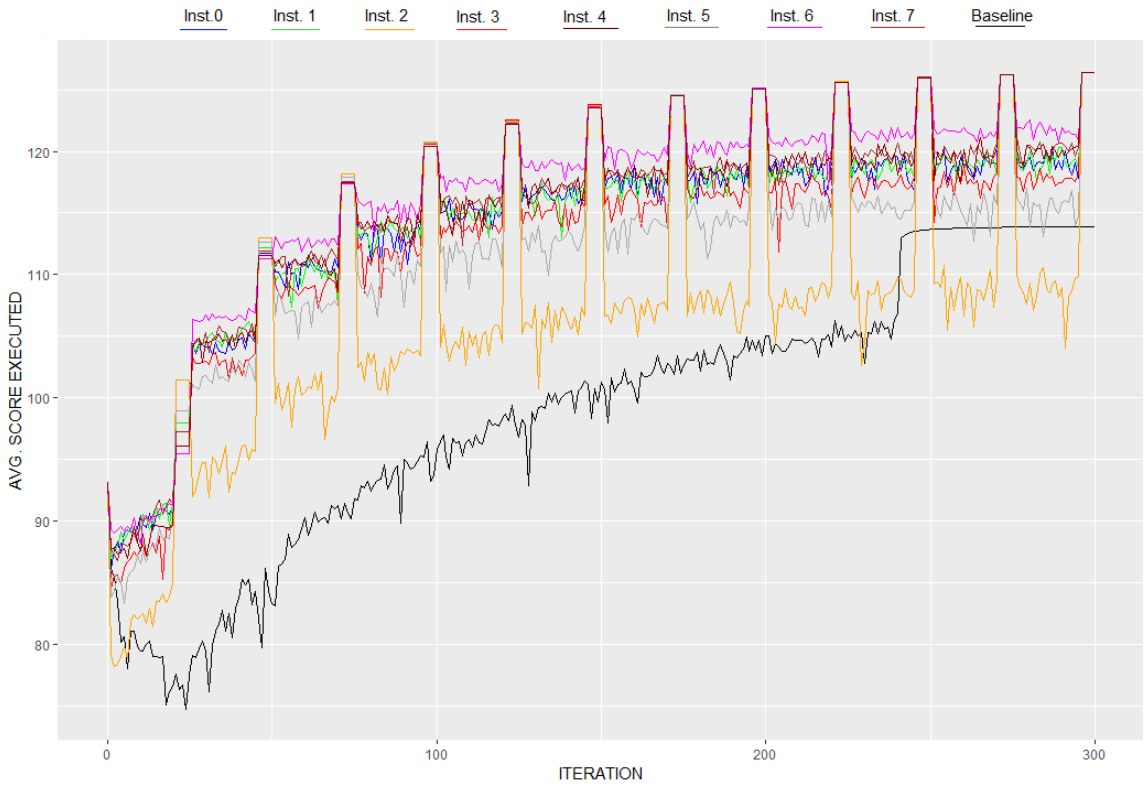

Figure 7.5: Average scores of executed plans of multiMATSim LA 0.1% across iterations (with number of instances $N = 8$ and $step = 25$)

Configuration: $step = 25$**; Instances:** $N = 4$ (Figure 7.6):

- No immediate improvement was discernible subsequent to the initial exchange.
- Notwithstanding, marked performance enhancements were observed following the second exchange.

Observation Summary: These findings highlight the efficacy of the multiMATSim methodology in accelerating convergence relative to the baseline. The cadence of data exchanges significantly influences overall performance. Moreover, the incremental execution time associated with an increased number of instances

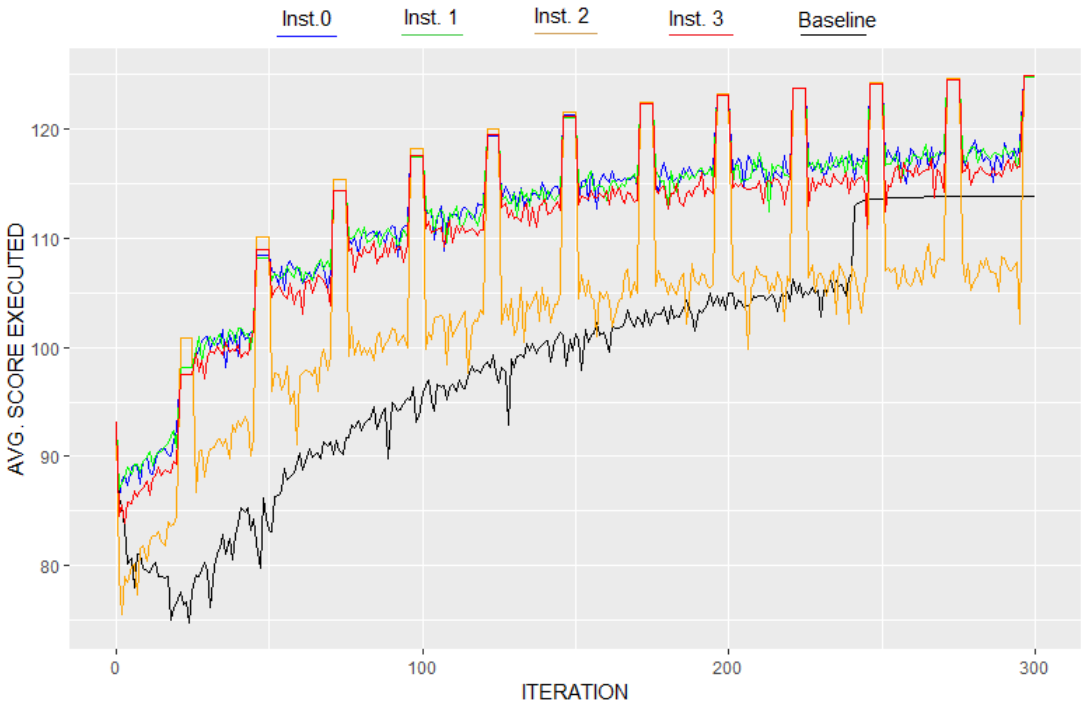

Figure 7.6: Average scores of executed plans of multiMATSim LA 0.1% across iterations (with number of instances $N = 4$ and $step = 25$)

is minimal. This observation lays the groundwork for further detailed analysis in subsequent sections. These results have been published in [48].

7.1.4 Performance Comparison between A64FX and Intel Xeon Gold 6230 for MATSim

As a reminder, we had the privilege of using the Fugaku supercomputer for our multiMATSim experiments. During our tests, we observed a significant performance difference, primarily attributable to MATSim itself. Our multiMATSim approach relies on running multiple instances of MATSim, which magnifies the impact of its performance on our overall results. Although the following observations are based on the Los Angeles 0.1% scenario, we found the same results for

the Los Angeles 1% scenario. In this section, we focus on comparing MATSim's performance on the A64FX CPU, which powers the Fugaku nodes, against the Intel Xeon Gold 6230, its counterpart in the Ruche cluster. Understanding these performance differences is crucial for optimizing our multiMATSim approach. We will attempt to explain these differences in the following section.

1. **Execution Duration**:
    - *A64FX*: For a typical MATSim LA 0.1% iteration, the execution time is 10 minutes.
    - *Intel Xeon Gold 6230*: Under similar conditions, the Intel Xeon Gold 6230 CPU completed an iteration in just 2/3 minutes.
2. **Processing Efficiencies**:
    - *A64FX*: Despite being designed for HPC workloads, efficiency for predominantly sequential programs like MATSim is lower, with notable observations of pipeline stalls, particularly for loops with a complex body.
    - *Intel Xeon Gold 6230*: Demonstrating better adaptability for sequential or slightly parallel applications, this architecture exhibited a superior ability to process MATSim efficiently.

Even though MATSim's performance is lower on Fugaku compared to Ruche, we wanted to verify if our approach, multiMATSim, offered better results on Fugaku compared to MATSim in terms of convergence.

Results on Fugaku with multiMATSim

We launched multiMATSim LA 0.1% with 4 instances ($N = 4$) and a reduced number of iterations ($max = 60$), exchanging at the 50th iteration ($step = 50$). On

Fugaku, multiMATSim LA 0.1% reached a stable state in about 8 hours, whereas it took about 30 hours for the baseline MATSim LA 0.1% to achieve the same. As on Ruche, multiMATSim allows the system to reach this stable state, where changes in agents' route plans become minimal from one iteration to the next, indicating that the system reaches an optimal equilibrium much faster compared to MATSim alone.

Architectural Differences

The profiler revealed differing behaviors of the JVM (Java Virtual Machine), with JVM settings varying between the CPUs. The number of calls to the garbage collector remained the same between the two CPUs, but the garbage collector intervention times were much longer on the A64FX. This might simply be due to the architectural differences between the two CPUs and may not be related to the difference in performance. Other differences were noted, such as thread management or memory management, which can be explained in the same manner.

7.1.5 Discussion on Performance Differences

We observe significant performance differences between the A64FX and Intel Xeon Gold 6230 architectures when executing MATSim, and we propose several hypotheses to explain these differences by examining the interplay of architectural attributes and MATSim's inherent software characteristics.

Sequential Nature of MATSim

MATSim's design leans heavily towards sequential execution, with only intermittent multithreading on specific modules. Such a design paradigm is intrinsically reliant on stable and robust CPU performance.

MATSim's computational framework largely operates on an event-based simulation model, which inherently limits the opportunities for parallel execution. This model processes events sequentially, simulating the behavior of agents within a transportation network. The event queue is typically processed by a single thread, with parallelism only exploited in certain phases, such as route replanning or specific computational tasks that can be isolated from the main event processing.

- **Core Structure and Out-of-Order Execution:** The A64FX's compact core structure and diminished resources for out-of-order optimization are not optimally suited for applications that are chiefly sequential. Unlike the Intel Xeon Gold 6230, which has extensive out-of-order execution capabilities, the A64FX focuses on energy efficiency and high-bandwidth memory access, features that do not benefit MATSim's sequential nature.

- **SIMD Instructions:** MATSim's inability to exploit SIMD instructions means that SVE functionalities within the A64FX are not leveraged. SIMD instructions are crucial for accelerating parallelizable tasks, but since MATSim does not inherently support these operations, the advanced vector capabilities of the A64FX remain underutilized.

- **Pipeline Structure:** The longer pipeline structure of the A64FX, which is prone to stalls for loops with intricate bodies, may act as a bottleneck for MATSim. MATSim's event processing and agent-based simulations involve complex looped structures with frequent branching and conditional operations. The deeper pipeline of the A64FX increases the penalty of branch mispredictions and pipeline stalls, leading to reduced performance efficiency.

- **Memory Access Patterns:** MATSim's performance is also influenced by memory access patterns. The A64FX's memory subsystem is optimized for

high-bandwidth access, suitable for applications with streaming data patterns. However, MATSim's memory access is more random and irregular, not fully benefiting from the A64FX's memory architecture.

- **Thermal and Power Considerations:** The A64FX is designed with a focus on power efficiency and thermal management, which can result in reduced peak performance for sustained high-load tasks. MATSim, requiring consistent high performance for extended simulation runs, might not achieve optimal performance on the A64FX due to these power and thermal constraints.

- **Parallel Task Management:** While MATSim does implement some level of parallelism, it is primarily in the form of task parallelism rather than data parallelism. This means that while certain tasks can run in parallel, the overall speedup is limited by the sequential processing of the main event queue. The Intel Xeon Gold 6230, with its robust multi-core architecture and support for efficient parallel task management, can handle these parallel tasks more effectively.

The architectural design of the A64FX is not fully utilized by MATSim due to the sequential nature of MATSim and its inability to leverage advanced vector processing capabilities. MATSim, with its reliance on both single-thread performance and limited multithreading, does not take full advantage of the A64FX's strengths, such as high memory bandwidth and SVE vector instructions. In contrast, the Intel Xeon Gold 6230, with its out-of-order execution, broader SIMD support, and efficient handling of both single-threaded and multithreaded tasks, provides an environment better suited to the computational demands of MATSim.

Java and MATSim

The performance of the JVM can vary significantly between different CPU architectures due to the optimizations applied during the compilation and runtime execution of Java code. Studies have shown that the performance of the A64FX architecture can be significantly enhanced through the use of libraries specifically compiled by Fujitsu [55][33]. These optimizations leverage the unique features of the A64FX architecture, such as its high memory bandwidth and vector processing capabilities.

However, during our experiments, the only available version of OpenJDK compiled by Fujitsu was OpenJDK 11, which is relatively old. Newer versions of OpenJDK, such as OpenJDK 17, include various performance improvements and optimizations, particularly for ARM processors. For instance, during an iteration of MATSim on Los Angeles 0.1% with Fugaku, the execution time reduced from approximately 15 minutes using OpenJDK 11 (compiled by Fujitsu) to 10 minutes using OpenJDK 17 (compiled by GCC). This significant improvement can be partly attributed to the specific optimizations for ARM processors incorporated in OpenJDK 17.

The JVM in OpenJDK 17 includes several enhancements over previous versions, such as:

- **Improved Garbage Collection**: Enhancements in garbage collection algorithms, such as the Z Garbage Collector (ZGC) and Shenandoah GC, provide better performance and lower pause times, which are crucial for applications like MATSim that handle large amounts of data.

- **Enhanced Just-In-Time (JIT) Compilation**: The JIT compiler in OpenJDK 17 has been optimized to generate more efficient machine code for ARM architectures, resulting in better runtime performance.

- **Additional ARM-Specific Optimizations**: OpenJDK 17 includes specific improvements for ARM processors, such as better handling of vector operations and memory access patterns, which are particularly beneficial for the A64FX architecture.

Given these improvements, it is plausible that a version of OpenJDK 17 or later, compiled by Fujitsu, could yield even better results. Fujitsu's compiler could potentially incorporate additional optimizations tailored specifically for the A64FX architecture, further enhancing performance. Therefore, utilizing a more recent version of OpenJDK compiled by Fujitsu may unlock additional performance benefits for MATSim running on the A64FX.

In conclusion, the performance of MATSim on different CPU architectures can be significantly influenced by the JVM optimizations specific to each architecture. While OpenJDK 11 compiled by Fujitsu provides a baseline improvement, leveraging newer versions of OpenJDK with specific ARM optimizations can lead to further performance gains.

Intel Xeon Gold 6230's Affinity

Considering the Intel Xeon Gold 6230 architecture:

- **Larger Die Size and Out-of-Order Executions**: Its larger die size provides enhanced resources for out-of-order executions, which can be particularly beneficial for tasks with a significant sequential component, such as MATSim. The architecture's extensive out-of-order execution capabilities help in reordering instructions to optimize CPU utilization and minimize idle cycles.

- **Versatile Approach for Sequential and Parallel Tasks**: This architecture's versatile approach encompasses optimizations that efficiently handle

both sequential and parallel tasks. The Intel Xeon Gold 6230 supports a broad range of optimizations that enhance performance across diverse workloads, accommodating the intermittent multithreading nature of MATSim. This allows the architecture to balance high single-thread performance with effective parallel processing, ensuring robust performance across different phases of the simulation.

- **Cache Hierarchy**: The Intel Xeon Gold 6230 features a robust cache hierarchy, including large L2 and L3 caches. This is advantageous for MATSim, as it helps reduce memory latency and improves data access speeds, particularly for large datasets typical of traffic simulations.

- **Hyper-Threading Technology**: The support for Hyper-Threading in Intel Xeon processors allows each core to handle two threads simultaneously. This can enhance the performance of MATSim during its multithreaded phases by improving parallel processing efficiency and maximizing CPU core utilization.

- **Advanced Vector Extensions (AVX-512)**: Although MATSim itself may not heavily rely on SIMD operations, the Intel Xeon Gold 6230 supports AVX-512, which can significantly boost performance for any underlying libraries or modules that do leverage SIMD. However, this advantage may also be seen in the A64FX's SVE capabilities. It is possible that MATSim and its dependent libraries currently leverage AVX-512 more effectively than SVE, potentially due to broader support and optimization in existing software.

- **Power and Thermal Management**: Intel's architecture includes sophisticated power and thermal management features that ensure sustained high performance without overheating. This is crucial for long-running simulations like MATSim, which require consistent processing power over extended

periods.

Concluding Thoughts

The A64FX is a notable architecture, recognized for its performance in memory-focused benchmarks and its power efficiency. Its design is optimized for massively parallel processing, as evidenced by its use in Fugaku, one of the leading supercomputers in the TOP500 ranking[69]. This makes it ideal for high-performance computing tasks that demand extensive data parallelism.

MATSim, however, presents a different set of requirements. As a simulation framework that primarily operates in a sequential manner with limited multithreading capabilities, MATSim does not fully leverage the parallel processing strengths of the A64FX. The core design of MATSim involves sequential event processing and sporadic multithreading, which does not align perfectly with the A64FX's capabilities designed for parallel workloads.

In contrast, the Intel Xeon Gold 6230, with its robust out-of-order execution, extensive SIMD support through AVX-512, and efficient handling of both sequential and parallel tasks, appears more suited to MATSim's operational patterns. This architecture provides a balanced environment where MATSim's limited parallelism can still benefit from enhanced single-thread performance and occasional multithreading.

The insights gathered suggest that future codes designed to handle massive datasets need to be better adapted to modern architectures. This involves rethinking core algorithms to increase parallelism, optimizing data structures for better concurrency, and leveraging advanced vectorization capabilities more effectively. These adaptations would enable applications to fully exploit the capabilities of architectures like the A64FX, potentially leading to substantial performance improvements.

In summary, while the A64FX represents a powerful architecture with significant potential, particularly for applications that can fully utilize its parallel processing capabilities, there is a substantial opportunity for the next generation of software to be designed with these advanced architectures in mind. This approach will ensure that future computational tasks, especially those involving large-scale data processing, can fully leverage the capabilities of cutting-edge hardware like the A64FX. These results have been published in [49].

7.1.6 Conceptual and Empirical Analysis of multiMATSim

Preserving the Integrity of MATSim Results

Here, the primary goal is to demonstrate how multiMATSim, leveraging an advanced strategy of parallel simulation, maintains the accuracy and reliability of the results produced by the standard MATSim transportation simulation system. This section aims to validate the hypothesis that the innovation introduced by multiMATSim does not compromise the integrity of the foundational simulations but, instead, enhances their efficiency while preserving their fidelity. To achieve this objective, a comprehensive comparative analysis is conducted, highlighting key performance indicators that are essential to assess the correspondence between the outcomes generated by multiMATSim and those produced by MATSim.

The selected indicators for this comparison include modal shares, road traffic volumes, and the distribution of trips (leg histogram) across hours of the day. These parameters are critical as they directly reflect the mobility behaviors of individuals as well as the utilization of the transportation network, thus providing a solid foundation for assessing the impact of any new simulation methodology on urban mobility dynamics.

By meticulously comparing these indicators, the analysis not only seeks to reaf-

firm the credibility of multiMATSim as a valuable extension of MATSim but also to ensure that any enhancements introduced by multiMATSim are rooted in the original system's robust simulation capabilities. This rigorous evaluation process involves not just a quantitative assessment of the similarities and discrepancies in the data but also a qualitative examination of how multiMATSim's parallel processing approach potentially leads to more efficient exploration of mobility solutions without deviating from the realistic urban transport scenarios modeled by MATSim.

Through this detailed examination, the section underscores the importance of retaining essential characteristics and outcomes of transport simulations when integrating innovative methodologies. By so doing, it reinforces the premise that advancements in simulation techniques should complement and extend the analytical precision of existing models, ensuring that the evolution in simulation technology continues to contribute constructively to the field of urban mobility studies.

Traffic Volume In our meticulous examination of the road traffic volumes produced by multiMATSim and MATSim, two critical time intervals were selected for comparative analysis: the morning peak hour from 8 am to 9 am and the evening peak hour from 7 pm to 8 pm. These intervals are quintessential for understanding commuting patterns and the daily rhythms of urban mobility.

Morning Peak Hour (8 AM to 9 AM) presented in Figure 7.7: The histograms for the morning peak period reveal distinct traffic peaks which correspond to the surge of commuters taking to the roads. This interval is often characterized by a significant spike in traffic as individuals begin their daily routines. In both multiMATSim and MATSim outputs, we observe a consistent distribution of these peaks across the x-axis, which likely maps to critical transit routes or

junctions within the city. While some fluctuations in the peak heights are evident — possibly a reflection of multiMATSim's parallel simulation approach — the overall correspondence in the spatial distribution of traffic volume suggests that multiMATSim preserves the integrity of MATSim's traffic flow patterns during this critical morning rush hour.

In Figure 7.7, the graph on the left represents MATSim and the one on the right represents an instance of multiMATSim, specifically the one that converged the fastest. Overall, we observed consistency among all instances of multiMATSim, which is why we chose to present only one here. The same applies to Figure 7.8, where the graph on the left represents MATSim and the one on the right represents the fastest converging instance of multiMATSim, and we observed uniform results across all multiMATSim instances.

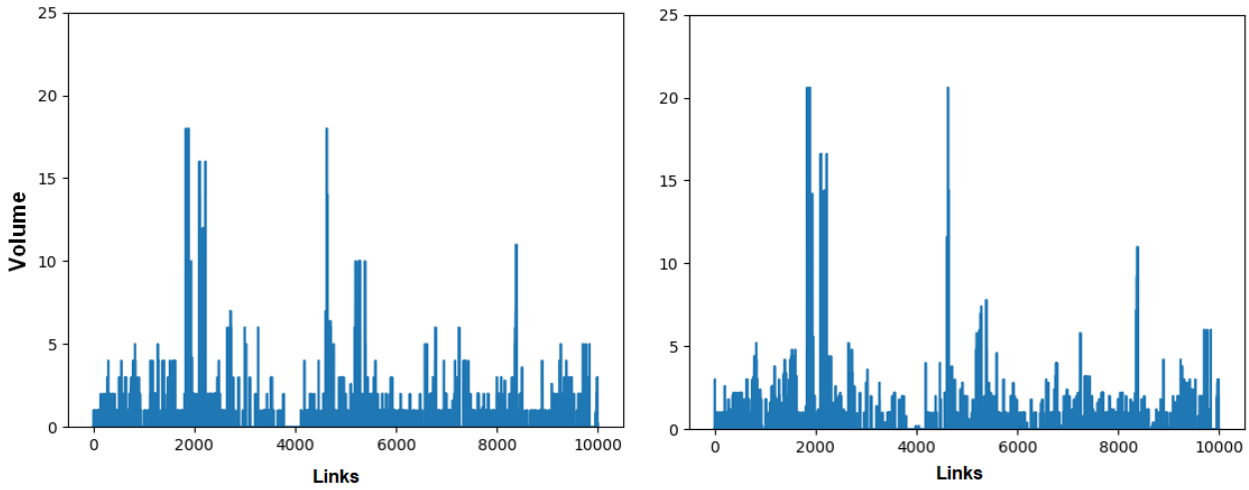

Figure 7.7: 8 AM to 9 AM Traffic Volume: Comparison between MATSim (on the left) and one instance of multiMATSim (on the right, $N = 4$, $step = 50$) During a Single Iteration for the LA 0.1% Scenario

Evening Peak Hour (7 PM to 8 PM) presented in Figure 7.8: The evening peak hour analysis paints a somewhat different picture. The end of the standard

workday typically leads to a dispersed exodus from the city center, visible in the histogram's distribution of traffic volumes. The comparison of multiMATSim and MATSim indicates that while there is an overall consistency in traffic distribution, the evening peaks are generally less pronounced than those in the morning, possibly reflecting a more staggered and varied return journey for commuters. Despite the slightly subdued nature of the peaks, the alignment between the histograms during this period further corroborates the fidelity of multiMATSim's simulations to real-world traffic behaviors captured by MATSim.

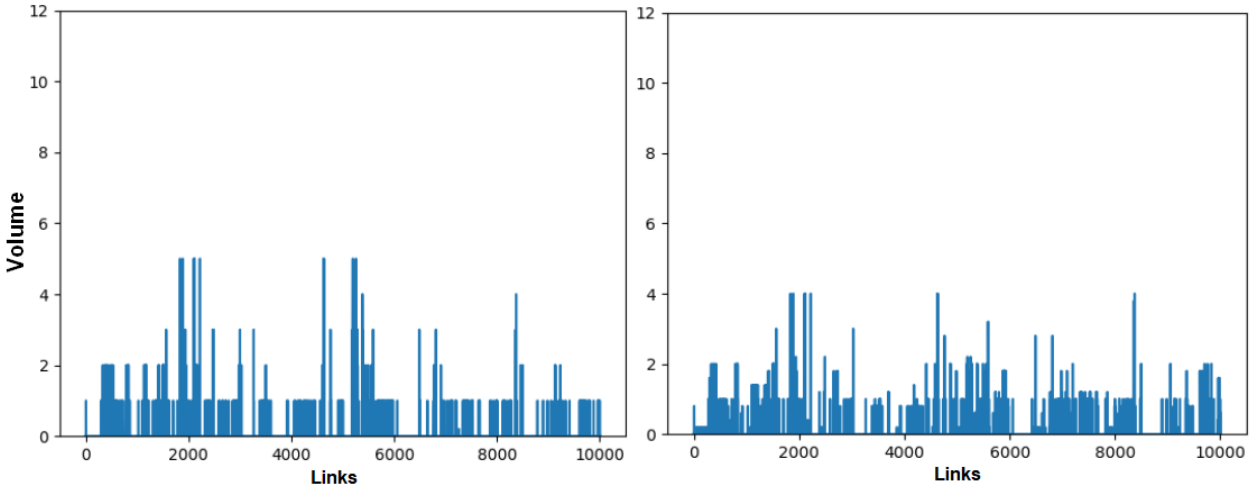

Figure 7.8: 7 PM to 8 PM Traffic Volume: Comparison between MATSim (on the left) and one instance of multiMATSim (on the right, $N = 4$, $step = 50$) During a Single Iteration for the LA 0.1% Scenario

Comparative Analysis: When juxtaposing the traffic patterns of these two peak periods, it's evident that the morning and evening rush hours have their unique characteristics, yet multiMATSim's performance remains steadfastly consistent with MATSim's established patterns. The peaks in the morning are generally sharper, reflecting a more concentrated and synchronous start to the day, while the evening shows a broader spread of traffic volumes, indicative of a more stag-

gered end to the day. This nuanced replication of different traffic behaviors across various times of the day demonstrates multiMATSim's robustness in modeling the complexities of urban transport dynamics.

The focused examination of these two peak periods — key touchstones for urban transportation analysis — provides a snapshot of multiMATSim's capabilities in terms of accurately simulating traffic patterns. It also illustrates the tool's potential utility in strategic planning and policy-making, ensuring that even with its advanced parallel simulation features, multiMATSim does not deviate from the realistic traffic flows established by the benchmark MATSim model.

Modal Shares: Another way to assess the fidelity of a new simulation method in comparison with the established MATSim model is to look at modal shares, as a standard approach. Modal shares provide insights into individuals' transportation preferences, indicating how various modes of transport such as cars, bicycles, public transit, and walking are utilized within the simulations.

A close alignment of modal shares between multiMATSim and MATSim would suggest that the mobility behavior of agents remains consistent despite the introduction of parallel simulations and the exchange of agent plans. This is crucial as modal shares are not just statistical outputs; they reflect underlying travel behaviors and choices that are influenced by a range of factors including infrastructure, policy, and personal preference.

In a multiMATSim setup, ensuring that modal shares do not deviate significantly from those of MATSim is essential. It confirms that the parallel processing and iterative refinement of strategies do not distort the travel patterns that the agents are supposed to emulate. For instance, if the modal share for public transportation within multiMATSim mirrors that of the MATSim output, it can be inferred that multiMATSim retains the core logic of agents' modal choice decision-

making.

Moreover, the reliability of modal shares as a measure of simulation fidelity is also grounded in their ability to indicate the accuracy of the model's representation of the real-world transportation system. If the simulated modal shares are in agreement with empirical data or expected trends, it demonstrates the model's capability to produce realistic outcomes.

In this context, we can analyze the modal share outputs from multiMATSim to ensure that the enhanced computational strategy enriches the simulation without compromising the validity of its results. An analysis that reveals similar modal shares across both multiMATSim and MATSim suggests that the advanced parallelization technique implemented in multiMATSim successfully captures the diversity and complexity of travel behaviors in urban settings. Such an outcome would provide robust evidence supporting the use of multiMATSim in transport planning and policy development scenarios, leveraging its computational advancements while remaining true to the validated behaviors of the standard MATSim model.

Figure 7.9 display mode statistics for transportation within a simulation environment, from a standard MATSim output. The other on Figure 7.10 is generated from a multiMATSim instance. These graphs are key to understanding how individuals within the simulation environment choose different modes of transportation over the course of the simulation iterations.

Analysis of the Standard MATSim Output: Figure 7.9 shows a stable selection of modes across iterations, with the most significant proportion being the 'car' mode, followed by 'public transport (pt)' and 'walk'. Other modes such as 'bike', 'ride', and 'freight' maintain a relatively low but consistent share. The stability of these shares over the iterations suggests that the simulation has reached equilibrium, with little change in modal choice as the simulation progresses.

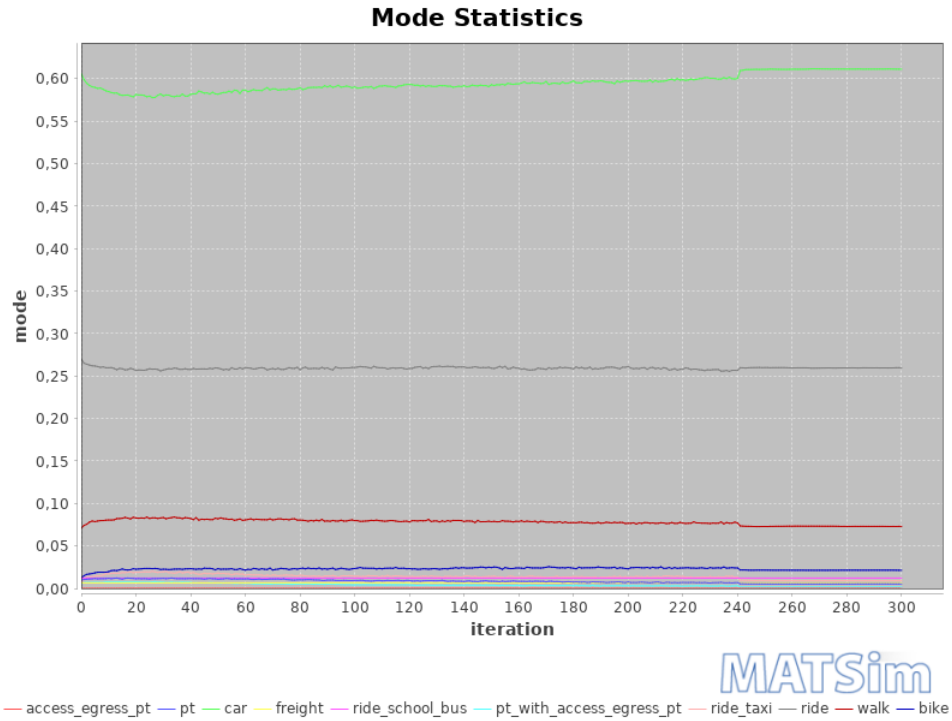

Figure 7.9: Modal Shares on MATSim LA 0.1% Scenario

Analysis of the multiMATSim Instance Output: Figure 7.10 although aesthetically different due to its generation from a multiMATSim instance, exhibits a similar trend in the distribution of mode choices. The dominant modes remain consistent with the 'car' mode being the most prevalent. One can observe slight variances in the shares of each mode, which may be attributable to the different dynamics introduced by multiMATSim's parallel processing and agent plan exchange features.

Comparative Analysis: When comparing the two images, the overall consistency in mode share distributions indicates that multiMATSim maintains the behavior patterns inherent in the MATSim model. This is crucial for validating multiMATSim as a reliable extension that can enhance computational efficiency without sacrificing the behavioral authenticity of the simulation results. Any minor discrepancies observed might be due to the enhanced dynamics of multiMATSim,

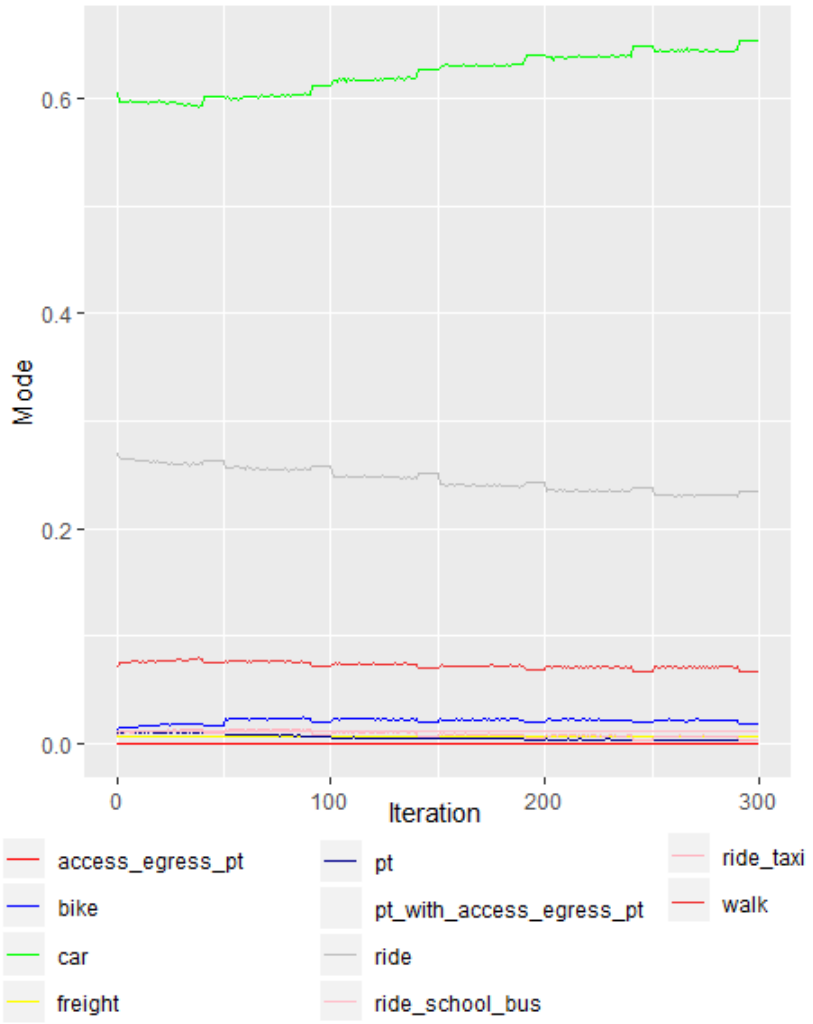

Figure 7.10: Modal Shares on one instance of multiMATSim LA 0.1% Scenario ($N = 4$, $step = 50$)

yet they do not signify a fundamental shift in modal choice behavior.

This comparison of mode shares from MATSim and multiMATSim allows us to confirm that multiMATSim's parallel simulation approach has not distorted the inherent transportation preferences of the agents within the simulation. The congruence of these results with the expected real-world modal preferences further substantiates multiMATSim's utility as a tool for accurate and efficient transporta-

tion modeling. These results have been published in [46].

7.1.7 Conceptual hypotheses explaining the effectiveness of multiMATSim

In the conceptual elucidation of multiMATSim's efficiency, the distributed optimization and collaboration among instances conjecturally contribute to a more robust exploration of the solution space. This distributed approach is philosophically akin to collective intelligence and swarm optimization, realms where pooling cognitive resources—here, agents' plans—purportedly leads to superior problem-solving.

Hypotheses on Distributed Optimization and Collaboration:

It is hypothesized that multiMATSim's distributed architecture might facilitate a form of collective intelligence akin to what is observed in social insect behavior. Each instance operates as an autonomous agent with the potential to probe disparate sections of the solution space. The parallel, yet interconnected, processing could theoretically mitigate the risks of convergence to local optima, a concern in centralized systems. The collaborative aspect, where instances share their best-found solutions, is reminiscent of swarm behavior, suggesting that such inter-instance communication could lead to a collective enhancement in identifying efficient routes or strategies within the transportation network.

Speculation on Exploration-Exploitation Dynamics:

The trade-off between exploration and exploitation is another area of academic interest that multiMATSim might navigate with speculative finesse. The system could hypothetically achieve a balance wherein exploration—identifying new po-

tential solutions—and exploitation—utilizing known efficient strategies—are dynamically harmonized through the iterative process of plan exchanges. Such a mechanism could allow for broader scanning of new strategies while ensuring that the cumulative knowledge from all instances is utilized effectively.

Conjectures on Robustness and Adaptability:

As for robustness and adaptability, multiMATSim is presumed to exhibit resilience across various simulation scenarios and configurations. The system's architecture, with its potential for adaptability, hypothetically allows for responsive adjustments to mobility behavior modeling and alterations in simulation parameters. This adaptability is pivotal in accurately simulating the fluid dynamics of urban transport systems, where variability is the norm.

These postulations, while theoretically grounded, underscore the potential of multiMATSim to embody principles of swarm intelligence and distributed problem-solving within a computational context. They suggest that by harnessing the collective capabilities of multiple simulation instances, multiMATSim might not only maintain the integrity of individual agent decisions but also enhance the overall quality and reliability of the simulation outcomes. However, such claims remain speculative and must be substantiated through empirical validation.

These results have been the subject of several publications. Notably, the multiMATSim method was first presented in its initial version in an early publication [47], before being refined to its current version with results on the Los Angeles 0.1% scenario and an evaluation of the reliability of its results compared to the baseline [46]. Two other publications described the study of various MATSim variables and their impact on performance and reliability [48], as well as the scalability and performance differences between two different CPU architectures [49].

7.2 AI-based Approach

This approach leverages machine learning methods to efficiently predict transport mode choices in a multi-agent traffic simulation environment. By addressing the computational intensity and time-consuming nature of traditional simulations like MATSim, it offers a faster and scalable alternative. By training a neural network on a representative dataset, we aim to develop a model capable of producing accurate predictions in a fraction of the time required by conventional simulation methods. All experiments and results presented in this section were obtained using the high-performance computing nodes on Ruche.

In this section, we evaluate the performance of our model using MSE as the loss function to establish a baseline for comparison. MSE was chosen due to its property of penalizing larger errors more heavily, making it a standard choice for minimizing the difference between predicted and actual values. Additionally, we tested the model with the Huber Loss function, which offers a balance between MSE and MAE, providing robustness to outliers while maintaining accuracy for smaller errors.

Following this, we will delve into various model performance metrics to gain a comprehensive understanding of the model's predictive capabilities. This will include a detailed error analysis to identify patterns and sources of inaccuracies in the model's predictions. We will also compare the errors between different loss functions to highlight their impact on model performance.

The discussion will then shift to the benefits of the AI-based approach, emphasizing time and resource savings. We will explore the scalability of the method and address its limitations, particularly focusing on the MLP networks used in our approach.

Finally, we will address the challenge of reproducing heuristic results obtained

by MATSim with our AI model. This involves comparing the outcomes of our neural network with those produced by MATSim's heuristic algorithms to evaluate the fidelity and reliability of our model in replicating complex agent behaviors.

7.2.1 Model Performance Metrics

We evaluated the overall performance of our model using several key metrics. These metrics provide a comprehensive view of the model's effectiveness in predicting transport modes in a multi-agent traffic simulation:

- **Global MAE**: The mean absolute error (MAE) for the model was 0.0524. This metric indicates the average magnitude of errors in the model's predictions, with a lower MAE suggesting closer alignment with the actual values. In our context, an MAE of 0.0524 means that, on average, the predicted count of transport mode usage deviates by 0.0524 from the actual count.

- **Global MSE**: The mean squared error (MSE) was 0.01. MSE is sensitive to larger errors due to the squaring of each error term, providing insight into the presence of larger errors within the predictions. A lower MSE indicates that large errors are infrequent. In our model, an MSE of 0.0106 suggests that the deviations between the predicted and actual values are generally small, and larger errors are rare.

- **Global R2**: The R-squared (R2) score was 0.945. The R2 metric measures the proportion of variance in the dependent variable that is predictable from the independent variables. An R2 value of 0.945 means that 94.5% of the variance in transport mode usage can be explained by our model, indicating a strong fit to the data and effective capture of underlying patterns.

- **Global Cosine Similarity**: The cosine similarity was 0.799. Cosine similar-

ity measures the cosine of the angle between two non-zero vectors, assessing how similar the direction of the predicted vector is to the actual vector of transport modes used. A cosine similarity of 0.799 indicates that the model's predictions are generally well-aligned with the actual usage patterns, even if there are differences in the magnitude of the counts.

Residual Analysis: To further evaluate the performance of the model and the distribution of prediction errors, we generated a residual plot (shown in the figure below). This histogram represents the absolute values of residuals, which are the differences between the predicted and actual values for transport mode usage.

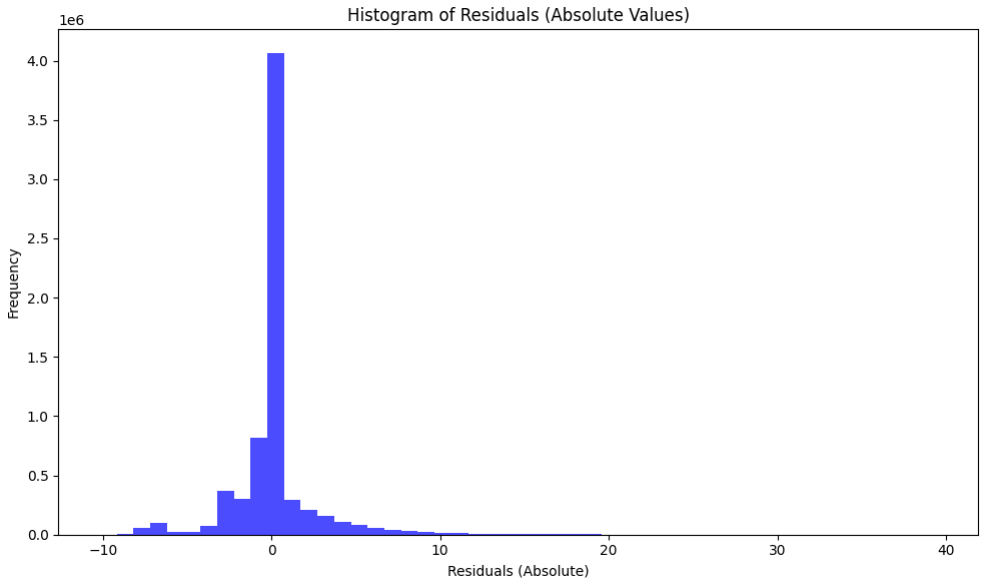

Figure 7.11: Histogram of Residuals (Absolute Values)

The residuals histogram reveals several key insights:

1. **Concentration of Errors Near Zero**: The majority of the residuals are clustered around zero, indicating that the model's predictions are generally close to the actual values. This supports the low MAE and MSE values, confirming that the model performs well in most cases.

2. **Long Tail for Larger Errors**: While most errors are small, there is a long tail extending toward larger residuals. This suggests that, although infrequent, the model does make larger prediction errors for some agents. However, these larger errors do not heavily impact the overall performance, as shown by the relatively low MSE.

3. **Limited Presence of Outliers**: The histogram does not show a significant number of extreme outliers. The few large residuals that exist are relatively controlled, further validating the model's robustness in handling outliers and maintaining generalization across different scenarios.

The residual plot, together with the performance metrics, confirms the reliability and effectiveness of our MLP model in predicting transport mode usage within the multi-agent simulation. The model maintains most errors within a small range, while larger errors, though present, are rare and do not significantly impact overall accuracy.

Collectively, these metrics demonstrate the strength of our MLP model using the MSE loss function. The results, particularly the MAE and the high R2 score, highlight the model's robustness and reliability in handling complex prediction tasks. Additionally, the strong cosine similarity shows that the predicted transport mode usage patterns are closely aligned with actual patterns, further confirming the model's capacity to generalize across various scenarios. This comprehensive analysis emphasizes the potential of AI-based approaches in improving the efficiency and accuracy of multi-agent traffic simulations, offering a promising direction for future research and development.

7.2.2 Error Analysis

Error Evaluation

Evaluating errors is crucial to understand the performance and limitations of the model. In this section, we provide a detailed analysis of prediction errors and compare errors across different loss functions.

Detailed Analysis of Prediction Errors

- **Error Distribution:** Analysis of prediction errors revealed that most errors were concentrated around certain transport modes. Commonly used transport modes, such as car and public transport, had relatively low prediction errors. This can be attributed to the larger volume of data available for these modes, allowing the model to learn their patterns more effectively. The abundance of training examples for these modes provides a more comprehensive representation of their usage, leading to more accurate predictions.

In contrast, less frequent modes, such as taxi and school bus, showed higher errors. The increased prediction errors for these modes can be attributed to their rarity in the training data. With fewer examples to learn from, the model struggles to capture the underlying patterns and variability associated with these less common modes of transport. This scarcity makes it challenging for the model to generalize well and accurately predict their occurrences.

- **Extreme Behaviors:** Scenarios with high variability in agent behaviors also led to larger errors. For example, agents with atypical behaviors or very specific transport modes were often mispredicted. These errors might be due to the model's limited ability to capture highly diverse behaviors with a limited sample of data.

- **Errors by Scenario:** Grouping errors by scenario, we observed that certain scenarios consistently had higher errors. This suggests that specific characteristics

of these scenarios pose particular challenges for the model. A deeper analysis of these scenarios could help identify specific characteristics or behaviors that are poorly represented in the training data.

7.2.3 Comparison of Loss Functions and Their Impact on Predictions

In our study, we compared two loss functions — MSE and Huber loss — to evaluate their respective impacts on model performance. The choice of loss function is crucial as it affects how errors are penalized, particularly in the presence of outliers.

MSE (Mean Squared Error): MSE was selected for its straightforward approach to penalizing errors, particularly its sensitivity to larger errors due to the squaring of residuals. This makes MSE more responsive to outliers, as large errors can significantly increase the total loss. In our experiments, the MSE loss function provided consistent results, with the mean absolute error and R2 metrics reflecting stable performance across both training and validation data.

Huber Loss: The Huber loss function, known for being less sensitive to outliers by combining the benefits of MSE and MAE, was also evaluated. Huber loss behaves quadratically for small errors and linearly for larger ones, making it more robust to outliers while maintaining accuracy for smaller errors. However, when comparing the results, we found that the overall performance metrics (MAE, MSE, and R2) were nearly identical to those obtained using MSE.

Conclusion: Despite the theoretical advantages of Huber loss in handling outliers, the performance of our MLP model remained comparable whether we used MSE or Huber loss. Both loss functions resulted in similar prediction accuracy

and generalization capability, as indicated by the similar MAE and R2 scores. Therefore, in this specific case, the choice of loss function had minimal impact on the model's overall performance, suggesting that both are suitable for our task.

7.2.4 Impact of Hyperparameters

The choice of hyperparameters had a significant impact on the model's performance. Through a thorough analysis, we experimented with different configurations to optimize learning and generalization. The key hyperparameters studied include:

- **Learning Rate:** We tested values of 0.01, 0.015, 0.001, and 0.0001. A learning rate of 0.001 was found to be optimal, providing a balance between fast convergence and stability.
- **Batch Size:** Batch sizes of 16, 32, and 64 were evaluated. A batch size of 64 provided the best results, ensuring both gradient stability and efficient training.
- **Number of MLP Layers and Neurons:** Configurations with 1 to 4 layers and neuron counts of 128 and 256 were tested. The optimal configuration was two fully connected layers with 256 neurons each, which offered a good trade-off between model complexity and generalization.
- **Dropout:** Dropout rates of 0.1, 0.2, 0.3, and 0.4 were evaluated. A dropout rate of 0.1 was optimal, minimizing overfitting without compromising the model's capacity to learn effectively.

Optimal Configuration The optimal configuration identified through these experiments consists of a learning rate of 0.001, a batch size of 64, two MLP layers

with 256 neurons each, and a dropout rate of 0.1. This setup achieved the best performance in terms of prediction accuracy and generalization, as reflected in the validation metrics. Fine-tuning these hyperparameters significantly improved the model's robustness and its ability to capture complex interactions among agents in the simulation.

7.2.5 Discussion on the Benefits of the AI-based Approach

Gains in Time and Resource Efficiency Compared to MATSim

The use of an AI-based model, specifically MLP neural networks, offers significant advantages in terms of time savings and resource efficiency compared to the direct use of MATSim. While MATSim is a powerful tool for multi-agent traffic simulation, it is notoriously resource-intensive and time-consuming. The complexity of MATSim lies in its ability to simulate thousands of individual agents, considering numerous variables and parameters such as schedules, costs, transportation mode preferences, and agent interactions.

The scoring function in MATSim, which evaluates the transport mode choices of agents, is particularly complex. This function takes into account various factors such as travel time, cost, comfort, and other context-specific variables. The mathematical formula for this scoring function is given by:

$$S = \sum_{i=1}^{n} (\beta_i \times V_i) + \sum_{j=1}^{m} (\gamma_j \times P_j)$$

where:

- S is the total score for an agent.
- β_i is the weight associated with the i-th activity or transport mode.
- V_i is the value of the i-th activity or transport mode.

- γ_j is the weight associated with the j-th penalty parameter (e.g., waiting time, cost, etc.).

- P_j is the value of the j-th penalty parameter.

This function evaluates both the positive and negative aspects of agents' choices, directly influencing their transport mode decisions. While this allows for precise simulation, the complexity of this function significantly increases the computational time and resources required.

In contrast, our AI-based approach significantly simplifies this process. By training an MLP model on data simulated by MATSim, we can predict simulation outcomes without needing to rerun lengthy and resource-intensive simulations each time. Our model utilizes a minimal dataset with only 8 input columns (including features such as total distance traveled and occurrences of different transport modes) and 7 output columns (counts of transport modes used). This reduction in input complexity allows the model to maintain acceptable accuracy while minimizing computational costs.

For instance, our model achieved a MAE of 0.051 on the training data and 0.0524 on the validation data. This MAE indicates the average magnitude of errors in our predictions, regardless of their direction. An MAE of 0.0524 means that, on average, our predictions were off by 0.0524 units per transport mode, which is a relatively small error given the discrete nature of the output values.

These results demonstrate that our model can provide reliable predictions while being significantly faster and less resource-intensive than running full MATSim simulations. Additionally, training our MLP neural network took only a few minutes in a sequential implementation. At larger scales, it is likely that parallelization would further enhance performance and significantly reduce training time. Once trained, the model can generate predictions almost instantaneously.

Although our AI-based approach does not entirely replace the need for detailed simulations like those provided by MATSim, it offers a rapid and efficient solution for applications where time and resources are limiting factors. Our approach reduces computational load while providing reasonably accurate results, offering a balance between complexity and practicality.

Simplicity of the Approach and Its Implications

This approach presents the advantage of greater simplicity compared to traditional simulation methods. Unlike MATSim, which requires detailed and complex configuration including geography-specific data, infrastructure, and agent behavior, the use of a neural network model simplifies this process. Once the model is properly trained, users can make predictions without delving into the intricate details of simulation configuration and management.

However, this simplification comes with certain limitations. By aiming to simplify the model and reduce computation time, we have deliberately limited the input data to the most relevant elements for our specific use case. This means that we do not capture all the nuances of a complex transport system as modeled by MATSim. Despite this limitation, our AI-based approach offers a quick and efficient solution for applications where time and resources are limiting factors.

In conclusion, while our current AI-based approach does not capture all the subtleties of MATSim, it offers promising results. These results suggest that more elaborate models could potentially match the accuracy of traditional simulations while retaining the advantages of speed and simplicity, paving the way for more advanced and detailed applications.

7.2.6 Scalability and Limitations of MLP

Challenges of Scalability and Limitations of MLP for Large-Scale Simulations

As the number of agents increases in multi-agent simulations, the computational and memory requirements grow exponentially. While MLP neural networks are effective at modeling structured data, their fully connected architecture presents significant challenges when scaling up to large-scale simulations. These limitations are mainly related to computational cost, memory usage, and scalability.

In our study, handling scenarios with 1,000 agents already posed a substantial computational task. However, when scaling to tens or hundreds of thousands of agents, three key issues arise:

- **Computational and Memory Costs**: The increase in agents significantly raises the computational load, as every neuron in one layer is connected to all neurons in the next. This leads to an exponential growth in the number of parameters, making MLPs computationally expensive and memory-intensive, as storing the weights and activations becomes a challenge.

- **Training Bottlenecks**: As the complexity of agent interactions grows, MLPs require deeper architectures and more neurons to capture these dynamics, which can result in overfitting and a decline in performance. The large number of parameters also slows down the training process, creating bottlenecks in computation.

- **Parallelization Limitations**: While MLPs are more parallelizable than other architectures like LSTMs, the dense connections between layers limit the extent to which parallel processing can be fully leveraged. For large-scale simulations, optimizing MLPs may require advanced parallelization

techniques or specialized hardware to distribute the computational load effectively.

While MLPs offer a powerful method for modeling agent interactions in multi-agent scenarios, scaling them up to larger simulations introduces significant challenges. Addressing these issues may require architectural adjustments and more sophisticated optimization methods to maintain performance as the simulation scale increases.

Adaptability to Different Geographic Regions and Numbers of Agents

Challenges of Model Generalization One of the significant challenges in deploying MLP-based neural networks for traffic simulations lies in their adaptability to different geographic regions and varying numbers of agents. Unlike traditional simulation tools like MATSim, which are designed to handle diverse scenarios with detailed configurations, MLP models require careful consideration when generalizing to new environments.

- **Geographic Generalization**: Each geographic region has unique characteristics, such as infrastructure, population density, and transportation behavior. An MLP model trained on data from one region may not perform well when applied to another region with different characteristics. Retraining or fine-tuning with region-specific data is often necessary to maintain accuracy, which can be resource-intensive.

- **Scalability to Different Numbers of Agents**: Our current model is trained on scenarios with a fixed number of 1,000 agents. Scaling the model to handle varying numbers of agents, especially in much larger simulations, introduces challenges. The model's architecture and training process need

adjustments to accommodate different scales, which can increase complexity and computational demands.

- **Data Availability and Quality**: Generalizing the model relies heavily on the availability and quality of data from different regions. In regions with sparse or low-quality data, model performance may degrade. Establishing robust data collection and preprocessing pipelines is essential for the successful application of MLP models across diverse geographic areas.

- **Transfer Learning and Domain Adaptation**: Techniques like transfer learning and domain adaptation can help mitigate these challenges by leveraging pre-trained models and fine-tuning them on new data. However, careful implementation is required to ensure the model captures region-specific nuances without overfitting.

While MLP models provide powerful capabilities for traffic simulation, their adaptability to different geographic regions and numbers of agents presents significant challenges. Addressing these issues involves advanced training techniques, robust data pipelines, and potential adjustments to the model architecture to ensure reliable and accurate performance across various scenarios.

Limitation of Scenario Data

Limited Data for Complete Scenarios One of the main challenges in training neural network models for traffic simulations is the need for multiple examples of each complete scenario. In practice, a complete scenario provides only a single data example, which is insufficient for the demands of deep learning. This limitation can hinder the model's ability to generalize and provide accurate predictions across varied scenarios.

Extraction of Thousands of Scenarios from Larger Ones To address this limitation, we extracted thousands of sub-scenarios from larger scenarios. This approach creates a more diverse and extensive dataset from existing simulations, which is crucial for training neural network models. By dividing large scenarios into many smaller ones, we were able to artificially increase the size of our training dataset while preserving the essential structure and dynamics of agent interactions.

Evaluation of Data Augmentation Techniques Despite this extraction approach, the data remains limited, making it essential to evaluate data augmentation techniques to further enhance the model's performance. Data augmentation techniques, such as Variational Autoencoders (VAE) and Generative Adversarial Networks (GAN), can generate new synthetic data based on the distributions of existing data. This allows the creation of a richer and more varied dataset, potentially improving the model's robustness and accuracy.

Potential for Improved Results with Larger Data Volumes With a higher volume of data, it is likely that the model's performance would improve significantly. A larger and more varied dataset better captures the different dynamics and interactions in traffic simulations, reducing the risk of overfitting and enhancing the model's ability to generalize to new scenarios. By continuing to explore and implement data augmentation techniques, we could potentially achieve even more accurate and reliable results, bringing the model's predictions closer to those obtained through comprehensive and detailed simulations like MATSim.

Exploration of Other Neural Network Architectures

Transformers Transformers have emerged as a powerful alternative to traditional neural networks, including MLPs, especially in tasks requiring the management of long-term dependencies and enabling parallel processing. Unlike MLPs,

which rely on fully connected layers that can become computationally expensive with large inputs, Transformers can handle entire sequences simultaneously through their self-attention mechanism. This results in improved computational efficiency and potentially faster training times.

In the context of large-scale traffic simulations, where it is crucial to handle long-term dependencies and interactions between numerous agents, Transformers offer a significant advantage. Their ability to model relationships over long distances within the data provides a more comprehensive understanding of agent behaviors over extended periods. This makes them particularly well-suited for complex traffic scenarios, where capturing interactions across a broad temporal or spatial scale is critical. Additionally, the parallel processing capabilities of Transformers allow them to scale more efficiently than MLPs, making them an attractive option for simulations involving large datasets or many agents.

Convolutional Neural Networks (CNNs) CNNs, although traditionally used for image recognition tasks, can be adapted to capture spatial relationships within traffic data. By treating traffic scenarios as grid-like structures, CNNs can effectively identify and learn spatial patterns and dependencies. This capability makes CNNs particularly suitable for geographic adaptability, enabling them to model variations in infrastructure and traffic patterns across different regions. The local connectivity and weight sharing properties of CNNs allow them to detect and learn from local dependencies, providing a more nuanced understanding of agent interactions and movement patterns. For instance, CNNs can help model how agents navigate through different urban layouts, taking into account local infrastructure details.

Graph Neural Networks (GNNs) GNNs are particularly well-suited for traffic simulation tasks due to their ability to model networks and relationships. In

traffic systems, the infrastructure can be naturally represented as a graph, with intersections as nodes and roads as edges. GNNs can incorporate these infrastructure details and model the interactions between agents, making them ideal for simulating complex transport networks. They excel in representing the connectivity and dependencies in a traffic system, allowing for a more accurate and detailed modeling of agent behaviors and interactions. By leveraging the graph structure, GNNs can capture the dynamics of transportation networks, providing insights into congestion, route optimization, and other critical aspects of traffic management. This capability makes GNNs a promising architecture for future research aimed at enhancing the scalability and accuracy of traffic simulations. Additionally, GNNs can model the non-Euclidean nature of traffic networks, capturing more complex relationships and interactions than traditional neural network architectures.

Overall, exploring these advanced neural network architectures can significantly improve the scalability, accuracy, and efficiency of traffic simulations, paving the way for more sophisticated and comprehensive models in traffic management and urban planning.

7.2.7 Reproduction of Heuristic Results by AI

Utilizing artificial intelligence to model multi-agent traffic presents the challenge of reproducing results obtained by heuristic methods like those used in MATSim. MATSim relies on heuristic algorithms that consider various constraints and objectives to realistically simulate agent behaviors. In contrast, neural networks learn directly from data without explicitly following specific rules or heuristics. This raises several important questions.

Fidelity of Predictions

Our results indicate that the MLP model achieved a MAE of 0.051 on the training data and 0.0524 on the validation data for predicting the transport modes used by agents. While these results are promising, it is essential to note that predictions may differ from the heuristic results of MATSim, especially in rare or complex scenarios. The fidelity of AI predictions relative to heuristics largely depends on the quality and representativeness of the training data.

Interpretability of Models

Neural networks are often considered "black boxes" making it challenging to interpret predictions. Unlike MATSim, where decisions are based on explicit rules, the decisions made by a neural network result from a complex and non-transparent learning process. To improve interpretability, we propose integrating techniques such as saliency maps, sensitivity analyses, and explainability methods like LIME or SHAP. These tools can help identify the most influential features in transport mode decisions and understand model behaviors.

Performance and Robustness

The robustness and performance of neural networks are highly dependent on the quality and diversity of the training data. Our results show that the MLP model performs well on validation data, but it is crucial to evaluate its performance across varied scenarios to ensure general robustness. To enhance this robustness, we plan to use data augmentation techniques to enrich our dataset. This includes creating additional synthetic scenarios that simulate varied traffic conditions, enabling the model to better generalize to real-world situations.

In conclusion, although our initial approach has shown promising results, many

challenges remain in fully reproducing the heuristic results of MATSim. Future work will need to focus on improving the robustness, accuracy, and adaptability of our models.

Conclusion

The AI-based approach for modeling multi-agent traffic simulations has shown promising results. By leveraging MLP networks, we have significantly reduced the complexity and resource requirements compared to traditional methods like MATSim. Our model, trained on a simplified dataset, achieved a MAE of 0.051 on training data and 0.0524 on validation data, indicating reliable predictive performance.

This approach not only streamlines the simulation process but also provides near-instantaneous predictions once the model is trained, offering substantial gains in time and computational efficiency. Despite the simplification, the results suggest that with further refinement and optimization, our model could potentially approach the precision of detailed simulations provided by MATSim.

However, challenges remain, particularly in scaling the model to handle larger datasets and more complex scenarios. Additionally, the need for extensive and diverse training data necessitates further exploration into data augmentation techniques.

An important consideration in this research is the difference between heuristic approaches and AI-based methods. MATSim relies on heuristic algorithms that integrate various constraints and objectives to realistically simulate agent behaviors. These heuristics are designed to replicate human decision-making processes and interactions within a transportation network. In contrast, our neural network model learns patterns directly from the data, raising questions about its ability to reproduce the nuanced results generated by heuristic approaches.

Several conceptual and scientific questions arise from this comparison:

- Can AI-based models fully capture the complexity and variability of human behavior in transportation networks as heuristics do?
- What are the limitations of AI in terms of interpretability and explainability compared to heuristic methods?
- How can we ensure that AI models generalize well across different geographical regions and scales?
- What methods can be employed to combine the strengths of both heuristic and AI-based approaches for more robust traffic simulations?

Although this research is encouraging, it represents only a first step. The results obtained open the way for more in-depth investigations and future developments. The simplicity and efficiency of our approach highlight its potential as a complementary tool to traditional simulations. With further development, it is conceivable that AI models could not only reproduce but also enhance the accuracy and scalability of traffic simulations, providing more sophisticated and comprehensive solutions in the future.

Chapter 8

Conclusion

8.1 Summary of Contributions

This dissertation explores various approaches to enhance multi-agent traffic simulations by focusing on optimizing computational performance and applying artificial intelligence techniques. Our work is structured around two major contributions, each aimed at addressing specific challenges in the field of traffic simulations.

We began with a comprehensive state-of-the-art review of multi-agent traffic simulators, highlighting historical advancements, persistent challenges, and emerging technological opportunities. This review identified two primary avenues for improving the performance and flexibility of these simulators: leveraging high-performance computing architectures and applying AI techniques.

The first contribution leverages the Unite and Conquer method to provide a new algorithmic approach for multi-agent traffic simulation, with an initial application on MATSim called MultiMATSim. Inspired by high-performance linear algebra techniques, this approach introduces intrinsic multi-level parallelism, heterogeneity, and fault tolerance. We demonstrated how this method could be adapted to enhance the speed and efficiency of traffic simulations, particularly for

large urban networks. By applying this approach, we observed a significant reduction in computational costs while maintaining or even improving the accuracy of simulations.

Next, we examined the integration of advanced HPC architectures and the associated software ecosystems. Studying systems like Fugaku, Cygnus, and Ruche highlighted the advancements towards faster, more efficient, and more capable systems to meet contemporary computational demands. These systems exemplify a trend towards better energy efficiency and increased performance, showing that cutting-edge technology can be both high-performing and environmentally conscious. This section also underscored the importance of the symbiosis between hardware and software to maximize performance and efficiency in executing complex computational tasks for multi-agent traffic simulators.

The second major contribution focused on applying machine learning techniques to accelerate multi-agent traffic simulations. Using Multilayer Perceptron (MLP) neural networks, we developed a model capable of predicting an initial discrete variable, specifically the transport modes used by agents. The results obtained, with a MAE of 0.051 on training data and 0.0524 on validation data, indicate reliable predictive performance. These findings are encouraging and suggest that AI-based models could potentially complement traditional simulation methods for certain applications.

We also explored the challenges related to scaling AI models to handle larger datasets and more complex scenarios. A major limitation of our MLP-based approach is the fully connected architecture, which can become computationally expensive as the size of the input data increases. While MLPs allow for greater parallelization compared to sequential models, efficiently scaling the training process still requires advanced techniques and significant computational resources. Optimizing the parallelization of neural networks remains a promising avenue to

overcome these limitations, but it necessitates further exploration of distributed computing strategies and hardware acceleration.

In conclusion, this dissertation proposes promising approaches to enhance multi-agent traffic simulations. While challenges remain, particularly in terms of scalability and model generalization, the results obtained open interesting perspectives for future research. The combination of high-performance computing techniques and artificial intelligence offers considerable potential for developing more efficient, accurate, and adaptive traffic simulators. These contributions lay the groundwork for significant advancements in the field, with potential applications for urban mobility planning and management.

8.2 Discussion and Perspectives

The research presented in this dissertation proposes new methods for enhancing multi-agent traffic simulations through the application of high-performance computing techniques and artificial intelligence. While the results are promising, they also highlight several challenges and areas for future research. This section discusses these challenges, provides insights into the implications of our findings, and outlines potential directions for future work.

8.2.1 Challenges and Limitations

Scalability and Computational Efficiency

multiMATSim:

One of the primary challenges identified in this research is the scalability of the proposed methods. The Unite and Conquer method implemented in multiMATSim offers a novel algorithmic solution for multi-agent traffic simulation.

This approach effectively divides the simulation into smaller, more manageable sub-problems that can be solved separately and then combined. The initial application of multiMATSim on MATSim demonstrated significant improvements in computational efficiency, making it feasible to simulate larger urban areas. However, further research is needed to optimize the integration and merging processes of these sub-problems to ensure accuracy and consistency across the entire simulation.

In our investigation of the multiMATSim method, the outcomes regarding scalability have been heartening. Although limiting our tests to 4 and 8 nodes doesn't capture the full essence of scalability, the data offers informative and optimistic views on how multiMATSim responds with an increasing number of nodes. These initial outcomes lay the groundwork for deeper dives into how our method scales as we enhance computational resources.

Additionally, as the per-node load grows, we noticed a reliable enhancement in performance. Augmenting the computational nodes while simultaneously upping the MATSim instances for multiMATSim maintains steady performance. Given the pronounced improvements in average plan scores, it's plausible that initiating exchanges earlier might yield a pronounced acceleration. Such outcomes leave us optimistic about achieving better results with heightened load, alluding to the potential of augmented horizontal and vertical scalability. For our forthcoming experiments, the LA 10% scenario emerges as a favorable choice, pressing our system's limits with additional instances, nodes, and optimized parameters.

On a distinct note, while running our framework on two varied CPU architectures - x86 and ARM, we detected subtle behavioral variations. Although the ISA (Instruction Set Architecture) may not be the sole determinant, the differences in core sizes between these CPUs probably have a role. It's pertinent to mention that the A64FX, which is tailored for intensive parallel tasks using innovations

like SVE or HBM memory, might not reach its full potential with a primarily sequential tool like MATSim. Given the fact that MATSim is optimized more for the x86 architecture and the opportunities presented by a JVM library recently launched by Fujitsu, there's an open avenue for additional inquiry. Still, in terms of convergence velocity on Fugaku, multiMATSim manages to surpass MATSim. Our grand vision is to gauge the universality of this Unite and Conquer strategy on alternative multi-agent traffic simulators, such as SUMO or POLARIS[7], underscoring its extensive relevance.

Future research on multiMATSim should focus on increasing the number of instances to further improve the granularity and accuracy of simulations. By running a higher number of simulation instances, the model can better capture the variability in urban transport systems, leading to more robust predictions and exploring a larger solution space, which can potentially enhance performance. Additionally, implementing asynchronous communications between simulation instances could greatly enhance the efficiency and scalability of multiMATSim. Asynchronous communication would allow different instances to exchange information and update their states without waiting for a synchronized global state, thus reducing bottlenecks and improving overall performance.

AI-Based Approach:

The scalability of the AI-based approach also presents challenges. Although the use of Multilayer Perceptron (MLP) neural networks has shown potential in reducing computational time and resources, the fully connected nature of MLPs can become computationally expensive when dealing with large-scale datasets. This limitation is particularly relevant when aiming to simulate entire urban areas with hundreds of thousands of agents.

To address this challenge, future research could explore advanced parallelization techniques for training neural networks. This would involve distributing the

training process across multiple computational nodes, leveraging the capabilities of modern HPC architectures. Additionally, techniques such as model pruning, quantization, and the use of more efficient neural network architectures (e.g., transformers) could further enhance scalability and efficiency.

Generalization and Adaptability

multiMATSim:

As for robustness and adaptability, multiMATSim is presumed to exhibit resilience across various simulation scenarios and configurations. The system's architecture, with its potential for adaptability, hypothetically allows for responsive adjustments to mobility behavior modeling and alterations in simulation parameters. This adaptability is pivotal in accurately simulating the fluid dynamics of urban transport systems, where variability is the norm.

These postulations, while theoretically grounded, underscore the potential of multiMATSim to embody principles of swarm intelligence and distributed problem-solving within a computational context. They suggest that by harnessing the collective capabilities of multiple simulation instances, multiMATSim might not only maintain the integrity of individual agent decisions but also enhance the overall quality and reliability of the simulation outcomes. However, such claims remain speculative and must be substantiated through empirical validation.

AI-Based Approach:

Generalization of AI-based models can be improved through the use of diverse and extensive training datasets that capture a wide range of scenarios. Data augmentation techniques, such as Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs), could be employed to generate synthetic data that mimics real-world variability. Moreover, transfer learning approaches, where models trained on one dataset are fine-tuned on another, could also enhance adapt-

ability.

The flexibility of neural networks allows for responsive adjustments to mobility behavior modeling and alterations in simulation parameters. This adaptability is crucial in accurately simulating the fluid dynamics of urban transport systems, where variability is the norm.

Heuristic vs. AI-Based Approaches

The distinction between heuristic-based and AI-based approaches presents both opportunities and challenges. Heuristic algorithms, like those used in MATSim, incorporate various constraints and objectives to realistically simulate agent behaviors, closely replicating human decision-making processes. In contrast, AI models learn patterns directly from data, which raises questions about their ability to capture the nuanced behaviors modeled by heuristics.

Future research should investigate hybrid approaches that combine the strengths of both heuristic and AI-based methods. For instance, AI models could be used to approximate heuristic solutions, providing a balance between computational efficiency and behavioral accuracy. Additionally, explainable AI techniques could be developed to enhance the interpretability of AI models, making them more transparent and trustworthy for real-world applications.

8.2.2 Future Research Directions

Future Research Directions for multiMATSim

Future research on multiMATSim should focus on several key areas to further enhance its scalability, performance, and general applicability:

1. **Increasing the Number of Instances:** Expanding the number of simulation instances can significantly improve the granularity and accuracy of

simulations. By running more instances, multiMATSim can better capture the variability inherent in urban transport systems, leading to more robust predictions. This increased granularity allows for a more detailed exploration of the solution space, potentially uncovering new insights into urban mobility patterns and system behaviors.

2. **Implementing Asynchronous Communications:** Introducing asynchronous communication between simulation instances could greatly enhance the efficiency and scalability of multiMATSim. Asynchronous communication allows different instances to exchange information and update their states without waiting for a synchronized global state, thus reducing bottlenecks and improving overall performance. This approach can make the system more responsive and capable of handling larger, more complex simulations.

3. **Exploring Larger Scenarios:** Testing multiMATSim with different and larger scenarios will push the system's limits and provide valuable insights into its performance under increased loads. This involves not only increasing the number of computational nodes but also optimizing parameters to handle the added complexity effectively. Exploring larger scenarios can help identify potential scalability issues and provide a more comprehensive understanding of the system's capabilities.

4. **Optimizing Integration and Merging Processes:** Further research is needed to refine the processes of integrating and merging sub-problems within multiMATSim. Ensuring accuracy and consistency across the entire simulation is crucial for reliable outcomes. Enhancing these processes will involve developing more sophisticated algorithms for combining results from different instances, thereby improving the overall fidelity of the simulation.

5. **Investigating Different CPU Architectures:** Running multiMATSim

on various CPU architectures, including ARM-based processors like Amazon's Graviton 3, can provide valuable insights into how different hardware configurations affect performance. Differences in core sizes and instruction set architectures, such as those between x86 and ARM, can influence simulation efficiency. By thoroughly understanding these variations, researchers can optimize multiMATSim's performance by tailoring its operations to leverage the strengths of each architecture. This approach can guide the selection of optimal hardware setups for large-scale simulations and ensure that the system takes full advantage of specific optimizations for parallel tasks.

6. **Generalizing the Unite and Conquer Strategy:** Applying the Unite and Conquer method to other multi-agent traffic simulators can demonstrate the versatility and broader applicability of this approach. This will help in understanding how well the principles of multiMATSim translate across different simulation platforms. Successful application to multiple platforms would underscore the robustness of the Unite and Conquer strategy and its potential for widespread use in traffic simulation.

7. **Early Exchange Initiation:** Investigating the impact of initiating exchanges earlier in the simulation process could yield substantial performance improvements. By experimenting with different timings for these exchanges, researchers can determine the most effective strategies for accelerating convergence and improving average plan scores. Early exchanges might facilitate quicker adjustments and more efficient use of computational resources, enhancing the overall speed and accuracy of the simulation.

8. **Machine Learning for Plan Exchanges:** Currently, the criterion for daily plan exchanges between instances is based on a mathematical calcula-

tion of standard deviation relative to the scores. This method could benefit from a machine learning-based approach. By leveraging machine learning algorithms, it may be possible to develop more sophisticated and adaptive criteria for plan exchanges, enhancing the overall efficiency and accuracy of the simulation. Machine learning can identify patterns and correlations that traditional methods might miss, leading to more effective and timely exchanges.

By addressing these research directions, multiMATSim can be further refined and optimized, paving the way for more efficient and accurate multi-agent traffic simulations. These advancements will not only improve the tool's performance but also expand its applicability to a wider range of scenarios and simulation platforms, enhancing its utility for urban planners and researchers.

Future Research Directions for the AI-Based Approach

1. **Exploration of Alternative Neural Network Architectures:** Beyond MLP networks, other neural network architectures hold promise for traffic simulation. Transformers, with their ability to handle long-range dependencies and parallel processing capabilities, could offer significant improvements in efficiency and scalability. Convolutional Neural Networks (CNNs), known for capturing spatial relationships, could be adapted to model traffic flow and interactions within a geographical context. Graph Neural Networks (GNNs) represent another exciting avenue, as they can model complex relationships and interactions between agents and infrastructure in a transportation network. By representing traffic networks as graphs, GNNs can naturally capture the connectivity and dependencies between different parts of the network, potentially leading to more accurate and robust simulations.

2. **Integration of Real-Time Data and Feedback Mechanisms:** Incorporating real-time data into traffic simulations could significantly enhance their accuracy and relevance. Future work could explore the integration of data from various sources, such as traffic sensors, GPS devices, and social media, to create dynamic and responsive simulation models. Feedback mechanisms could also be developed to allow simulations to adapt in real-time based on observed traffic conditions and patterns.

3. **Enhanced Data Augmentation and Synthetic Data Generation:** As previously mentioned, data augmentation and synthetic data generation are crucial for improving model generalization and performance. Future research could focus on developing more sophisticated techniques for generating realistic synthetic data, ensuring that it captures the complexity and variability of real-world traffic scenarios. This could involve using GANs, VAEs, and other advanced generative models to create diverse and representative datasets for training AI models.

4. **Evaluation and Validation Frameworks:** Robust evaluation and validation frameworks are essential for assessing the performance and reliability of traffic simulation models. Future work should develop standardized benchmarks and metrics for comparing different models and approaches. These frameworks should also incorporate uncertainty quantification methods to provide confidence intervals and probabilistic assessments of model predictions.

5. **Policy and Ethical Considerations:** The deployment of AI-based traffic simulation models in real-world applications raises important policy and ethical considerations. Future research should address issues related to data privacy, algorithmic bias, and the transparency of AI models. Ensuring that

these models are used responsibly and ethically will be crucial for gaining public trust and maximizing their societal benefits.

.

8.2.3 Conclusion

In conclusion, this dissertation proposes new approaches for enhancing multi-agent traffic simulations through the application of HPC and AI techniques. While the results are promising, they also highlight several challenges and opportunities for future research. By addressing these challenges and exploring new directions, we can develop more efficient, accurate, and adaptable traffic simulation models that better meet the needs of modern urban mobility systems. Our work lays the foundation for future advancements in this rapidly evolving field, contributing to the development of more sustainable and intelligent transportation solutions.

Bibliography

[1] Ali R. Abdellah and Andrey Koucheryavy. "Deep Learning with Long Short-Term Memory for IoT Traffic Prediction". In: *Internet of Things, Smart Spaces, and Next Generation Networks and Systems: 20th International Conference, NEW2AN 2020, and 13th Conference, RuSMART 2020, St. Petersburg, Russia, August 26–28, 2020, Proceedings, Part I*. <conf-loc content-type="InPerson">St. Petersburg, Russia</conf-loc>: Springer-Verlag, 2020, pp. 267–280. ISBN: 978-3-030-65725-3. DOI: 10.1007/978-3-030-65726-0_24. URL: https://doi.org/10.1007/978-3-030-65726-0_24.

[2] Andrés Acosta, Jairo Espinosa Oviedo, and Jorge Espinosa Oviedo. "Distributed Simulation in SUMO Revisited: Strategies for Network Partitioning and Border Edges Management". In: May 2016.

[3] Yazed Alsaawy et al. "A Comprehensive and Effective Framework for Traffic Congestion Problem Based on the Integration of IoT and Data Analytics". In: *Applied Sciences* 12.4 (2022). ISSN: 2076-3417. DOI: 10.3390/app12042043. URL: https://www.mdpi.com/2076-3417/12/4/2043.

[4] Pablo Alvarez Lopez et al. "Microscopic Traffic Simulation using SUMO". In: *IEEE Intelligent Transportation Systems Conference (ITSC)*. 2018.

[5] Tarique Anwar et al. "Tracking the Evolution of Congestion in Dynamic Urban Road Networks". In: *Proceedings of the 25th ACM International on Con-*

ference on Information and Knowledge Management. CIKM '16. Indianapolis, Indiana, USA: Association for Computing Machinery, 2016, pp. 2323–2328. ISBN: 9781450340731. DOI: 10.1145/2983323.2983688. URL: https://doi.org/10.1145/2983323.2983688.

[6] Argonne National Laboratory. *Aurora Supercomputer*. Accessed: 2024-06-17. 2024. URL: https://www.anl.gov/aurora.

[7] J. Auld et al. "POLARIS: Agent-based modeling framework development and implementation for integrated travel demand and network and operations simulations". In: *Transportation Research Part C: Emerging Technologies* (2016). DOI: 10.1016/j.trc.2015.07.017.

[8] P.G. Balaji, X. German, and D. Srinivasan. "Urban traffic signal control using reinforcement learning agents". In: *IET Intelligent Transport Systems* 4.3 (Sept. 2010), pp. 177–188. ISSN: 1751-956X. DOI: 10.1049/iet-its.2009.0096. URL: https://digital-library.theiet.org/content/journals/10.1049/iet-its.2009.0096.

[9] Lala Bhaskar et al. "Intelligent traffic light controller using inductive loops for vehicle detection". In: *2015 1st International Conference on Next Generation Computing Technologies (NGCT)*. 2015, pp. 518–522. DOI: 10.1109/NGCT.2015.7375173.

[10] Taisuke Boku et al. "Cygnus - World First Multihybrid Accelerated Cluster with GPU and FPGA Coupling". In: *Workshop Proceedings of the 51st International Conference on Parallel Processing*. ICPP Workshops '22. <conf-loc>, <city>Bordeaux</city>, <country>France</country>, </conf-loc>: Association for Computing Machinery, 2023. ISBN: 9781450394451. DOI: 10.1145/3547276.3548629. URL: https://doi.org/10.1145/3547276.3548629.

[11] Center for Computational Sciences, University of Tsukuba. *Pegasus Supercomputer*. Accessed: 2024-06-17. 2023. URL: https://www.ccs.tsukuba.ac.jp/wp-content/uploads/sites/14/Pegasus.pdf.

[12] Ameni Chetouane, Sabra Mabrouk, and Mohamed Mosbah. "Traffic Congestion Detection: Solutions, Open Issues and Challenges". In: *Distributed Computing for Emerging Smart Networks*. Ed. by Imen Jemili and Mohamed Mosbah. Cham: Springer International Publishing, 2020, pp. 3–22. ISBN: 978-3-030-65810-6.

[13] RIKEN Center for Computational Science. *Fugaku Supercomputer*. Accessed: 2024-06-17. 2024. URL: https://www.r-ccs.riken.jp/en/fugaku/.

[14] MATSim Scenarios Contributors. *MATSim Berlin Scenario*. https://github.com/matsim-scenarios/matsim-berlin. Accessed: 2024-06-20. 2024.

[15] MATSim Scenarios Contributors. *MATSim Los Angeles Scenarios*. https://github.com/matsim-scenarios/matsim-los-angeles. Accessed: 2024-06-20. 2024.

[16] MATSim Scenarios Contributors. *MATSim NYC Scenario*. https://github.com/matsim-scenarios/matsim-nyc. Accessed: 2024-06-20. 2024.

[17] Fei Dai et al. "Spatio-Temporal Deep Learning Framework for Traffic Speed Forecasting in IoT". In: *IEEE Internet of Things Magazine* 3.4 (2020), pp. 66–69. DOI: 10.1109/IOTM.0001.2000031.

[18] Shima Damadam et al. "An Intelligent IoT Based Traffic Light Management System: Deep Reinforcement Learning". In: *Smart Cities* 5.4 (2022), pp. 1293–1311. ISSN: 2624-6511. DOI: 10.3390/smartcities5040066. URL: https://www.mdpi.com/2624-6511/5/4/66.

[19] Ewa Deelman. *Pegasus: Workflow Management System*. Accessed: 2024-06-17. 2023. URL: https://pegasus.isi.edu/.

[20] Ewa Deelman. *Pegasus: Workflow Management System*. Accessed: 2024-06-17. 2023. URL: https://pegasus.isi.edu/about/.

[21] Ewa Deelman et al. "Pegasus: A Framework for Mapping Complex Scientific Workflows onto Distributed Systems". In: *Scientific Programming* 13 (Jan. 2005), pp. 219–237. DOI: 10.1155/2005/128026.

[22] Nahid Emad and Serge Petiton. "Unite and Conquer approach for high scale numerical computing". In: *International Journal of Computational Science and Engineering* 14 (2016). hal-01609342, pp. 5–14. DOI: 10.1016/j.jocs.2016.01.007.

[23] Nahid Emad, S.-A. Shahzadeh-Fazeli, and Jack Dongarra. "An asynchronous algorithm on the NetSolve global computing system". In: *Future Generation Computer Systems* 22.3 (2006), pp. 279–290. ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2005.10.003. URL: https://www.sciencedirect.com/science/article/pii/S0167739X05001378.

[24] Martin Fellendorf and Peter Vortisch. "Microscopic traffic flow simulator VISSIM". In: June 2011, pp. 63–93. ISBN: 978-1-4419-6141-9. DOI: 10.1007/978-1-4419-6142-6_2.

[25] Fujitsu. *A64FX® Microarchitecture Manual (English)*. Accessed: 2024-06-17. 2022. URL: https://github.com/fujitsu/A64FX/blob/master/doc/A64FX_Microarchitecture_Manual_en_1.8.pdf.

[26] P. G. Gipps. "A behavioural car-following model for computer simulation". In: *Transportation Research Part B: Methodological* 15.2 (1981), pp. 105–111. DOI: 10.1016/0191-2615(81)90037-0.

[27] Emmanuel Hermelin, Fabien Michel, and Jacques Ferber. "Etat de l'art sur les simulations multi-agents et le GPGPU". In: *Revue d'intelligence artificielle* 29 (Aug. 2015), pp. 425–451. DOI: 10.3166/ria.29.425-451.

[28] Sebastian Hörl. "Exploring accelerated evolutionary parameter search for iterative large-scale transport simulations in a new calibration testbed". In: June 2022.

[29] Sebastian Hörl, Felix Becker, and Kay W. Axhausen. "Simulation of price, customer behaviour and system impact for a cost-covering automated taxi system in Zurich". In: *Transportation Research Part C: Emerging Technologies* 123 (2021), p. 102974. ISSN: 0968-090X. DOI: https://doi.org/10.1016/j.trc.2021.102974. URL: https://www.sciencedirect.com/science/article/pii/S0968090X21000115.

[30] A Horni and K Nagel. "More About Configuring MATSim". In: *The Multi-Agent Transport Simulation MATSim*. Ed. by A Horni, K Nagel, and K W Axhausen. License: CC-BY 4.0. London: Ubiquity Press, 2016, pp. 35–44. DOI: http://dx.doi.org/10.5334/baw.4.

[31] A Horni, K Nagel, and K W Axhausen. "Introducing MATSim". In: License: CC-BY 4.0. London: Ubiquity Press, 2016. Chap. Introducing MATSim, pp. 3–8. DOI: http://dx.doi.org/10.5334/baw.1.

[32] A. Horni and K. Nagel. "More About Configuring MATSim". In: *The Multi-Agent Transport Simulation MATSim*. Ed. by A. Horni, K. Nagel, and K. W. Axhausen. London: Ubiquity Press, 2016, pp. 35–44. DOI: 10.5334/baw.4.

[33] A. Jackson et al. "Investigating Applications on the A64FX". In: *2020 IEEE International Conference on Cluster Computing (CLUSTER)*. Los Alamitos, CA, USA: IEEE Computer Society, Sept. 2020, pp. 549–558. DOI: 10.1109/CLUSTER49012.2020.00078. URL: https://doi.ieeecomputersociety.org/10.1109/CLUSTER49012.2020.00078.

[34] Lejun Jiang, Tamás G. Molnár, and Gábor Orosz. "On the deployment of V2X roadside units for traffic prediction". In: *Transportation Research Part*

C: Emerging Technologies 129 (2021), p. 103238. ISSN: 0968-090X. DOI: https://doi.org/10.1016/j.trc.2021.103238. URL: https://www.sciencedirect.com/science/article/pii/S0968090X21002515.

[35] George Karypis and Vipin Kumar. *A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs*. Vol. 20. SIAM Journal on Scientific Computing 1. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1998, pp. 359–392. DOI: 10.1137/S1064827595287997.

[36] George Karypis and Vipin Kumar. *hMETIS 1.5: A Hypergraph Partitioning Package*. Available at http://www-users.cs.umn.edu/~karypis/metis/hmetis/. 1999.

[37] Shilpa Khedkar, R. Canessane, and Moslem Lari Najafi. "Prediction of Traffic Generated by IoT Devices Using Statistical Learning Time Series Algorithms". In: *Wireless Communications and Mobile Computing* 2021 (Aug. 2021). DOI: 10.1155/2021/5366222.

[38] Juan J. Lamas-Seco et al. "SiDIVS: Simple Detection of Inductive Vehicle Signatures with a Multiplex Resonant Sensor". In: *Sensors (Basel)* 16.8 (Aug. 2016), p. 1309. DOI: 10.3390/s16081309.

[39] Tal Laor and Yair Galily. "In WAZE we trust? GPS-based navigation application users' behavior and patterns of dependency". In: *PLOS ONE* 17.11 (Nov. 2022), pp. 1–17. DOI: 10.1371/journal.pone.0276449. URL: https://doi.org/10.1371/journal.pone.0276449.

[40] M. J. Lighthill and G. B. Whitham. "On Kinematic Waves. II. A Theory of Traffic Flow on Long Crowded Roads". In: *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences* 229.1178 (1955), pp. 317–345. DOI: 10.1098/rspa.1955.0089.

[41] Chang Liu et al. "Dynamic Road Traffic Network Division for Real-Time Management and Control". In: *Journal of Physics: Conference Series* 2491.1 (2023), p. 012005. DOI: 10.1088/1742-6596/2491/1/012005.

[42] Qiang Liu, Qing Wang, and Shu-an Liu. "An Improved Sub-Networks Partitioning Method for Urban Traffic Networks". In: *2019 Chinese Control And Decision Conference (CCDC)*. 2019, pp. 6405–6410. DOI: 10.1109/CCDC.2019.8833298.

[43] Matthieu Mastio et al. "Distributed Agent-Based Traffic Simulations". In: *IEEE Intelligent Transportation Systems Magazine* 10 (Jan. 2018). DOI: 10.1109/MITS.2017.2776162.

[44] Scott McQuire. "One map to rule them all? Google Maps as digital technical object". In: *Communication and the Public* 4 (June 2019), pp. 150–165. DOI: 10.1177/2057047319850192.

[45] Mésocentre de l'Université Paris-Saclay. *Plateformes du Mésocentre - Université Paris-Saclay*. 5th October, 2023. 2020. URL: https://mesocentre.universite-paris-saclay.fr/platforms/platforms.html.

[46] Sara Moukir, Nahid Emad, and Stéphane Baudelocq. "A high performance algorithmic variant of MATSim road traffic simulator". In: *IPDPSW 2023 - IEEE International Parallel and Distributed Processing Symposium Workshops*. St. Petersburg, United States, May 2023, pp. 914–922. DOI: 10.1109/IPDPSW59300.2023.00149. URL: https://hal.archives-ouvertes.fr/hal-04496008.

[47] Sara Moukir, Nahid Emad, and Stéphane Baudelocq. "A high performance approach with MATSim for traffic road simulation". In: *2022 12th International Congress on Advanced Applied Informatics (IIAI-AAI)*. Kanazawa,

Japan, July 2022, pp. 679–681. DOI: 10.1109/IIAIAAI55812.2022.00140. URL: https://hal.archives-ouvertes.fr/hal-04495989.

[48] Sara Moukir, Nahid Emad, and Stéphane Baudelocq. "From MATSim to MultiMATSim: Rethinking Traffic Modeling Using the 'Unite and Conquer' Approach". In: *2023 IEEE International Conference on High Performance Computing Communications, Data Science Systems, Smart City Dependability in Sensor, Cloud Big Data Systems Application (HPCC/DSS/SmartCity/DependSy* 2023, pp. 288–295. DOI: 10.1109/HPCC-DSS-SmartCity-DependSys60770.2023.00047.

[49] Sara Moukir et al. "Advancements in Traffic Simulations with multiMATSim's Distributed Framework". In: *Proceedings of the 16th International Conference on Agents and Artificial Intelligence - Volume 1: ICAART*. INSTICC. SciTePress, 2024, pp. 374–385. ISBN: 978-989-758-680-4. DOI: 10.5220/0012452600003636.

[50] M. E. J. Newman. "Fast algorithm for detecting community structure in networks". In: *Physical Review E* 69.6 (2004), p. 066133. DOI: 10.1103/PhysRevE.69.066133.

[51] Johannes Nguyen et al. "An overview of agent-based traffic simulators". In: *Transportation Research Interdisciplinary Perspectives* 12 (2021), p. 100486. ISSN: 2590-1982. DOI: https://doi.org/10.1016/j.trip.2021.100486.

[52] Sébastien Noël et al. "A Multi-level Scheduler for the Grid Computing YML Framework". In: vol. 4375. Aug. 2006, pp. 87–100. ISBN: 978-3-540-72226-7. DOI: 10.1007/978-3-540-72337-0_9.

[53] Oak Ridge Leadership Computing Facility. *Frontier Supercomputer*. Accessed: 2024-06-17. 2024. URL: https://www.olcf.ornl.gov/frontier/.

[54] University of Paris-Saclay. *Mésocentre de l'Université Paris-Saclay*. Accessed on: June, 2024. 2020. URL: https://mesocentre.universite-paris-saclay.fr/platforms/platforms.html.

[55] Andrei Poenaru et al. "An Evaluation of the Fujitsu A64FX for HPC Applications". English. In: *Cray User Group 2021*. Cray User Group 2021 ; Conference date: 03-05-2021 Through 05-05-2021. May 2021. URL: https://cug.org/cug-2021/.

[56] Tomas Potuzak. "Current Trends in Road Traffic Network Division for Distributed or Parallel Road Traffic Simulation". In: *2022 IEEE/ACM 26th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*. 2022, pp. 77–86. DOI: 10.1109/DS-RT55542.2022.9932112.

[57] Tomas Potuzak. "Distributed-Parallel Road Traffic Simulator for Clusters of Multi-core Computers". In: *2012 IEEE/ACM 16th International Symposium on Distributed Simulation and Real Time Applications*. 2012, pp. 195–201. DOI: 10.1109/DS-RT.2012.36.

[58] Tomas Potuzak. "Reduction of Inter-process Communication in Distributed Simulation of Road Traffic". In: *2020 IEEE/ACM 24th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*. 2020, pp. 1–10. DOI: 10.1109/DS-RT50469.2020.9213673.

[59] UVSQ PRiSM Laboratory. *YML: Workflow Programming Environment*. Accessed: 2024-06-17. 2023. URL: http://yml.prism.uvsq.fr/.

[60] Steven F. Railsback and Volker Grimm. *Agent-Based and Individual-Based Modeling: A Practical Introduction*. Princeton, NJ: Princeton University Press, 2011. ISBN: 9780691136745.

[61] Daniel Rajf and Tomas Potuzak. "Comparison of Road Traffic Simulation Speed on CPU and GPU". In: *2019 IEEE/ACM 23rd International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*. 2019, pp. 1–8. DOI: 10.1109/DS-RT47707.2019.8958702.

[62] Kotagiri Ramamohanarao et al. "SMARTS: Scalable Microscopic Adaptive Road Traffic Simulator". In: *ACM Transactions on Intelligent Systems and Technology* 8 (Dec. 2016), pp. 1–22. DOI: 10.1145/2898363.

[63] P. I. Richards. "Shock Waves on the Highway". In: *Operations Research* 4.1 (1956), pp. 42–51. DOI: 10.1287/opre.4.1.42.

[64] Aleksandr Saprykin, Ndaona Chokani, and Reza S. Abhari. "GEMSim: A GPU-accelerated multi-modal mobility simulator for large-scale scenarios". In: *Simulation Modelling Practice and Theory* 94 (2019), pp. 199–214. ISSN: 1569-190X. DOI: https://doi.org/10.1016/j.simpat.2019.03.002. URL: https://www.sciencedirect.com/science/article/pii/S1569190X19300267.

[65] M. Sato et al. "Co-Design for A64FX Manycore Processor and "Fugaku"". In: *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*. Atlanta, GA, USA, 2020, pp. 1–15. DOI: 10.1109/SC41405.2020.00051.

[66] Björn Schünemann. "V2X simulation runtime infrastructure VSimRTI: An assessment tool to design smart traffic management systems". In: *Computer Networks* 55.14 (2011). Deploying vehicle-2-x communication, pp. 3189–3198. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2011.05.005. URL: https://www.sciencedirect.com/science/article/pii/S1389128611001605.

[67] Z. Shen et al. "GPU Based Genetic Algorithms for the Dynamic Sub-area Division Problem of the Transportation System". In: *IFAC Proceedings Volumes* 47.3 (2014). 19th IFAC World Congress, pp. 5115–5120. ISSN: 1474-6670. DOI: https://doi.org/10.3182/20140824-6-ZA-1003.02693. URL: https://www.sciencedirect.com/science/article/pii/S1474667016424080.

[68] David Strippgen and Kai Nagel. "Using common graphics hardware for multi-agent traffic simulation with CUDA". In: Jan. 2009, p. 62. DOI: 10.1145/1537614.1537693.

[69] TOP500. *TOP500 List - June 2024*. Accessed: 2024-06-17. 2024. URL: https://top500.org/lists/top500/2024/06/.

[70] Chanaka Withanage et al. "A modified multilevel k-way partitioning algorithm for trip-based road networks". In: *MATEC Web of Conferences* 272 (Jan. 2019), p. 01038. DOI: 10.1051/matecconf/201927201038.

[71] Xinzhe Wu and Serge G. Petiton. "A distributed and parallel asynchronous unite and conquer method to solve large scale non-Hermitian linear systems with multiple right-hand sides". In: *Parallel Computing* 89 (2019), p. 102551. ISSN: 0167-8191. DOI: https://doi.org/10.1016/j.parco.2019.102551. URL: https://www.sciencedirect.com/science/article/pii/S0167819119301425.

[72] Yadong Xu, Heiko Aydt, and Michael Lees. "SEMSim: A Distributed Architecture for Multi-scale Traffic Simulation". In: *2012 ACM/IEEE/SCS 26th Workshop on Principles of Advanced and Distributed Simulation*. 2012, pp. 178–180. DOI: 10.1109/PADS.2012.40.

[73] Yadong Xu et al. "A Graph Partitioning Algorithm for Parallel Agent-Based Road Traffic Simulation". In: May 2017, pp. 209–219. DOI: 10.1145/3064911.3064914.

[74] Yadong Xu et al. “Relaxing Synchronization in Parallel Agent-Based Road Traffic Simulation”. In: *ACM Transactions on Modeling and Computer Simulation* 27 (May 2017), pp. 1–24. DOI: 10.1145/2994143.

[75] Yan Xu and Gary Tan. “An Offline Road Network Partitioning Solution in Distributed Transportation Simulation”. In: *2012 IEEE/ACM 16th International Symposium on Distributed Simulation and Real Time Applications*. 2012, pp. 210–217. DOI: 10.1109/DS-RT.2012.38.

[76] Eduard Zadobrischi. “Intelligent Traffic Monitoring through Heterogeneous and Autonomous Networks Dedicated to Traffic Automation”. In: *Sensors* 22.20 (2022). ISSN: 1424-8220. DOI: 10.3390/s22207861. URL: https://www.mdpi.com/1424-8220/22/20/7861.

Résumé

1 Introduction et contexte

La modélisation multi-agent appliquée au trafic routier repose sur la simulation du comportement individuel de chaque agent (véhicules, piétons, cyclistes, etc.) dans un réseau de transport. Chaque agent prend des décisions autonomes en fonction de son environnement et de ses interactions avec d'autres agents, reproduisant ainsi des dynamiques réalistes de flux de trafic. Cette approche rend possible la capture de la complexité des systèmes de transport en simulant des scénarios à grande échelle, tout en tenant compte des choix de transport individuels et des activités quotidiennes des usagers. Parmi les simulateurs de trafic multi-agent, SUMO (Simulation of Urban MObility) est l'un des plus utilisés. Il permet de simuler le trafic à l'échelle urbaine avec une grande précision, incluant des modèles pour les véhicules, les transports publics et les piétons. SUMO est open-source et est conçu pour être extensible, bien que sa conception majoritairement séquentielle limite ses performances pour les simulations très complexes. Un autre simulateur populaire est PTV Vissim, un logiciel propriétaire qui offre une modélisation détaillée des interactions entre différents modes de transport. Il est souvent utilisé dans la planification urbaine pour simuler l'impact des infrastructures sur le trafic, mais son architecture essentiellement séquentielle peut également ralentir les simulations à grande échelle. Enfin, MATSim (Multi-Agent Transport Simulation) est l'un des simulateurs de trafic multi-agent les plus cités dans la littérature. Il offre la possibilité de simuler les activités quotidiennes et les choix de transport des agents dans un réseau donné, tout en étant hautement flexible et extensible. Cependant, comme les autres simulateurs mentionnés, sa conception essentiellement séquentielle limite ses performances dans les simulations à très grande échelle.

L'objectif de cette thèse est de répondre à ces limites des simulateurs de trafic multi-agents actuels, en particulier leur manque de scalabilité et leur incapacité à exploiter pleinement les architectures HPC (High-Performance Computing) modernes. Ces simulateurs, bien qu'essentiels pour comprendre et gérer les flux de trafic, voient leur utilité limitée par leurs performances lorsqu'ils sont appliqués à des simulations de très grande envergure ou nécessitant des temps de réponse courts. À mesure que les infrastructures urbaines se développent et que les systèmes de transport deviennent plus interconnectés

et dynamiques (notamment avec l'arrivée de véhicules autonomes), il est de plus en plus nécessaire de disposer d'outils capables de simuler ces environnements complexes de manière plus rapide et plus efficace. Cette exigence est particulièrement importante dans des domaines tels que les villes intelligentes, où la prise de décision en temps réel et l'adaptation rapide aux conditions de trafic sont essentielles. Afin de répondre à ces enjeux, cette thèse propose deux contributions principales visant à améliorer la performance et la scalabilité des simulateurs de trafic multi-agents. La première contribution repose sur l'adaptation de la méthode Unite and Conquer (UC), une approche initialement développée pour résoudre des problèmes d'algèbre linéaire à grande échelle, aux simulations de trafic multi-agents. Cette méthode facilite l'exploration efficace de l'espace des solutions et la répartition de la charge de calcul entre plusieurs unités, et rend possible une exécution distribuée et parallèle des simulations. En exploitant le parallélisme multiniveau et des communications synchrones et/ou asynchrones entre nœuds, cette approche offre une vitesse de convergence plus importante par rapport aux systèmes originels. La deuxième contribution repose sur l'utilisation de l'intelligence artificielle, et plus précisément sur l'intégration de réseaux de neurones Multilayer Perceptrons (MLP) pour compresser le processus de simulation. Cette approche aide à réduire la complexité du problème en prédisant les résultats d'une simulation à partir d'un ensemble réduit de variables d'entrée, offrant ainsi une solution plus rapide et efficace, particulièrement adaptée aux applications nécessitant des réponses en temps réel. Finalement, cette thèse vise à proposer des solutions combinant des concepts liés au HPC et à l'IA pour surmonter les limitations des simulateurs de trafic actuels, en introduisant des méthodes capables de réduire considérablement le temps et les ressources nécessaires pour simuler des systèmes de transport de plus en plus complexes.

2 Première contribution : multiMATSim

La première contribution de cette thèse porte sur l'adaptation de la méthode Unite and Conquer pour améliorer les performances des simulations de trafic multi-agents. L'idée centrale de la méthode UC est d'explorer de manière exhaustive les espaces de solutions complexes en s'appuyant sur le calcul parallèle distribué. Contrairement à l'approche classique du Divide and Conquer, qui divise les problèmes en sous-problèmes indépendants, UC traite le problème dans son ensemble à travers plusieurs instances traitant chacune une méthode de résolution différente, maximisant ainsi l'utilisation des ressources de calcul disponibles. Cela permet de maintenir un haut niveau de détail dans les simulations tout en réduisant les temps de calcul grâce à l'exploitation des architectures HPC. Dans les simulations de trafic multi-agents, l'approche Unite and Conquer se révèle partic-

ulièrement adaptée pour gérer la volumétrie importante de données générée par l'activité des agents. Chaque agent produit des interactions complexes et dynamiques, qui doivent être traitées de manière efficace et rigoureuse dans le cadre d'une simulation distribuée.

L'un des points forts de l'approche UC réside dans la coordination des calculs distribués entre plusieurs nœuds de calcul. Grâce à cette coordination, les différentes instances de la simulation peuvent échanger des informations, et assurent ainsi la cohérence des résultats à l'échelle globale. Les communications entre les nœuds de calcul, qu'elles soient synchrones ou asynchrones, permettent de s'adapter aux besoins spécifiques de chaque scénario, et de garantir que les dynamiques des agents soient mises à jour de manière continue et précise.

En plus du calcul inter-nœuds évoqués, l'approche UC permet également un calcul intra-nœud, en exploitant les processeurs multi-cœurs de chaque unité. La combinaison de calculs inter et intra-nœuds confère à l'approche UC un parallélisme multiniveau, qui maximise l'efficacité des ressources matérielles. Elle permet de simuler le trafic à grande échelle, impliquant des milliers, voire des millions d'agents, avec des temps de simulation optimisés.

Dans l'application à MATSim, chaque instance du simulateur, exécutée sur un nœud, partage des informations sur les plans des agents pour assurer la cohérence globale. Les échanges peuvent être faits via des communications synchrones, avec des mises à jour régulières, ou asynchrones, et offrent ainsi une plus grande flexibilité dans la gestion des calculs en fonction des besoins du scénario.

Les simulations à grande échelle, notamment sur des systèmes distribués, sont souvent exposées à des pannes matérielles ou des interruptions de communication. UC intègre donc des mécanismes de tolérance aux pannes, pour assurer la continuité des calculs même en cas de défaillance de certains nœuds, sans perte de données ni besoin de redémarrer la simulation.

La combinaison de ces propriétés favorise l'exploitation quasi-exhaustive de l'ensemble des solutions possibles.

Les tests effectués avec multiMATSim, une version de MATSim combinée à l'approche UC, ont montré des gains importants en termes de temps de calcul et de scalabilité. Lors de simulations impliquant des milliers d'agents dans de vastes réseaux urbains, UC a permis de réduire les temps de calcul d'un facteur de 4 à 8 selon la taille des ensembles de données et proportionnellement au nombre de nœuds utilisés. Aussi, cette méthode a démontré une bonne scalabilité, aussi bien verticale qu'horizontale, avec une amélioration continue des performances à mesure que la charge de données augmentait. Nous avons sélectionné le scénario de Los Angeles en utilisant 0,1% et 1 % de la population totale, correspondant respectivement à environ 19 000 et 191 000 agents. Ces scénarios ont été choisis pour tester la robustesse et la scalabilité de notre méthode sur des populations de tailles différentes, tout en maintenant un niveau élevé de détail dans les simulations. Pour ces tests, nous avons utilisé un nombre variable de nœuds de calcul, à savoir 4 et 8 nœuds,

ce qui correspond à 4 et 8 instances de MATSim exécutées en parallèle. MATSim étant un programme itératif, il prend en entrée un plan journalier pour chaque agent. Chaque itération consiste en une simulation de 24 heures pour l'ensemble des agents, après laquelle les plans sont ajustés en fonction de leur performance. Le processus continue jusqu'à ce qu'un équilibre soit atteint, où les plans des agents sont optimisés et stables, reflétant un usage optimal du réseau de transport. Ces plans journaliers proviennent de diverses sources, telles que des données d'enquêtes de mobilité ou des modèles synthétiques de population. Il est souvent nécessaire de procéder à plusieurs centaines d'itérations pour que le modèle commence à converger vers une solution stable. Ce processus est extrêmement coûteux en termes de temps de calcul, notamment pour les scénarios de grande envergure. Par exemple, pour le scénario de Los Angeles avec 0.1 % de la population totale, l'approche traditionnelle de MATSim peut nécessiter jusqu'à 6 heures pour commencer à converger, dans nos conditions expérimentales. Ce temps de convergence élevé est dû à la nature itérative de MATSim, où chaque itération affine progressivement les plans des agents en fonction de leurs interactions avec le réseau de transport et les autres agents. Cependant, en utilisant multiMATSim, avec 8 nœuds de calcul et après seulement un échange de solutions intermédiaires entre les nœuds, nous avons observé une convergence après seulement 45 minutes, soit un facteur d'accélération (speed-up) de 8 par rapport à l'approche traditionnelle. Avec 4 nœuds de calcul, le temps de convergence a été réduit à environ 90 minutes, ce qui correspond à un speed-up de 4. Bien que ces résultats soient prometteurs, il est important de noter que nous ne pouvons pas attribuer directement et systématiquement cette propriété de proportionnalité à l'augmentation du nombre de nœuds. En effet, une évaluation plus approfondie nécessiterait un plus grand nombre d'expérimentations avec différentes configurations de calcul et des scénarios plus diversifiés. Néanmoins, ces premiers résultats ont mis en évidence la possibilité d'un speed-up proportionnel au nombre de nœuds, ouvrant la voie à de futures recherches. De plus, nos expérimentations ont montré des résultats encore plus encourageants avec le scénario à 1 % de la population totale, soit avec environ 191 000 agents. En effet, nous avons observé un speed-up légèrement supérieur à celui des simulations avec un plus petit nombre d'agents. Ces résultats sont particulièrement intéressants car ils démontrent la capacité de multiMATSim à non seulement gérer un grand nombre d'agents, mais également à bénéficier d'une amélioration en termes de scalabilité lorsque la taille de la population simulée augmente. Ces résultats ouvrent de nouvelles perspectives pour l'utilisation de l'approche Unite and Conquer appliquée à MATSim, et à d'autres simulateurs de trafic routier multi-agent.

À l'avenir, il serait pertinent d'augmenter encore le nombre de nœuds de calcul et d'instances de MATSim afin de tester cette méthode avec des scénarios encore plus conséquents, tels que des simulations à 100 % de la population de Los Angeles ou Berlin. En augmentant le nombre d'agents et d'instances en parallèle, nous pourrions potentielle-

ment atteindre des niveaux de performance encore plus élevés et explorer des solutions jusqu'ici inaccessibles avec des méthodes traditionnelles. Cela permettrait aussi d'évaluer la robustesse de l'approche dans des environnements à très grande échelle, où les interactions entre agents et les ajustements dynamiques du réseau de transport deviendraient encore plus complexes.

Enfin, cette méthode se veut générique et vise à être applicable à d'autres simulateurs de trafic multi-agents. MATSim a été choisi comme premier champ d'application en raison de sa pertinence pour les simulations de trafic multi-agent et multimodales à grande échelle.

3 Deuxième contribution : une méthode basée sur l'IA pour la simulation de trafic multi-agent

La deuxième contribution de cette thèse repose sur l'utilisation de l'intelligence artificielle, spécifiquement les réseaux de neurones Multilayer Perceptrons, pour compresser le processus de simulation de trafic multi-agent et accélérer l'obtention des résultats. Contrairement à la méthode Unite and Conquer, qui vise à augmenter et paralléliser les calculs pour explorer de manière exhaustive l'espace des solutions, cette approche repose sur l'apprentissage automatique pour prédire les résultats d'une simulation à partir d'un ensemble de données d'entrée. En modélisant les comportements d'agents via des MLP, l'objectif est de réduire considérablement la complexité et le temps de calcul nécessaire pour atteindre des résultats suffisamment précis.

Pour cette deuxième approche, nous nous sommes également basés sur MATSim, tout comme pour la première contribution. L'utilisation répétée de MATSim dans cette thèse s'explique non seulement par sa large adoption dans la communauté scientifique des simulations multi-agents, mais aussi par son aptitude à fournir des résultats rigoureux dans des scénarios complexes. En choisissant MATSim comme premier champ d'application de nos approches, nous avons pu garantir une certaine continuité dans nos expériences. Toutefois, tout comme la première contribution, cette approche se veut générique et est conçue pour être applicable à d'autres simulateurs de trafic multi-agents.

Dans cette deuxième contribution, nous avons mis en place un proof of concept. Pour ce faire, nous avons extrait des scénarios de 1 000 agents à partir de simulations plus vastes, impliquant un plus grand nombre d'agents. Ces scénarios réduits permettent de mener des expérimentations à petite échelle tout en capturant la diversité des comportements observés dans des simulations de plus grande envergure. Dans un premier temps, nous nous sommes concentrés uniquement sur les variables discrètes. Une fois ces scénarios exécutés sur MATSim, ils ont permis de générer 7 000 ensembles de données

de sortie associés aux 7 000 ensembles de données d'entrée correspondants. Ces ensembles de données ont ensuite servi à l'entraînement et à la validation du modèle MLP. La réduction de l'échelle des simulations a été justifiée par la nécessité de tester, dans un premier temps, l'efficacité du modèle prédictif dans un cadre contrôlé. L'objectif était de vérifier si un réseau de neurones pouvait capturer les dynamiques d'un groupe d'agents tout en réduisant drastiquement le temps de calcul. Le modèle MLP développé comprend plusieurs couches entièrement connectées, avec deux couches cachées de 256 neurones chacune, et utilise la fonction d'activation ReLU (Rectified Linear Unit) pour modéliser les relations non linéaires entre les variables d'entrée et de sortie. De plus, un mécanisme de dropout a été intégré avec un taux de 0,1 pour prévenir le surapprentissage. Les sorties du modèle MLP, sous forme de vecteurs continus, ont été arrondies pour générer des valeurs discrètes représentant les modes de transport empruntés par les agents tout au long de la journée. Le processus d'entraînement a été réalisé en minimisant la Mean Squared Error (MSE) entre les valeurs prédites par le MLP et celles issues des simulations effectuées par MATSim. Cette fonction de perte a été choisie en raison de sa capacité à pénaliser fortement les grandes erreurs, afin de garantir une attention particulière aux écarts significatifs dans les prédictions. Nous avons également expérimenté la fonction Huber loss pour mieux gérer les valeurs aberrantes. L'entraînement du modèle s'est effectué sur 50 époques avec une taille de lot de 64, en utilisant l'optimiseur Adam avec un taux d'apprentissage de 0,001. Les résultats montrent une Mean Absolute Error (MAE) de 0,051 sur les données d'entraînement et 0,0524 sur les données de validation, indiquant une bonne précision dans la prédiction des modes de transport utilisés par les agents. Ces résultats confirment que l'approche basée sur les MLP peut offrir des prédictions rapides et précises, avec des temps de calcul considérablement réduits par rapport à une simulation complète réalisée avec MATSim. Le caractère proof of concept de cette expérimentation réside dans la démonstration qu'il est possible d'approximer efficacement une simulation multi-agents de trafic détaillée grâce à un modèle prédictif beaucoup plus rapide, sans trop sacrifier la précision des résultats pour certaines variables discrètes. En utilisant un ensemble conséquent de données d'entrée-sortie, nous avons montré que le modèle MLP pouvait capturer des dynamiques de trafic complexes et fournir des prédictions en un temps presque instantané une fois le modèle entraîné. Les résultats obtenus sont très prometteurs, et ouvrent la voie à des applications plus larges de l'intelligence artificielle pour la simulation de trafic multi-agent. Ces résultats suggèrent qu'il serait tout à fait envisageable de poursuivre sur cette voie en cherchant à générer des variables continues, en plus des variables discrètes, qui jouent toutes deux un rôle central dans les simulations de trafic multi-agents. L'extension à des variables continues permettrait d'élargir considérablement le champ d'application de cette approche et d'atteindre une couverture complète des variables de sortie des simulations. Cela inclurait non seulement les choix de modes de transport des agents, mais aussi d'autres paramètres tels que les dis-

tances parcourues, les durées de trajet, et les interactions complexes entre les agents et l'infrastructure routière. Une telle extension transformerait notre modèle prédictif en un outil capable de générer la totalité des plans ajustés des agents dans une simulation donnée. Ces plans ajustés représentent un résultat clé, car ils décrivent les comportements quotidiens des individus dans un réseau de transport dynamique, en tenant compte des ajustements que ces individus font en fonction de leurs expériences passées, des contraintes temporelles, et des interactions avec les autres agents.

L'étape suivante consisterait à appliquer cette approche à des scénarios de plus grande envergure, avec des centaines de milliers, voire des millions d'agents, répartis sur des zones géographiques étendues. Evaluer le modèle sur de tels scénarios permettrait de vérifier sa robustesse et sa capacité à s'adapter à des environnements plus complexes et variés. Il s'agirait ici de simuler des métropoles entières, où la dynamique du trafic est plus imprévisible, impliquant des interactions massives entre les agents et des structures urbaines hétérogènes. L'intelligence artificielle pourrait jouer un rôle clé dans la gestion de ces grands réseaux urbains, facilitant l'anticipation des flux de trafic et le test rapide des effets de nouvelles infrastructures ou de politiques de transport en un temps record. Une question fondamentale reste cependant en suspens : dans quelle mesure peut-on développer un modèle générique, c'est-à-dire un modèle capable de générer des solutions prédictives fiables indépendamment de la zone géographique simulée, de sa topologie, du comportement de ses habitants, ou du nombre d'agents impliqués ? Cette problématique est essentielle car, actuellement, les simulations de trafic sont souvent calibrées pour des environnements spécifiques, ce qui limite leur portabilité. Le développement d'un tel modèle pourrait transformer la manière dont les simulations de trafic sont utilisées, en permettant aux décideurs et aux urbanistes de tester rapidement des scénarios dans n'importe quelle ville, région ou contexte géographique, sans avoir à passer par des étapes fastidieuses de recalibrage. Cela soulève également la question de la capacité des modèles d'IA à s'adapter et à apprendre en continu à partir de nouvelles données. Une des pistes de recherche prometteuses serait d'introduire des mécanismes d'apprentissage dynamique dans les modèles d'IA, où le modèle pourrait s'améliorer au fur et à mesure qu'il reçoit de nouvelles informations provenant de la réalité, par exemple à travers des capteurs ou des données en temps réel. Cela offrirait la possibilité de créer des simulations adaptatives, capables de s'ajuster au fur et à mesure que les conditions de trafic évoluent, offrant ainsi une représentation plus précise et plus réactive des réseaux de transport urbains.
Finalement, l'utilisation de l'intelligence artificielle pour la simulation de trafic multi-agents, à travers des approches comme les réseaux de neurones MLP, présente un potentiel considérable pour transformer ce domaine. À mesure que les méthodes d'IA continuent de se développer, elles pourront aider à surmonter les limitations actuelles des simula-

tions basées sur des modèles heuristiques ou purement déterministes, en apportant des solutions plus flexibles, rapides et généralisables. Bien que de nombreuses problématiques techniques et théoriques subsistent, ces premiers résultats constituent une base solide pour la poursuite de recherches dans cette direction. L'ambition ultime serait de parvenir à un système de simulation intelligent, capable de prédire et d'optimiser les flux de transport de manière efficace, pour toute ville ou région, en s'adaptant constamment aux changements dans l'environnement et dans les comportements des utilisateurs.
Comme précisé précedemment, même si cette méthode a été pratiquée principalement avec MATSim; elle se veut générique et adaptable à d'autres simulateurs multi-agents. MATSim a servi ici comme plateforme initiale en raison de sa capacité à générer des données complexes et diversifiées, mais l'ambition à terme est d'étendre cette méthodologie à des environnements de simulation variés, en démontrant que l'intelligence artificielle peut offrir des solutions efficaces pour accélérer et simplifier les simulations de trafic, tout en maintenant des niveaux élevés de précision et de scalabilité.

4 Conclusions et perspectives

Cette thèse a exploré deux approches complémentaires pour résoudre les problèmes de scalabilité et d'efficacité dans les simulations de trafic multi-agents. Bien que les deux approches développées dans cette thèse apportent des résultats prometteurs, certaines limites subsistent et ouvrent des pistes pour des recherches futures. Dans le cadre de l'approche basée sur Unite and Conquer, une optimisation plus poussée de la répartition de charge entre les nœuds et une réduction des surcharges de communication permettraient d'améliorer encore les performances. Par ailleurs, l'intégration de modèles prédictifs basés sur l'IA pour anticiper les comportements des agents pourrait alléger davantage la charge de calcul. Pour l'approche basée sur les MLP, l'enjeu principal reste la généralisation à de nouveaux scénarios. Des techniques telles que le Transfer Learning ou l'adaptation de domaine devraient être explorées pour améliorer la capacité du modèle à s'adapter à des environnements différents de ceux utilisés pour l'entraînement. Enfin, l'exploration de modèles neuronaux plus avancés, tels que les Transformers, pourrait offrir une meilleure capture des interactions entre agents tout en maintenant l'efficacité des calculs. En perspectives, une piste de recherche particulièrement prometteuse consisterait à combiner l'approche Unite and Conquer avec celle basée sur les réseaux de neurones, pour tirer parti des avantages de chacune et surmonter certaines de leurs limites. L'idée serait d'explorer le champ des poids possibles pour chaque neurone dans le modèle prédictif, en utilisant la puissance de calcul distribué inhérente à la méthode Unite and Conquer. En d'autres termes, il s'agirait de répartir le processus d'apprentissage des réseaux de

neurones sur plusieurs nœuds de calcul, en parallèle, afin d'accélérer considérablement la convergence du modèle, ceci en explorant efficacement les différents paramètres. Chaque nœud pourrait explorer une combinaison spécifique de poids et de biais pour les neurones, tout en communiquant les résultats à intervalles réguliers pour ajuster la direction de l'apprentissage global du modèle. Cela permettrait de tester simultanément plusieurs configurations de neurones, réduisant ainsi le temps nécessaire pour trouver les paramètres optimaux. En combinant ces deux méthodes, on pourrait non seulement optimiser le temps d'apprentissage, mais aussi augmenter la précision du modèle, car l'approche distribuée faciliterait l'exploration d'un plus large espace de solutions. Cette exploration distribuée pourrait également s'adapter dynamiquement en fonction des performances observées dans chaque nœud, ce qui favoriserait un ajustement intelligent des ressources de calcul vers les configurations les plus prometteuses. De plus, l'intégration de cette approche hybride permettrait également de résoudre partiellement les problématiques liées à la généralisation des modèles MLP. En explorant un espace de solutions plus large et en adaptant les configurations de neurones en temps réel, on pourrait concevoir un modèle plus robuste, capable de mieux s'adapter à des environnements ou des scénarios très différents de ceux utilisés lors de l'entraînement initial. Cela offrirait une capacité de généralisation bien supérieure, en combinant la flexibilité des réseaux de neurones avec l'efficacité de la distribution de charge offerte par Unite and Conquer.

En conclusion, cette thèse a démontré que l'utilisation de concepts liés au calcul haute performance et à l'intelligence artificielle apporte des résultats prometteurs pour améliorer les performances des simulateurs de trafic multi-agents. L'approche UC offre une solution robuste pour des simulations à grande échelle nécessitant une exploration détaillée, tandis que l'approche MLP propose une alternative rapide et efficace pour des applications nécessitant des réponses quasi instantanées. Ces deux approches représentent un pas important vers des outils de simulation plus performants, adaptés aux besoins croissants des villes intelligentes et des systèmes de transport modernes.