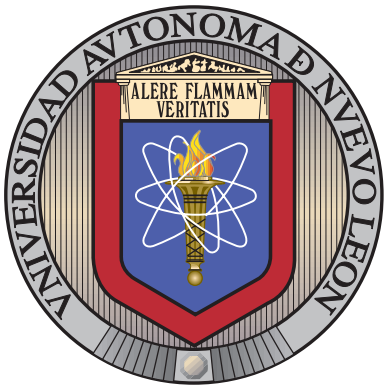
Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Subdirección de Estudios de Posgrado



Solving the Bus Bunching Problem with a Multiagent System

por

Jesús Ángel Patlán Castillo

en opción al grado de

Maestría en Ciencias

de la Ingeniería con Orientación en Sistemas

San Nicolás de los Garza, Nuevo León octubre 2020

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Subdirección de Estudios de Posgrado

Solving the Bus Bunching Problem with a Multiagent System

por

Jesús Ángel Patlán Castillo

en opción al grado de

Maestría en Ciencias

de la Ingeniería con Orientación en Sistemas

San Nicolás de los Garza, Nuevo León octubre 2020

Universidad Autónoma de Nuevo León
Facultad de Ingeniería Mecánica y Eléctrica
Subdirección de Estudios de Posgrado

Los miembros del Comité de Tesis recomendamos que la Tesis "Solving the Bus Bunching Problem with a Multiagent System", realizada por el alumno Jesús Ángel Patlán Castillo, con número de matrícula 1595261, sea aceptada para su defensa como requisito para obtener el grado de Maestría en Ciencias de la Ingeniería con Orientación en Sistemas.

El Comité de Tesis

Dr. Romeo Sánchez Nigenda
Director

Dr. Francisco Torres Guerrero
Revisor

Dra. Yasmín Á. Ríos Solís
Revisor

Vo. Bo.

Dr. Simón Martínez Martínez
Subdirector de Estudios de Posgrado

SUBDIRECCIÓN DE ESTUDIOS DE POSGRADO
FIME
059

San Nicolás de los Garza, Nuevo León, octubre de 2020

Ciudad Universitaria Pedro de Alba s/n, C.P. 66455. A.P. 076 Suc. "F"
San Nicolás de los Garza, Nuevo León, México. Tels: (81) 8332 0903 /
Conm.: 8329 4020 / Fax: (81) 8332 0904

A mis padres, Miguel Angel Patlán Rodríguez y Yolanda Margarita Castillo Prieto, quienes están siempre para apoyarme.

Contents

List of Figures

List of Tables

Agradecimientos

Agradezco principalmente a mis padres por haberme enseñado los hábitos del estudio.

Agradezco al Dr. Romeo Sánchez Nigenda por su tiempo y dedicación para guiarme por el buen camino para concluir con esta tesis y con mis estudios en el posgrado.

Al Dr. Francisco Torres Guerrero y a la Dra. Yasmín Águeda Ríos Solís por su tiempo y rigor para la revisión de la tesis.

Agradezco al cuerpo docente de PISIS por su disponibilidad para resolver todo tipo de dudas tanto dentro como fuera del aula.

Agradezco a mi compañera Citlali Maryuri Olvera Toscano por compartirme datos del medio de transporte Ecovia del estado de Nuevo León, junto con nuestros compañeros que en su momento la apoyaron para la recolección de datos, y también por su aporte de su modelo lineal que fue utilizado durante las distintas fases de esta investigación.

Agradezco a mis amigos y compañeros, tanto del posgrado de PISIS como de licenciatura en la Facultad de Ciencias Físico Matemáticas y de la Facultad de Ingeniería Mecánica y Eléctrica, con quienes pase grandes momentos e hicieron más ameno el duro camino que me llevo por terminar con esta investigación.

Agradezco al Consejo Nacional de Ciencia y Tecnología (CONACyT) por la beca que me permitió continuar con mis estudios, deseando que así como a mí me

fue otorgado este apoyo, también tengan la oportunidad muchos estudiantes más para superarse y aportar dentro del ámbito científico que existe en México.

Espero que la calidad de este trabajo refleje el esfuerzo del estudio que he llevado hasta el momento de mi vida.

Resumen

Jesús Ángel Patlán Castillo.

Candidato para el grado de Maestría en Ciencias
de la Ingeniería con Orientación en Sistemas.

Universidad Autónoma de Nuevo León.
Facultad de Ingeniería Mecánica y Eléctrica.

Título del estudio:

Solving the Bus Bunching Problem with a Multiagent System

Número de páginas: 86.

Objetivos y método de estudio: El objetivo de esta investigación consiste en el desarrollo de un sistema multiagente que simule las características de un sistema de transporte de camiones, integrando modelos matemáticos para la generación de planes de acción para los camiones con el fin de reducir el fenómeno del amontonamiento de camiones (*Bus Bunching*) y proporcionar un mejor servicio a los usuarios de la ruta.

Contribuciones y conlusiones: La principal contribución es la simulación en un sistema multiagente, el cual está preparado para ser adaptado a distintos casos de sistemas de transportes que cumpla con características similares a la ruta

de camiones. Este sistema es de código libre, por lo que es modificable y adaptable para cualquiera que tenga interés en aplicar sus modelos o implementar nuevas características al sistema, particularmente es posible añadir nuevos modelos matemáticos para la generación de nuevos planes de acción de los camiones.

Firma del asesor: 

Dr. Romeo Sánchez Nigenda

Summary

Jesús Ángel Patlán Castillo.

Candidate to the degree of Master of Science on Systems Engineering

Universidad Autónoma de Nuevo León.
Facultad de Ingeniería Mecánica y Eléctrica.

Thesis:

Solving the Bus Bunching Problem with a Multiagent System

Number of pages: 86.

Objectives and study method: The objective of this research is to develop a multiagent system that can simulate the characteristics of a bus system, with the implementation of mathematical models to generate plans to reduce the bus bunching phenomenon and provide a better service to route users.

Contribution and conclusions: The main contribution the thesis is the intorudction of a multiagent system model and a siomulation platform to address the bus bunching problem in public transportation networks. Both, the model and simulation environment, are flexible to consider further adaptations to new problems

and transportation environment. Particularly, it is possible to add new mathematical models to generate new action plans for the agents that simulate the bus units in the transportation network. The simulation platform code is free and publicly released under an Apache 2.0 license.

Adviser signature: 

Dr. Romeo Sánchez Nigenda

Chapter 1

Introduction

In recent years, the use of public transportation has increased, as the population in cities and displacement from job centers to suburban areas has risen. The use of public transport in the US increased by 21 % since 1997, which is more than 19% of the population growth rate [1]. More than 6800 enterprises provide public transportation services in the US [1], which have invested in research for methods to increase the effectiveness of their resources to provide better performance and quality of service for their users. Metrics of performance rely on maximizing the number of passengers that can travel in a single transport unit and minimizing the traveling time that passengers take. With the increase in gasoline prices and other resources to maintain and operate transportation lines, it is critical to improve their performance to reduce systems' costs, which affect the economy of the enterprises that manage transportation systems and the end-user that pay for their services.

Cities work as a one, wellness of the city's population depends on the performance of the enterprises and people that live on it. Hospitals need their doctors and resources to build reliable healthcare systems. Universities must have their teachers well prepared to deal with the stress of classes and students. Office workers have to maintain a good quality of service to increase the utility of the enterprises for whom they work. We all depend on someone, and everybody must perform well so we can all improve our quality of life.

Public transport is essential because it has a direct impact on the economy of the city. In the US, 87% of trips are made by students and workers to reach

their schools and enterprises [46]. It is a stunning quantity of people that rely on public transportation to get to their homes and workplaces every day, proving that not only the economy of the city depends on it, but also the lives of everyone living on it. Without a reliable public transport system, doctors, teachers, office workers, and students may not reach their respective destinations on time. The average time an American waits at a bus stop is 40 minutes. In a year, this waiting time becomes 150 hours, more than six days waiting for the bus. This time could be well used for passengers to increase their productivity at work or their times with families, reducing their stress levels. These numbers significantly depend on the public transportation system in the city [33].

In the US, there are estimates that every dollar spent in public transportation returns a gain of four dollars and more than 50,000 new jobs per billion dollars in investments [1]. Furthermore, it is ten times safer per mile to use public transportation rather than private automobiles. Public transportation is not only is safer but also is cheaper. A family can save almost 10,000 dollars a year per car with the use of public transport [1].

Even for people that uses their own automobile, it is much safer to use public transport. It is estimated that public transportation is 10 times safer per mile than traveling by automobile. Not only it is safer, but also cheaper. A family can save almost $10,000 dollars if they decide to use public transportation and not buy a car [1].

The use of public transportation also helps with current environmental problems. In the US, there is an estimated consumption of 4.2 billion of gasoline gallons annually. Any reduction in gasoline consumption can also help to reduce carbon emissions by 37 million metric tons per year [1].

Furthermore, traffic is a severe problem that affects everyone using private or public transportation. Car drivers tend to spent 40 hours stuck in traffic every year. This quantity could be even higher without public transport.

There are multiple public means of transport like the tram, suburban rail, metro, and bus. In this work, we focus on bus lines and routes. This type of public transport concentrates the 63% of the journeys in cities. Given that it is the most popular public transportation service, further research could help to maximize its efficiency [1].

1.1 Problem description

Bus lines are one of the most used public means of transport in cities. One of the most severe problems that occur in Bus lines is known as the Bus Bunching Problem (BBP). BBP happens when several buses of the same route arrive at the same stop around the same time, or when they are traveling side-by-side. BBP originates that buses agglomerate in some parts of the journey, increasing the waiting times of passengers at bus stops. Distributing buses efficiently along the route line is a complex problem, mainly due to the dynamism of the environment and the partial observability of the transportation network. For example, a bus driver might accelerate to maintain a reasonable distance from the rear bus; however, the rear bus might be as well decelerating, creating a longer delay between both buses. Besides, the front bus might get stuck in traffic, provoking that the but that increased its speed to reach it, originating the bus bunching phenomenon. This scenario is just an example of what could happen in one part of the route, without even considering other properties of the transportation network that affect its efficiency (e.g., passenger arrivals at bus stops). The complexity of this problem resides in the unknown factors that might arise during the daily work schedule of bus units and the way they interact to satisfy the network's demand.

There are methods to reduce bus bunching on a route. The most common is Speed Regulation; this is, increase or decrease the speed of buses during travel. Another strategy is Bus Holding, which introduces waiting times for buses at bus stops [50, 21, 53]. Another alternative is skip-stop, which allows for a bus to skip

given bus stops to maintain its speed and balance its distance from the rear bus [24, 7, 37]. However, this strategy implies that passengers keep waiting at the skipped stops. The last method is deadheading. Deadheading allows marking bus stops that have a low passenger rate of arrival. A marked stop with deadheading will only be serviced by buses with less frequency, allowing buses to arrive more frequently to more demanding stops in the route [15, 8, 12].

By using mathematical models and, more recently, multiagent systems (MAS), the bus bunching can be reduced, impacting directly to the user with a lower waiting time (a better service), and indirectly to the traffic by keeping buses spread through the route.

1.2 Justification

As mentioned earlier, there are different methods to treat the bus bunching problem and reduce the headway between buses in a route [3]. Most of the approaches involve generating a solution from a linear programming problem in a specific time and use the solution to hold (or not) the buses at a stop [23]. Some approaches are using MAS, in which different kinds of agents interact with each other to share information about the bus and its route, and with this, each agent can determine how much time the bus should wait (if they should) at the next bus stop [56, 10]. Work by [9] shows that using MAS, by distributing the bus holding decision between buses, is more effective than using linear programming. Other approaches require buses not only to wait at bus stops but also to regulate their speed in real-time to balance the headways [54, 41, 34, 13]. In recent years, documents related to the bus bunching problem have increased, making this a hot topic for research, as shown in figure 1.1.

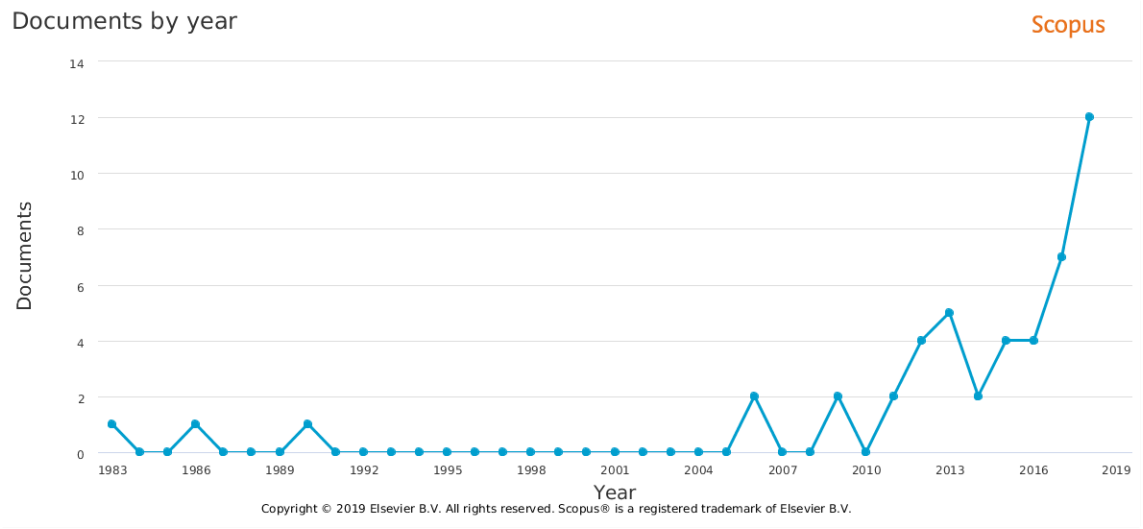


Figure 1.1: Numbers of works related to BBP through the years

A multiagent system is a system that involves the interaction of multiple intelligent agents to solve a problem [52, 43]. The agents in a multiagent system work together by communicating and performing actions based on their perspective of the environment, making a MAS suitable for the bus bunching problem. Unknown factors may still arise during journeys, but if an agent is controlling each bus of the route, each one can select actions depending on the situation that the agent is. Even if the agent cannot deal with the problem by himself, he can communicate with other agents to decide for the best plan of action based on the information that all agents have of the environment.

There are two main kinds of Multiagent Systems, those with a centralized architecture and a distributed one [45]. In a centralized architecture, there is a specific agent in charge of commanding the others based on their perspectives collected from them. Meanwhile, a distributed architecture allows for agents to take individual actions while communicating the results of them to other agents in the system to enrich their views of the environment for future decisions.

For the Bus Bunching problem, we might have a centralized architecture in which a central command receives information updates from the agents (bus units)

in the route, generates a global model of the current situation, and orders to each transportation unit the action to perform. A distributed architecture, on the other hand, allows each bus unit to perform actions based on its view of the environment, informing the front and rear closest buses about such decisions, to let them construct their future action plans [22, 28, 47].

Given the external factors of the environment that affect the bus routes, a distributed multiagent system becomes the perfect platform to deal with the bus bunching problem. External factors may impact differently through the transportation network. Therefore, the decision-making process of the agents must consider the closest state of the environment they perceive. Table 1.1 summarizes the most recent works for solving the bus bunching problem and contrasts them with the proposed research of this thesis.

Paper	Year	Use MAS	Use Bus Holding Strategy	Use Speed-Regulation Strategy	Use Skip-Stop	Use deadheading
Solving the Bus Bunching Problem with Multiagent System	2020	Yes	Yes	Yes	Yes	Yes
Real-time public-transport operational tactics using synchronized transfers to eliminate vehicle bunching	2016	No	Yes	Yes	Yes	No
Real Time Bus Holding Control on a Transit Corridor Based on Multi-Agent Reinforcement Learning	2016	Yes	Yes	Yes	Yes	No
Linear bus holding model for real-time traffic network control	2015	No	Yes	No	Yes	No
A Multi-Agent Reinforcement Learning approach for bus holding control strategies.	2015	Yes	Yes	No	Yes	No
Dynamic bus holding strategies for schedule reliability: Optimal linear control and performance analysis	2011	No	Yes	Yes	Yes	No
A Self-coordinating bus route to resist bus bunching	2011	No	Yes	No	Yes	No
An approach to reducing bus bunching	2009	No	Yes	Yes	Yes	No
Reducing bunching with bus-to-bus cooperation	2009	No	No	Yes	No	No
Distributed architecture for real-time coordination of bus holding in transit networks	2003	Yes	Yes	No	Yes	No

Table 1.1: Summary of Recents Works in BBP

1.3 Objectives

This investigation have the following interrogatives to be answered:

1.3.1 How can multiple strategies improve the solution of the bus bunching problem?

We expect that giving more strategies to the agents will have a positive impact on reducing the bus bunching problem, since the agents can perform these different strategies depending on the situation that are in the route. We expect that the agents won't have problems with the response time since the agents will only act accordingly to the local state of the environment.

1.3.2 How does the communication between agents in the multiagent system improve the solution of the bus bunching problem?

We expect that communication between agents will improve the effectiveness of the route, since the agents will be able to communicate the external factors that arise on the route to nearby agents, and then the agents will have a more accurately state of the environment to take more precises decisions.

1.3.3 How does a centralized multiagent system perform against the bus bunching problem, compared to a linear programming model?

We expect that a centralized multiagent system will have a better performance compared to a linear programming model. The use of a centralized multiagent system will help in formulating the mathematical model of the current state of

the environment, and the solution given by this model will be applied only if the centralized agent considers that the solution obtained is acceptable for the time that was formulated.

1.3.4 How does a distributed multiagent system performs against the bus bunching problem, compared to a linear programming model?

We expect that a distributed multiagent will have better performance compared to a linear programming model. Similar to the centralized architecture, the distributed model will also use an agent that will decide if the solution given by the model is acceptable for the current time.

1.3.5 How can a multiagent system improve the solution of the bus bunching problem compared to linear programming?

We expect that the implementation of multiagent system gives a better performance than using a linear programming model, since the multiagent system can take into account the different external factors that may arise during the route. It is possible to model external factors through a linear programming model, however, it is too complex to model every external factor that can affect the route, and when we aggregate more variables to the problem we tend to have a model that can take more time solve. Increasing the time that the solver takes to solve the problem might impact on the performance if the solution given by the model becomes infeasible on the new current state.

1.3.6 Does the distributed multiagent system performs better than the centralized multiagent system in dealing with the bus bunching problem?

We expect that the distributed multiagent system will perform better than the centralized multiagent system. This is because the distributed architecture will take into account the decisions taken by the centralized agent, and will give the bus agents the will to decide if the commanded action is the adequate based on their local updated perspective of the environment, which will be a better view of the state than the one the centralized agent has.

1.3.7 Hypothesis

In general, we expect that the implementation of the multiagent system with multiple strategies to deal with the bus bunching problem and the communication architecture of the multiagent system will minimize the passengers waiting times compared with linear programming using a single strategy. The multiagent system with multiple strategies is expected to make the agents to work together to reduce the bus bunching phenomenon, increasing the effectiveness of the route during the dailybus schedules. For this, we will perform multiple tests based on various configurations of what a bus route might have: the numbers of buses on the route, the number of stops at the route, the distances between bus stops, and the boarding and dwelling rate of passengers at each stop.

1.3.8 Thesis Structure

Chapter 2 explains the basic definitions used in the thesis. Chapter 3 introduces the literature review of the bus bunching problem. Then, Chapter 4 presents the methodology used to design the proposed architecture for the Multiagent system. Chapters 5, 6, and 7 introduce the different components of our solution: the sim-

ulation platform, the centralized architecture, and the distributed MAS. Chapter 8 presents and analyses the results of the proposed work. Finally, Chapter 9 presents conclusions and future research directions. Additionally, Appendix A presents the system BusiMA, the multiagent system developed during this research, with installation instructions.

Chapter 2

Theoretical Framework

This chapter introduces all the definitions and terms used throughout the rest of the thesis. For any in-depth explanations about the background work, we suggest taking a look at the bibliography provided.

2.1 Conceptual Framework

A simulation is the imitation of the operation of a real-world process or system over time [2]. With simulations, we can tests how mathematical models and algorithms may respond to a real-world scenario, to evaluate if they may have a positive impact before implementing them [42].

A probability distribution for a discrete random variable is defined as a mathematical formula that gives the probability of each value of the variable [17]. Particularly, a Poisson distribution is a probability distribution, given by:

$$p(r;\mu) = \frac{\mu^r e^{-\mu}}{r!}$$

Where the variable r is an non-negative integer and the parameter μ is a real positive quantity. It describes the probability to find exactly r events in a given length of time if the events occur independently at a constant rate μ [49]. It is one of the most important probabilistic distributions since it has multiple applications [29]. In this work, we use a Poisson distribution to simulate the rate of passengers that arrive at

each stop in a bus route, since Poisson distribution has been used to simulate these arriving rates in previous works [38, 23].

Linear programming is concerned with the optimization of a linear function while satisfying a set of linear equality and/or inequality constraints or restrictions [4]. Linear programming has been used to deal with the bus bunching phenomenon [23, 38].

One way to deal with bus bunching problem is using multiagent systems [57, 35, 27]. These systems contain multiples agents, each one being a reactive system that exhibits some degree of autonomy, which means that the agents decide the actions that will perform to solve a set of tasks in an environment [6]. The characteristics of agents are:

- Autonomy: An agent is autonomous since they decide which actions take to solve given tasks. One way to represent the "will" of an agent is by the belief-desire-intention (BDI) model programmed to the agent, which we will talk later.

- Proactiveness: An agent is proactive by being able to exhibit an intention to reach given goals, which means that the agent will take the necessary actions to execute a set of tasks to reach the goal. This is also decided by the programmed BDI model of the agent.

- Reactivity: An agent is reactive if they can take actions when the environment evolves. When an external factor of the environment modifies in any way the action plan of the agent, the agent must be ready to respond by modifying correctly the plan or by developing a completely new plan to solve the task.

- Social ability: An agent has social ability when he exchanges information with other agents. Having a social ability is the core property of a multiagent system, because by exchanging information with multiple agents, they can cooperate to develop a richer plan by having a better idea of how the whole

environment currently is.

The belief-desire-intention model is the "brain" of the agent. An agent takes decisions based on the following three concepts [6]:

- Beliefs: The beliefs of the agent is the information that it has about the world. This information may or may not be correct or accurate. Beliefs on the multi-agent system allows the agents to take decisions based on what they think the current environment is. This property allows the simulation to have a more precise behaviour, as required to model public transportation networks.

- Desires: The desires of the agent are all the affairs that the agent might want to accomplish, meaning that the agent will probably complete these affairs, but in some cases the agent probably will not. In addition, one agent can have desires that are mutually incompatible with one another. For example, in our work, desires correspond to the distances between bus units encouraged to maintain to reduce Bus Bunching. Such desires might trigger actions (intentions) in the agents to overcome the problem.

- Intentions: The intentions of the agent are the actions selected for execution among all the possible options. Based on the beliefs that the agent has on the environment and the desires that it has, each agent will have an intention to perform certain action to accomplished their desires. Nevertheless, even if an agent has the intention to perform an action it may fail to execute it. For example, an agent might want to increase the speed of the bus, but the traffic on the avenue may not allow it to increase any further. In these scenarios, agents might take contingency actions to mitigate problems in the environment.

Chapter 3

Related Work

3.1 Background

The first strategy used to deal with bus bunching is control models. Newell and Potts (1964) recognized the instability of bus systems, and years later Newell and Osuna worked together to add slack time to maintain the buses on a schedule [36, 39]. Since then, multiple works have used different control models to reduce the bus bunching phenomenon. Li, Liu, Yang and Gao (2019) recent work used a robust dynamic control model with the strategies of bus holding and speed control to deal with the bus bunching phenomenon [30].

Multiple studies have concluded that people value more the time spent waiting for a bus at a stop than the time spent on the bus [31, 5, 14]. This observation highlights the users' perspective on how buses must operate by giving more value to taking stops rather than skipping them. However, several works have used skip-stop strategies to reduce bus bunching. Cao, Zhichao, and Ceder (2019) recently proposed a skip-stop strategy based on service timetables [7]. Another strategy similar to skip-stop is called deadheading. Yu, Yang, and Li (2012) proposed deadheading divided into two different phases: reliability assessment of further transit services and optimization of pathway deadheading operation [55].

Hernandez-Landa et al. (2015) proposed a linear model to establish the bus holding times that buses must perform at stops to reduce bus bunching [23]. Olvera

(2018) introduces another linear model that decreases bus bunching in time windows [38]. Gkiotsalitis and Cats (2019) also introduced a mathematical model that implements discrete nonlinear constrained optimization to control bus holdings in time windows [19].

In recent years, there are proposals of multiagent based solutions. Wang and Sun (2020) model agents in every bus unit in their network with communication capabilities between them, and develop a multiagent deep reinforcement learning framework as a control strategy for the buses [50]. Weiya Chen, Zhou, and Chunxiao Chen (2016) presented a coordinated holding control framework based on multiagent reinforcement learning to reduce bus bunching [10]. Zhou, Wang, and Cui (2017) use a distributed scheduling strategy, using an agent on every bus, to recollect its necessary information [57].

3.2 Patents

The flexible fare bus framework is a patent used to reduce bus bunching by dynamically adjusting the headway-threshold between buses depending on the passengers' demand. The proposed FlexiFare algorithm specifies the headway that each bus must have on the route. In figure 3.1, we can see a diagram of its functionality preprinted from its source [51].

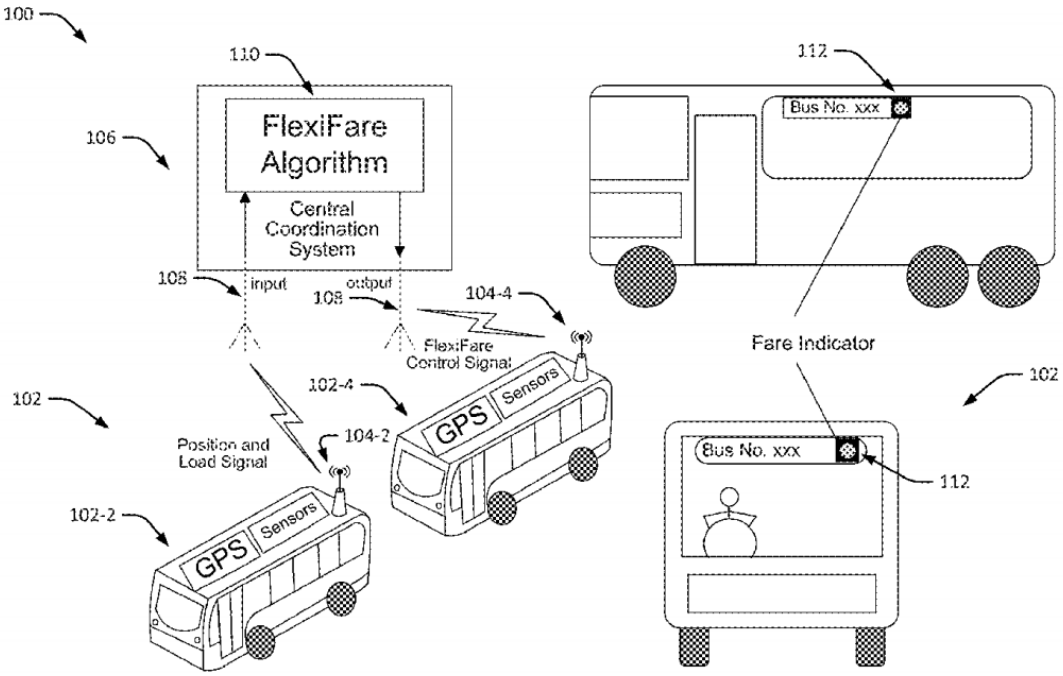


Figure 3.1: Flexible fare bus framework

The automated system for preventing bus bunching is another patent that reduces the bunching between vehicles by sharing information between them and by specifying the holding times and speeds they must have during travel. On the driver interface, the bus driver can check how much time must hold at a stop. Figure 3.2 represents the driver interface preprinted from its source [44].

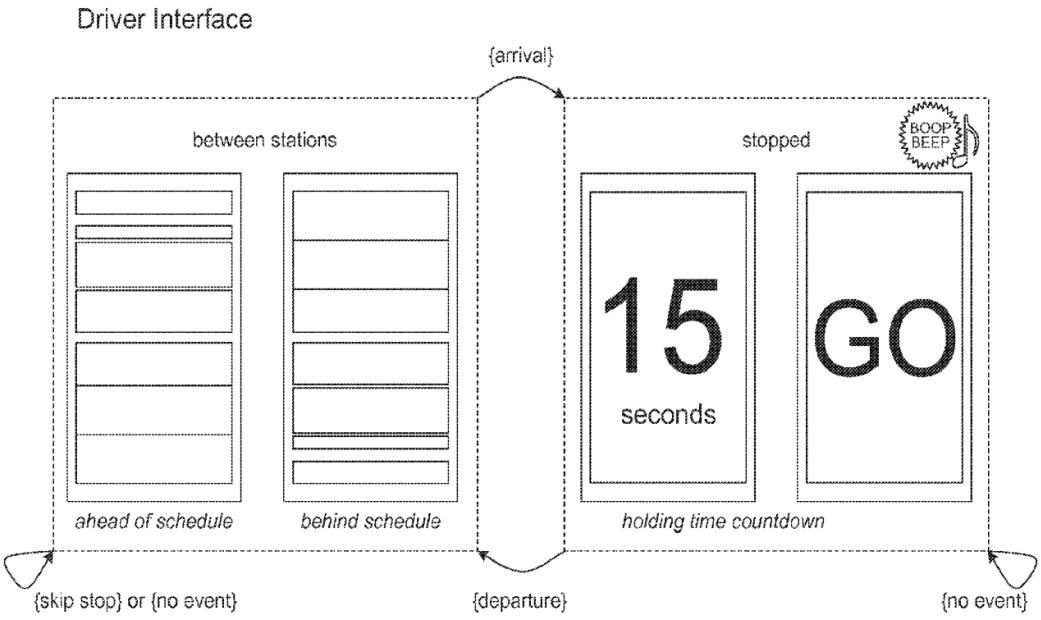


Figure 3.2: Automated system for preventing vehicle bunching

The real-time vehicle spacing control detects when vehicle bunching occurs and resolves it by regulating the arrival and departure times of vehicles at stops [32]. Figure 3.3 shows a summarized functionality of the patent retrieved from its source [44].

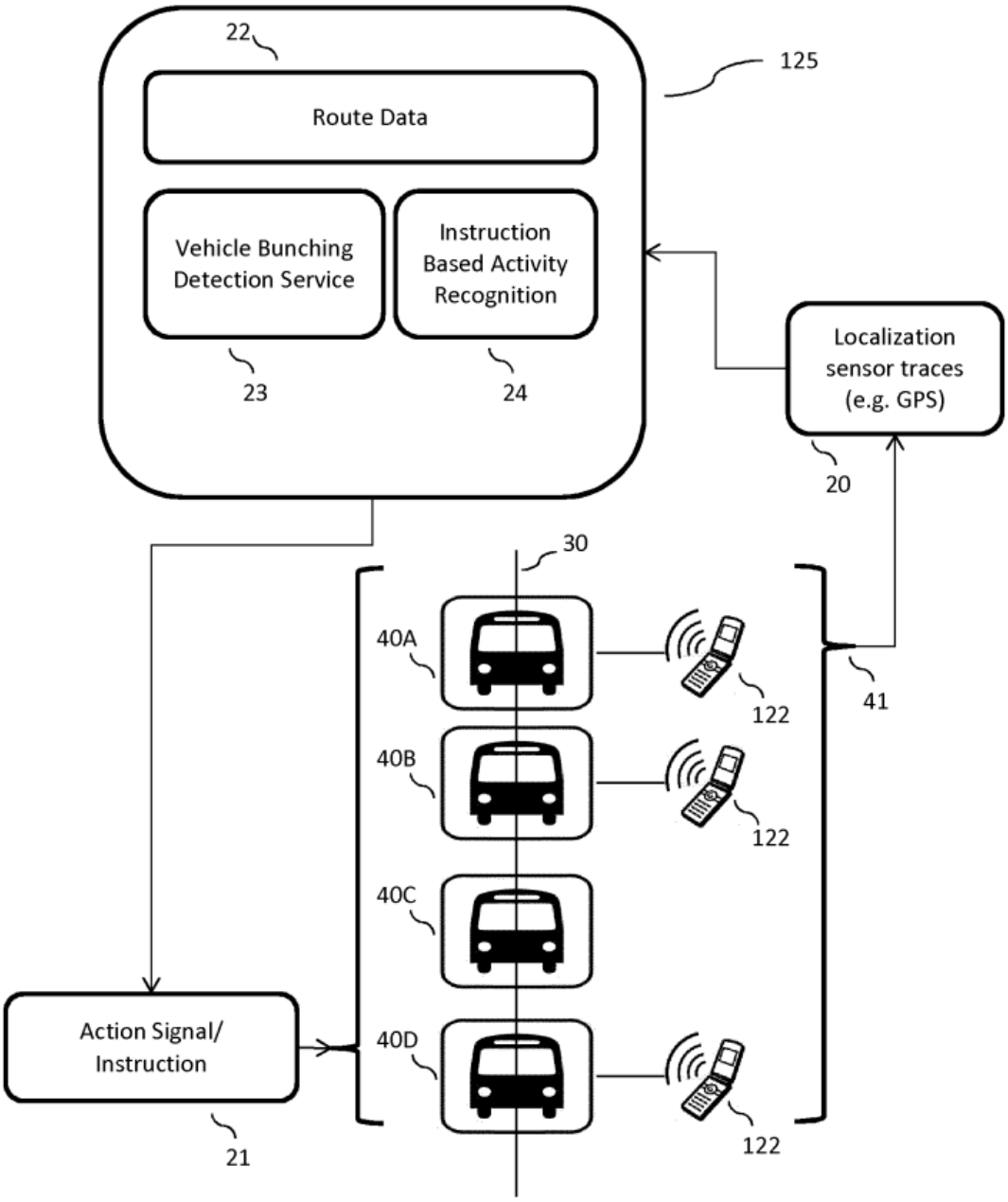


Figure 3.3: Real-time vehicle spacing control

Chapter 4

Methodology

In the previous chapter, we introduced some related works for solving the Bus Bunching problem using Multiagent systems (MAS). One distinctive difference between those works and the one proposed in this thesis is the number of strategies supported to address the problem. The most popular technique supported in the literature is Bus Holding. Even a MAS using a unique method shows positive results in solving the problem. With this information in mind, we consider alternative algorithms that could enrich the MAS paradigm. In particular, Table 4.1 shows recent strategies identified in the literature. The original implementation of the methods falls under different methodologies like linear programming, stochastic processes, Markov chains, and mathematical modeling [30, 31, 5, 50]. We analyzed these methods in the context of Multiagent systems. Table 4.1 shows a brief justification of why we believe a MAS can use and improve over the problem-solving strategy from the literature.

Strategy	Description	How can the MAS improve it
Bus holding	Holds a bus in a stop for some time to adjust the route	The B-Agent could communicate with nearby B-Agents to determine if it should hold for a particular time in a stop
Speed regulation	Regulates the speed of the bus by increasing it or decreasing it	The B-Agent could communicate with nearby B-Agents to determine if it should increase or decrease its speed to adjust their headway
Skip-stop	Commands a bus to skip a particular stop	The CP-Agent could command a particular B-Agent to skip a stop
Deadheading	A stop is marked in such a way that 1 out of every n (depending on demand) buses make a stop	The CP-Agent could command the buses of the route to program which stop should be implemented the deadheading strategy

Table 4.1: Strategies to deal with BBP.

4.1 Programming Architecture

We used the Java Programming language [11] and JASON libraries [26] to build a MAS architecture and platform simulation for the proposed solution. Furthermore, we also consider libraries from Gurobi to solve the mathematical models used by the MAS. The design of the MAS system is scalable to update it to further strategies or new environmental factors in the future. The system is also flexible since it supports several configurations to evaluate it under different scenarios. The different configurations of the MAS are:

- Initial Configuration: The initial configuration consists of a given number of agents of each type, the number of bus stops to be considered in the route, the distances between each pair of stops, and the strategies the agent can use to execute a new simulation.

- Agents Configuration: The agent configuration builds upon a generic bus class, which different types of agents inherit. Therefore, it is possible to create multiple kinds of bus agents with different strategies, BDI architecture, capacity, and speed limits, among other factors. In the current version of the multiagent system, every bus agent shares the same characteristics.

- Environmental Configuration: The environment configuration allows us to represent those factors from the environment that interact with the MAS. For example, there could be some links between stops in which traffic speeds vary or become unavailable due to random events in the environment. Such environmental settings have the purpose of modeling more realistic environments in the simulation platform and verifying the effectiveness of the implemented strategies under different circumstances. The evaluated version of the multiagent system, for the thesis work, does not consider dynamic environmental factors.

- Configuration of Strategies: In this part, we can add new strategies for bus agents to interact with the environment. This configuration defines the types of agents that can perform a given method, the information that agents need to use the strategy, and the requirements needed to apply the strategy procedure. Notice that this configuration must be compatible with the environmental setting to work since the agent must be capable of obtaining information from the environment.

- Route Configuration: The route configuration involves the possible speed limits between a pair of bus stops and the number of edges available between them.

- Daily configuration: This configuration involves global settings that could change depending on the planning horizon. The global settings are the current speed limits between stops, the rate of passengers arriving at stops, the maximum number of agents that can be on the route. There could be multiple configurations for any amount of days that the user might want to simulate. For example, there could be a weekday configuration, weekend configuration, or a holiday configuration to enact days in which holiday events could occur, which can affect parts of a route. In the current version of the multiagent system, we run single day configurations.

4.2 Multiagent Model

4.2.1 Objective Function

Mathematical models given to solve the BBP focus on minimizing the passengers waiting time [23, 56] or minimizing the deviation headways [9, 54, 34, 10, 3]. This particular model focus on minimizing the passenger waiting time.

4.2.2 Model Diagram

In the figure 4.1 and 4.2, we can observe how the actions of each agent interact with each other along with the environment.

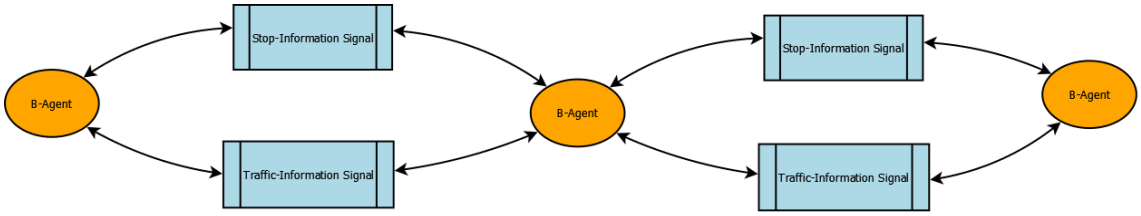


Figure 4.1: Buses communication model.

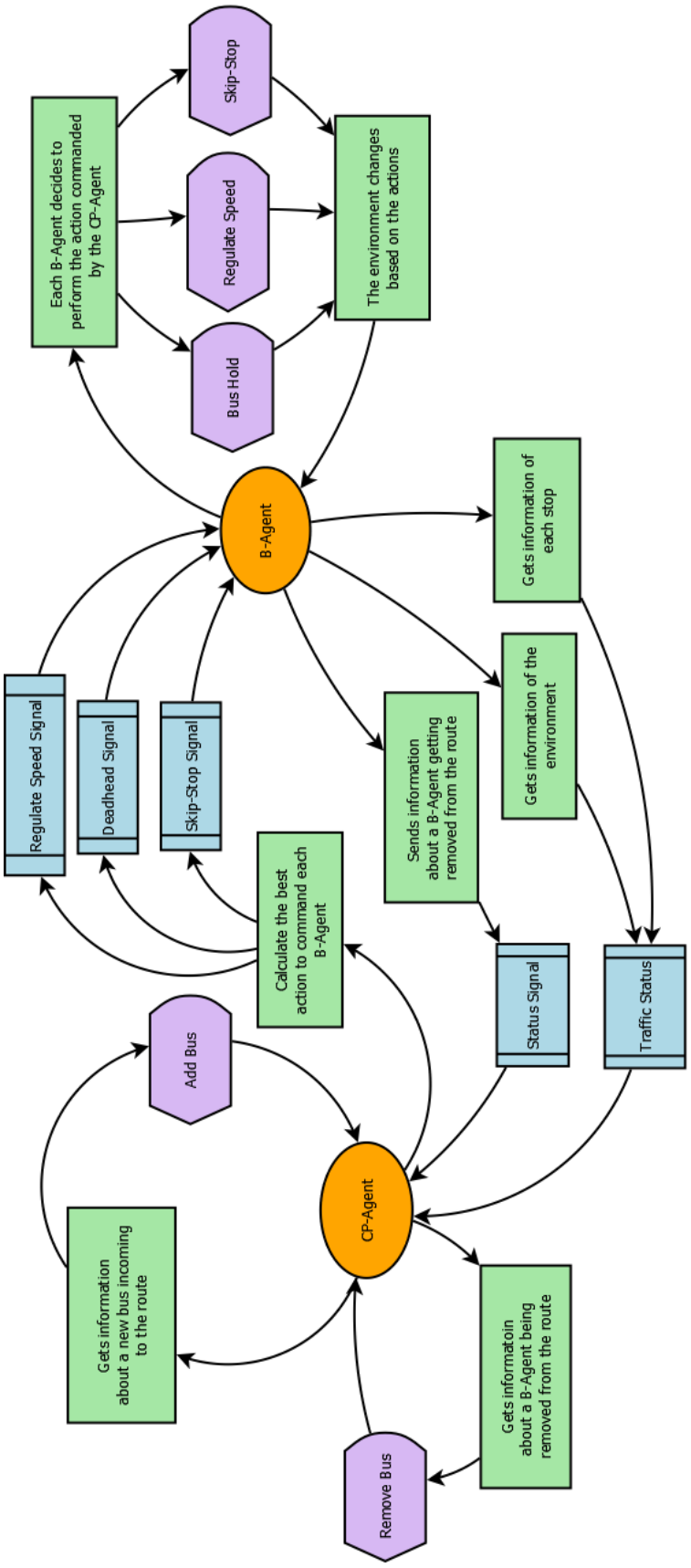


Figure 4.2: Full model diagram.

4.2.3 Multiagent Architecture

We propose two different architectures to deal with the bus bunching problem, a centralized and a distributed architecture. Most of the operations, in both of the architectures, will be trigger on a control point agent (CP-Agent) to coordinate the B-agents on the field when external factors affect them. The B-agents will control their bus actions and their communications with the CP-Agent and other adjacent B-Agents in the route (i.e., the next and previous buses).

4.2.4 Properties of the Multiagent Environmental Model

Partially Observable

The agents can only get information about their environment by using their sensors, and such information might not be correct. In the case of a bus route, it is impossible to know in advance every variable from the environment affecting the transportation network. Sensors on the bus can help to gather information about the number of passengers currently on the bus, the current traffic near the bus, and the number of people left at the stops where the bus went through. However, it is impossible to know in advance the people waiting at the next stop unless other buses communicate that information.

Dynamic

The environment is dynamic since multiple factors or problems, outside the control of the agent, can alter it. Some of these problems are traffic on the route, flow of passengers, and traffic accidents. Many of these factors cannot explicitly be enumerated, much less coded. Therefore, in our approach, we design agent plans solely based on information agents obtain from the routes.

Continuous

The environment is continuous since the environment is always changing, no matter if the agents take an action or not. Even if a bus stops working, the other buses must keep acting accordingly to the changes of the environment and must take into account the bus that stopped working.

Stochastic

Although the environment is dynamic, it is not stochastic. Every action in the model has deterministic outcomes, and at this version of the framework, we are not modeling yet failure.

Sequential

The environment is sequential since an action taken by a B-agent will affect its position on the route and thus its future actions. Notice also that actions of agents (i.e., buses) have implications on the environment and in other agents.

4.2.5 Types of Agents

There will be two kinds of agents:

- Control Point Agent : The control point agent (CP-agent) is a singleton agent from the model, who is in charge of coordinating the bus agents in case of exogenous events. A CP-agent receives information from bus agents to sketch action plans for them. For example, if the CP-Agent knows that a particular bus stop has a low flow of passengers, it could send a signal to bus agents in the route to skip it (i.e., deadheading strategy).
- Bus Agent : The bus agent (B-agent) controls the bus actions; that is, increasing or decreasing the speed of the bus unit, stopping at or skipping bus stops.

Besides, they also communicate valuable information for coordination from the route to the CP-Agent, like passenger flow and traffic situation. They can also exchange messages with other nearby B-Agents to inform about their location and current speed.

4.2.6 Agent's Actions

- Control Point Agent : The CP-agent actions can be seen by figure 4.3

  [Remove bus]: The CP-agent removes an agent from the route.

  [Add bus]: The CP-agent adds a bus to the route,

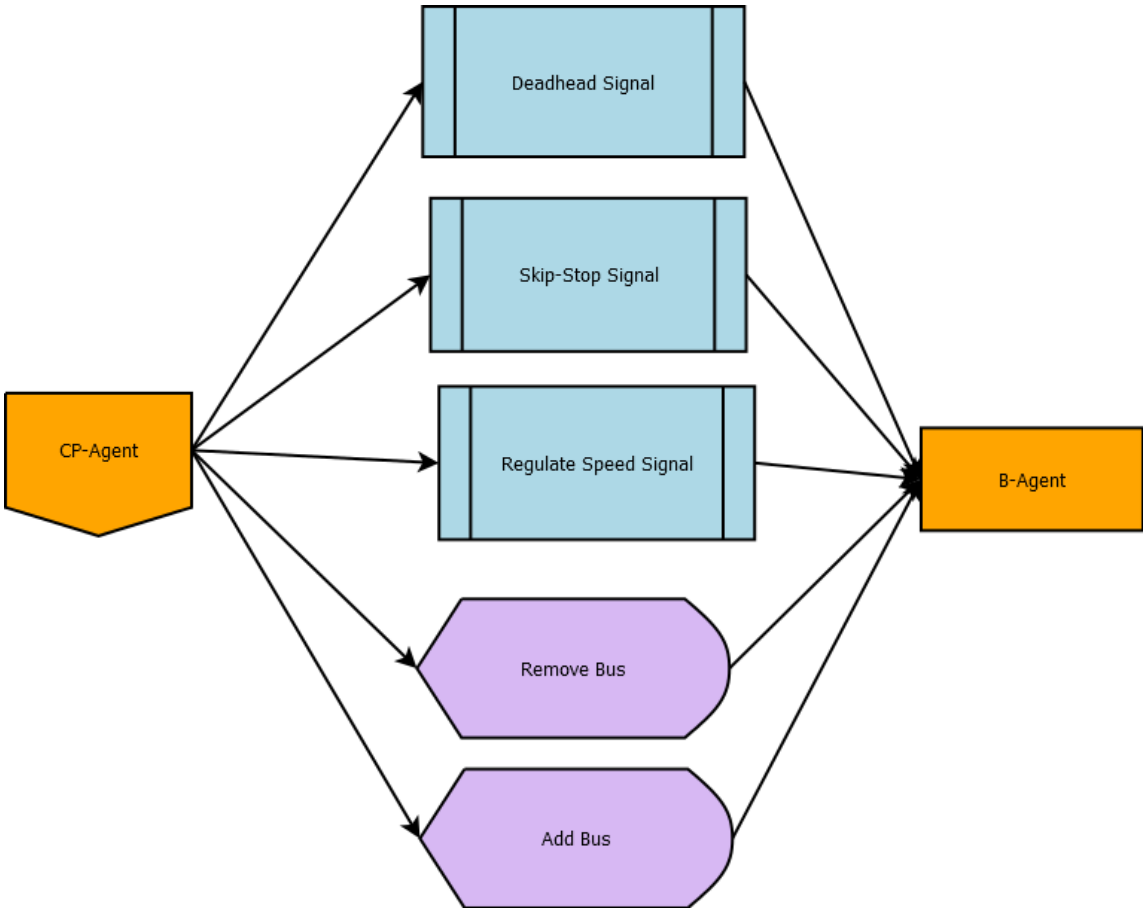


Figure 4.3: CP-agent actions.

- Bus Agent : The B-agent actions can be seen by figure 4.4

[Bus hold]: This action holds a bus at a stop for a period of time.

[Regulate Speed]: Increases or decreases the speed of a bus

[Skip-stop]: Skips the next stop in the route

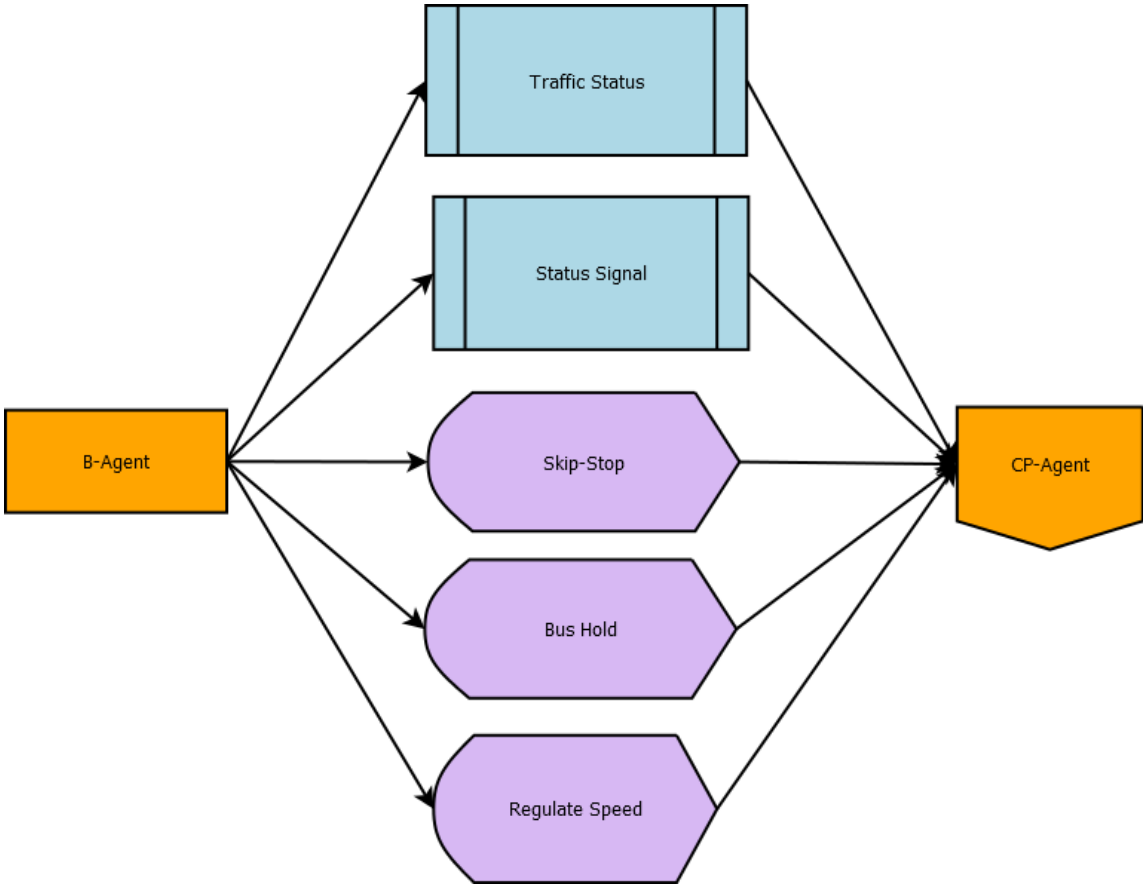


Figure 4.4: B-agent actions.

4.2.7 Agent's Communication

- Control Point Agent :

  [Deadhead Signal]: The CP-agent communicates to the B-agents about a specific deadhead stop.

  [Skip-stop Signal]: The CP-agent communicates to a B-agent to skip a particular stop.

[Regulate Speed Signal]: The CP-agent communicates to the B-agents to regulate their speed by increasing or decreasing it.

- Bus Agent :

  [Status Signal]: The B-agent sends its current status to the CP-agent: "OK" status, "Remove" status. The "OK" status indicates when the bus is currently available to perform any action, meanwhile the "Remove" status determines when the bus ends their activities on the route.

  [Traffic Status]: The B-agent communicates to the CP-agent about the traffic in the current position: "Heavy", "Medium", "Light".

4.3 Software Architecture

The system is developed in Java [11] with the libraries of Jason and Gurobi. Jason [26] integrates the multiagent system and Belief-Desire-Intentions model aspects to the program, and the Gurobi [20] library is used to solve the mathematical models that are used in the system. Java is used as the bridge between these two libraries and to support the simulation environment of the transportation network. Additionally, code developed in Python[48], and the Matplotlib library[25], were used to graph the result data given the multiagent system. Figure 4.5 shows the connections between the libraries.

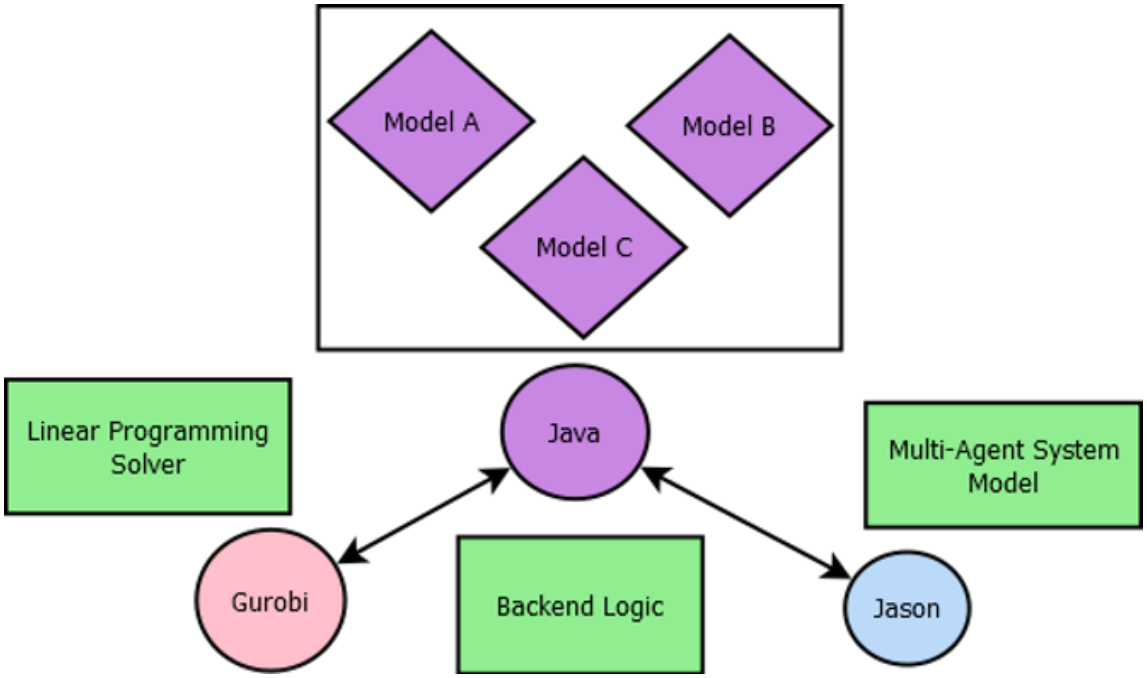


Figure 4.5: Software Architecture.

The system was developed in 3 phases, in each phase we integrate a new paradigm to the simulation until we have the full multiagent system. We construct the simulation platform and the MAS system incrementally to evaluate the characteristics of the environment and the strategies of the agents.

4.3.1 Phase 1: Bus System Simulation

Figure 4.6 shows a diagram of the architecture of the MAS system and simulation environment build in the first phase of development. In this phase, agents are "dummy" in the sense that they cannot decide on their actions. The job of the agents in this phase is to carry out the assigned tasks, from the mathematical model, in the simulation environment. The objective of this phase is to construct the baseline for our experimentation; that is, the mathematical model without any agent decision.

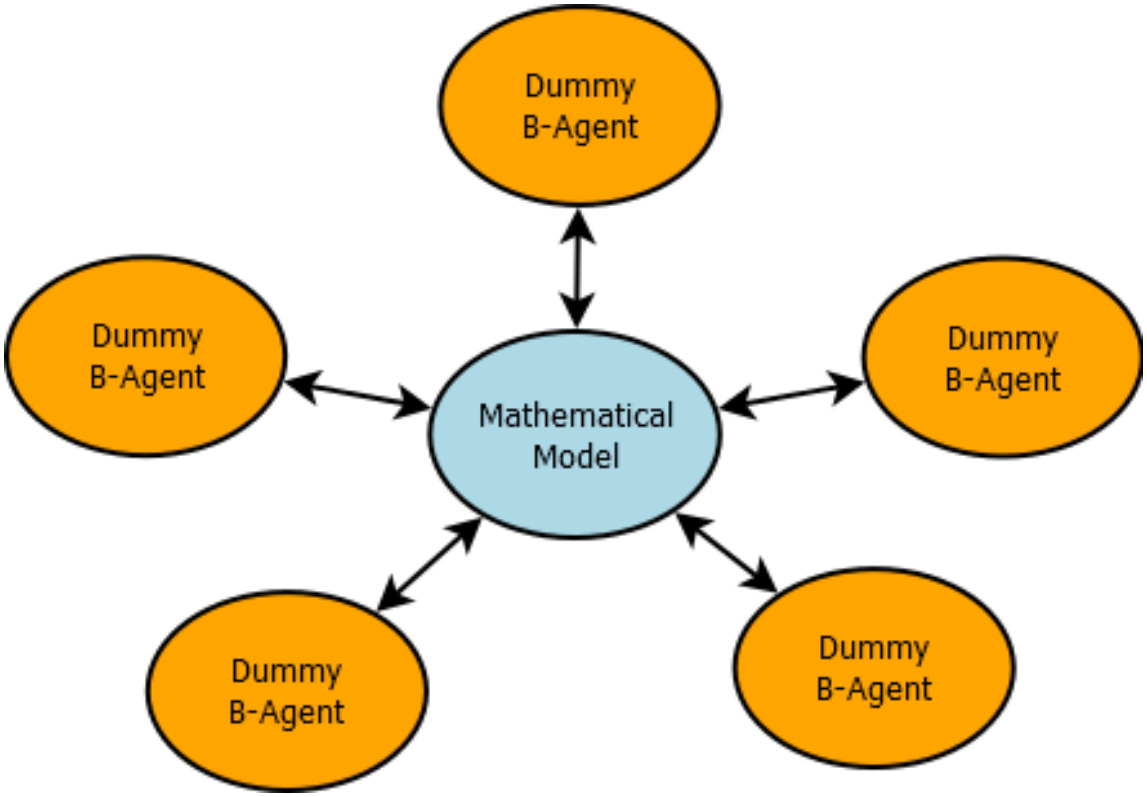


Figure 4.6: Phase 1: Bus System Simulation

4.3.2 Linear Model for Bus holding

The system has two different mathematical models for the bus holding: the first one modeled by Citlali Olvera [38] for a bus rapid transit system, and the second one is a modified version of the same model. This second version lacks of the restriction of overtake between buses, making it a case scenario of a single bus route system. In table 4.2 we can see the variables of this model and table 4.3 summarizes the mathematical model.

Linear Model Variables	Description
K	Number of buses
S	Number of stops
x_{kj}	Number of passengers that aboard the bus k at stop j
y_{kj}	Number of passengers that descends from the bus k at stop j
z_{kj}	Number of passengers that desire to aboard bus k at stop j
m_j	Time that bus j takes to reach next stop
d_{0k}	Distance between bus k and its last visited stop at time t0
$s(k)$	Last stop that bus k has visited at t0
c_{0s}	Number of passengers waiting at stop s at time t0
td_{ks}	Departure times of bus k at stop s
α	Proportionality constant that descends from the buses
β	Proportionality constant that enters to the buses
γ_j	Ratio of passengers that descends from the bus j
δ	Maximum holding time
λ_S	Passengers arrival time (Deterministic)
M	Big value auxiliary variable
R_{kj}, Q_{kj}	Auxiliary variables
I_{kj}	Gap between two consecutive buses k and k+1 at stop j
r_{kj1}, r_{kj2}	Auxiliary binary variables
Tol	Allowed time that buses have to reach or get behind a bus
Decision Variable	**Description**
h_{ks}	Holding times for each bus k at stop s

Table 4.2: Mathematical Model Variables.

Linear Model Restrictions	Description
$d_{kj} = d_{kj-1} + m_j + \beta x_{kj} + \alpha y_{kj} + h_{kj}$	Departure time from the buses
$d_{ks(k)+1} = d_{ks(k)} + m_{s(k)} - m_k^0 + \beta x_{ks(k)+1} + \alpha y_{ks(k)+1} + h_{ks(k)+1}$	Buses departure schedule
$h_{kj} < \delta$	Maximum holding time
$z_{kj} = w_j^0 + \lambda_j(d_{kj} - t^0) - \sum_{k' where s(k') \geq s(k)}^{k-1} x_{k'j}$	Number of passengers that desire to aboard each bus
$x_{kj} \leq z_{kj} - \sum x_{k'j}$	Departure before bus k and have not reach yet stop j
$x_{kj} \leq c - c_k^0 + \sum_{j'=s(k)+1}^{j}(y_{kj'} - x_{kj'})$	Departure before bus k and have not reach yet stop j
$x_{kj} \geq (z_{kj} - \sum x_{k'j}) - c(1 - r_{kj1}$	Departure before bus k and have not reach yet stop j
$x_{kj} \geq (c - c_k^0 + \sum_{j'=s(k)+1}^{j}(y_{kj'} - x_{kj'})) - M(1 - r_{kj2})$	Departure before bus k and have not reach yet stop j
$r_{kj1} + r_{kj2} = 1$	Departure before bus k and have not reach yet stop j
$y_{kj} = \gamma_j(c_k^0 + \sum_{j'=s(k)+1}^{j}(y_{kj'} - x_{kj'})$	Number of passengers that descends from stop j
$d_{kj} \geq d_{(k-1)j}$	Restriction of overtake between buses
$R_{kj} \geq 0$	Allowed time that buses have to reach or get behind a bus
$R_{kj} \geq -I_{kj} - Tol$	Allowed time that buses have to reach or get behind a bus
$Q_{kj} \geq 0$	Allowed time that buses have to reach or get behind a bus
$Q_{kj} \geq -I_{kj} - Tol$	Allowed time that buses have to reach or get behind a bus
Objective Function	**Description**
$min \sum_{k=1}^{\|K\|-1} \sum_{j=1}^{\|J\|-1} R_{kj} + Q_{kj}$	Aims to keep the same distances between buses

Table 4.3: Summary Mathematical Model

4.3.3 Phase 2: Centralized Multiagent System

Figure 4.7 shows the architecture of the system built during the second phase of the project. The architecture integrates one of the essential characteristics of a MAS, the communication. For the first time, we have the two types of agents represented in the architecture, the CP-Agent, and the B-Agent. In this phase, the CP-Agent is in charge of communicating actions to the B-Agents to reduce the Bus Bunching phenomenon. As mentioned earlier, the CP-Agent can command bus holding, skipping-stops, or deadheading actions to the B-Agents. B-Agents are not "dummy" anymore, because they can decide if they follow the requests of the CP-Agent. B-Agents are aware of the environment in this phase, so they have more accurate local information on the bus network to react accordingly. Contradicting a request from the CP-Agent could improve the performance of the system, given the dynamic factors of the environment. The objective of this phase is to analyze how communications and awareness impact the performance of the MAS.

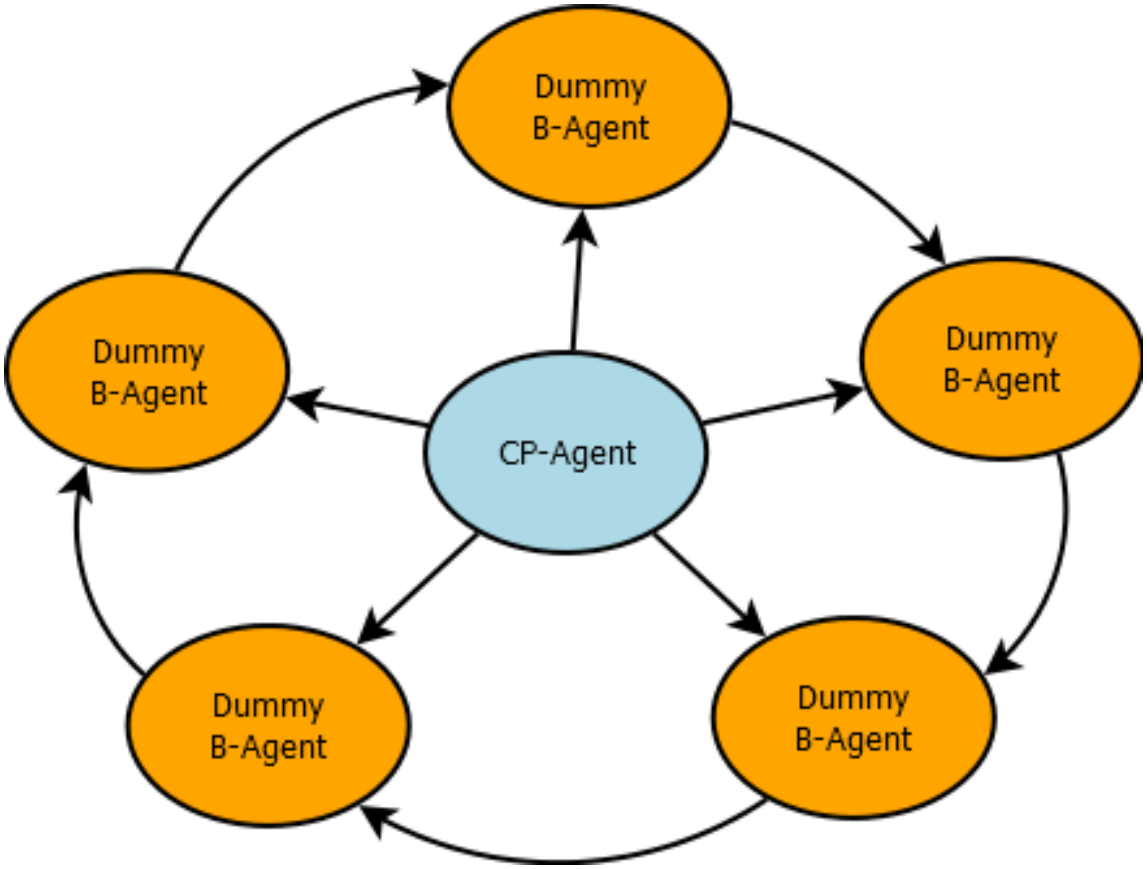


Figure 4.7: Phase 2: Centralized Multiagent System

4.3.4 Phase 3: Distributed Multiagent System

The last phase, seen in Figure 4.8, enables full functionality on B-Agents. In this phase, B-agents can make intelligent decisions based on what they perceive from their environment. In other words, they enact their plans to reduce the Bus Bunching present in the system. B-Agents can communicate information to the CP-Agent, which makes the communication bidirectional. Information from B-Agents helps to deliver a more accurate view of the system at any given point in time, which allows the mathematical models to generate better action plans.

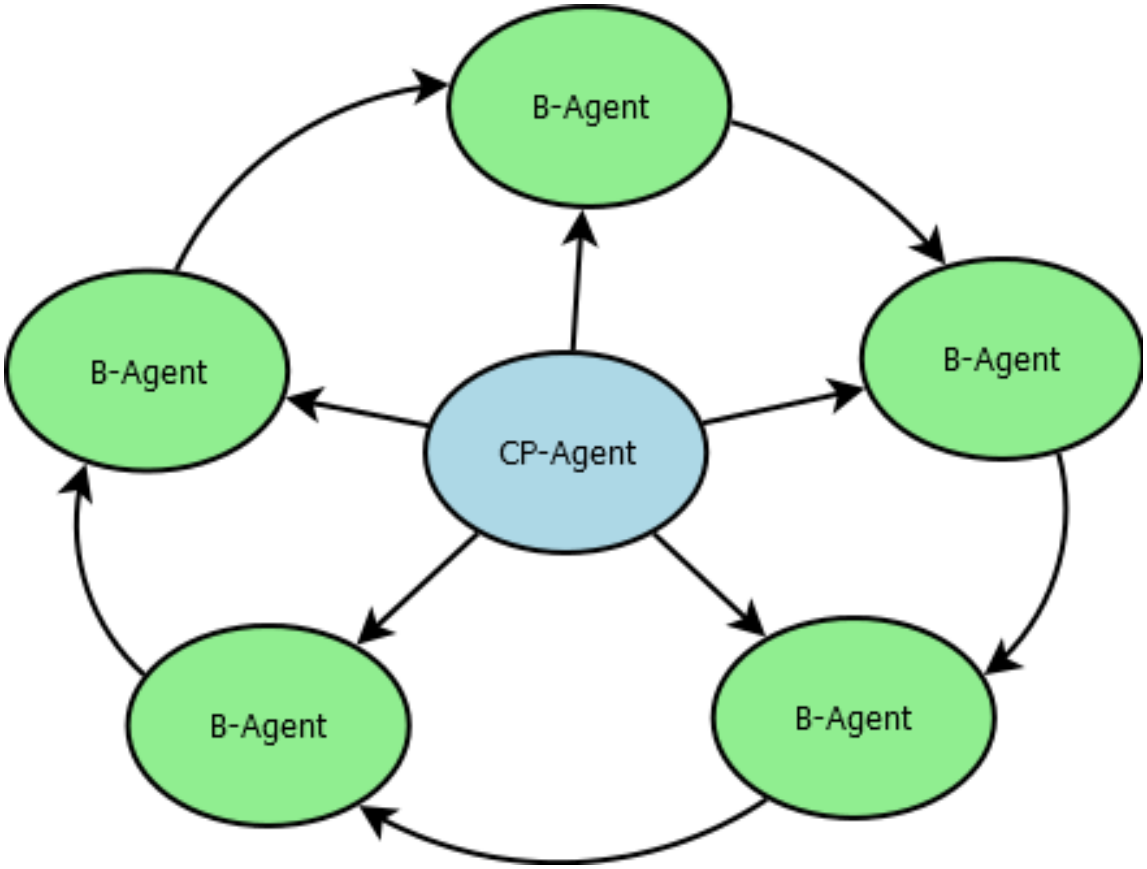


Figure 4.8: Phase 3: Distributed Multiagent System

Chapter 5

Phase 1: Bus System simulation

5.1 Introduction

The first phase of the system implements the bus system simulation platform and environment. It lacks any agent functionality. However, the objective of this phase is to evaluate different mathematical models to reduce bus bunching on the same simulation platform that our MAS will use. Therefore, this phase constructs the baseline system for comparing the performance of the MAS approach. In this chapter, we explain the algorithm for the simulation of the bus system and analyze the results of using two different mathematical models. The first model represents a bus rapid transit network, while the second one models a regular public bus transport system.

5.2 Initial Instance Configuration

The input to the simulation is a set of parameters. The parameters and their values are provided to the system in an initial configuration file. Table 5.1 shows the parameters and their descriptions. Notice that the set of parameters increase the flexibility of the simulation environment and overall system, since different scenarios could be analyzed and empirically evaluated.

Table 5.1: System Configuration.

Parameter	Description	Input
Snapshot time	Initial time	Integer
Number of Stops	Number of stops in the route	Integer
Bus Capacity	Capacity of each bus	Integer
Max Holding Time	Maximum holding time at the stops	Integer
Aboarding Time	Time that takes 1 passenger to aboard	Decimal
Descending Time	Time that takes 1 passenger to descend	Decimal
Release Time	Ticks needed to release the next bus to the route	Integer
Simulation End Time	Simulation end time	Integer
Bus Holding Period	Ticks to call the bus hold solver	Integer
Buses Overtake	If the overtake is allowed	Boolean
Circular Route	If a circular route is allowed	Boolean
Bus Holding Method	Name of the method defined in the code	String
Arriving Rates	Arriving rates at each stop	Array[Decimal]
Descending Rate	Descending rates at each stop	Array[Decimal]
Distance Between Stops	Distance between each stop	Array[Integer]
Buses Position	Initial position of each bus	Array[Integer]

5.3 Bus System Simulation

The simulation algorithm, shown in algorithm 1, consists in 7 main steps which are repeated until the simulation reaches its *endTime*.

Algorithm 1: Bus System Simulation

```
1  currentTime=0;
2  while currentTime ≤ EndTime do
3      if currentTime % releaseTime == 0 then
4          activeNextBus();
5      end
6      if currentTime % busHoldingPeriod == 0 then
7          solveBusHolding();
8      end
9      simulateStopArives();
10     simulateBusHolding();
11     simulateBusPosition();
12     simulateBusDescend();
13     simulateBusAboard();
14     currentTime=currentTime+1;
15 end
16
```

The *activeNextBus()* function, shown in algorithm 2 releases a new bus in the system until the system reaches its maximum capacity of buses. The release of every bus depends on the *releaseTime* parameter which is specified in the initial configuration.

Algorithm 2: activeNextBus()

```
1 for bus in listBuses do
2     if !bus.isActive then
3         b.isActive=true;
4         break;
5     end
6 end
```

The *solveBusHolding()* function solves the bus bunching of the current envi-

ronment snapshot. The model to solve the bus bunching and the *busHoldingPeriod* are parameters specified in the initial configuration.

The *simulateStopArrives()* function, shown in algorithm 3 simulates the arrival of people at every stop using a Poisson distribution function. The parameters of the Poisson distribution are specified for every stop in the initial configuration.

Algorithm 3: simulateStopArrives()

```
1 for stop in listStops do
2 |   stop.simulateArrive();
3 end
```

The *simulateBusHolding()* function, shown in algorithm 4 simulates the bus holding of every bus in the system that is currently at a stop. The bus holding time is generated by the solver. The maximum bus holding time is a parameter specified in the initial configuration.

Algorithm 4: simulateBusHolding()

```
1 for bus in listBuses do
2 |   if bus.isOnStop() and bus.mustHold then
3 |   |   b.isBusHolding=true;
4 |   else
5 |   |   b.isBusHolding=false;
6 |   end
7 end
```

The *simulateBusPosition()* function, shown in algorithm 5 simulates the bus

position every time. If the bus is performing a bus hold, then is doesn't move.

Algorithm 5: simulateBusPosition()

```
1 for bus in listBuses do
2     if !bus.isBusHolding() then
3         b.position=b.position+1;
4     end
5 end
```

The *simulateBusDescend()* function, shown in algorithm 6, simulates the descend of passengers from the bus. The initial configuration indicates the number of passengers that get off at each stop.

Algorithm 6: simulateBusDescend()

```
1 for bus in listBuses do
2     if bus.isOnStop() then
3         b.passengers*=b.descend;
4     end
5 end
```

The *simulateBusAboard()* function, shown in algorithm 7 simulates the aboard of passengers from the bus. The number of passengers that aboard in each stop is specified in the initial configuration.

Algorithm 7: simulateBusAboard()

```
1 for bus in listBuses do
2     if bus.isOnStop() then
3         b.passengers*=b.aboard;
4     end
5 end
```

5.4 Results

We designed three different scenarios for evaluation. The first two scenarios use artificial data. The third scenario uses real routing data from the Ecovia network from the city of Monterrey, Mexico [38, 16]. The objective of these tests is to analyze the impact that the mathematical model solution has on the simulation, based on average headway. To do so, we generate multiple evaluation instances, modifying the number of bus-holding solver calls.

Stops	10	10	10	10	10
Buses	7	7	7	7	7
Bus holding solver calls	0	2	4	6	15
Bus Alight	0.15	0.15	0.15	0.15	0.15
Bus Dwell	0.25	0.25	0.25	0.25	0.25
Overtake	FALSE	FALSE	FALSE	FALSE	FALSE
Circular	FALSE	FALSE	FALSE	FALSE	FALSE
Average Headway	1.379	1.379	1.386	1.379	1.379
Average Passengers Waiting Time	72.38	72.38	71.24	72.38	72.38

Table 5.2: Experiment 1: Bus Rapid Transit Instance 1

Stops	10	10	10	10	10
Buses	7	7	7	7	7
Bus holding solver calls	0	2	4	6	15
Bus Alight	0.25	0.25	0.25	0.25	0.25
Bus Dwell	0.35	0.35	0.35	0.35	0.35
Overtake	FALSE	FALSE	FALSE	FALSE	FALSE
Circular	FALSE	FALSE	FALSE	FALSE	FALSE
Average Headway	1.572	1.593	1.572	1.572	1.572
Average Passengers Waiting Time	70.14	69.78	70.14	70.14	70.14

Table 5.3: Experiment 2: Bus Rapid Transit Instance 2

We can see in Tables 5.2 and 5.3 the input parameters and aggregated results for a rapid bus transit network. Notice that increasing the times taken at bus stops (by alighting and dwelling) tends to increase the average headway between buses. Furthermore, it appears that the calls to the mathematical model do not have a significant impact either on the headway factor or the average passenger waiting times.

Stops	10	10	10	10	10
Buses	7	7	7	7	7
Bus holding solver calls	0	2	4	6	15
Bus Alight	0.15	0.15	0.15	0.15	0.15
Bus Dwell	0.25	0.25	0.25	0.25	0.25
Overtake	TRUE	TRUE	TRUE	TRUE	TRUE
Circular	TRUE	TRUE	TRUE	TRUE	TRUE
Average Headway	0.641	0.641	0.779	0.938	0.641
Average Passengers Waiting Time	84.38	84.38	83.79	83.08	84.38

Table 5.4: Experiment 3: Public Bus Transport Instance 1

Stops	10	10	10	10	10
Buses	7	7	7	7	7
Bus holding solver calls	0	2	4	6	15
Bus Alight	0.25	0.25	0.25	0.25	0.25
Bus Dwell	0.35	0.35	0.35	0.35	0.35
Overtake	TRUE	TRUE	TRUE	TRUE	TRUE
Circular	TRUE	TRUE	TRUE	TRUE	TRUE
Average Headway	1.007	1.007	1.007	0.945	0.945
Average Passengers Waiting Time	84.38	84.38	83.79	83.08	84.38

Table 5.5: Experiment 4: Public Bus Transport Instance 2

Tables 5.4 and 5.5 show the results of a bus route transit system where buses can overtake other units in the route. Notice this time that, when overtaking is allowed, the number of times we invoke the mathematical model does not have any effect in reducing the average passenger waiting time. However, increasing the time buses spend at bus stops tends to increase the headway. It makes sense since the longer buses stay idle higher the distance they get separated from other units.

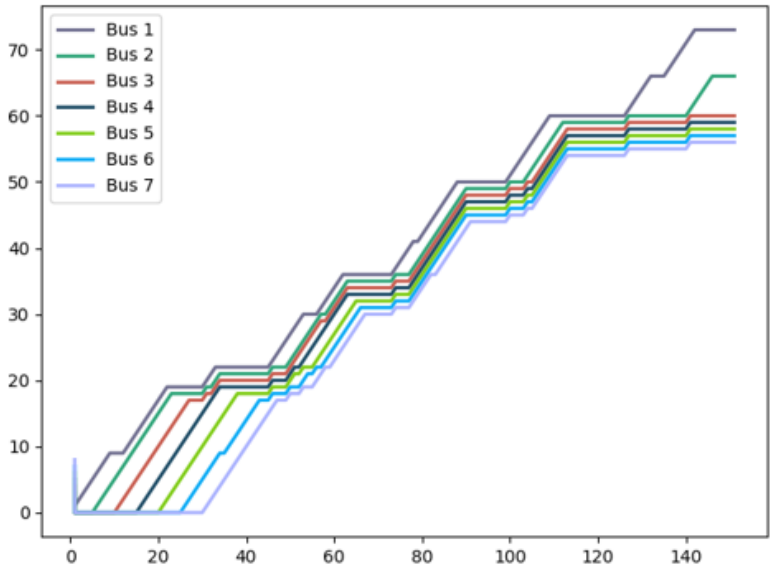
Stops	40	40	40	40	40
Buses	10	10	10	10	10
Bus holding solver calls	0	2	4	6	15
Bus Alight	0.07	0.07	0.07	0.07	0.07
Bus Dwell	0.07	0.07	0.07	0.07	0.07
Overtake	FALSE	FALSE	FALSE	FALSE	FALSE
Circular	FALSE	FALSE	FALSE	FALSE	FALSE
Average Headway	1.629	1.624	1.629	1.731	1.629
Average Passengers Waiting Time	111.28	112.43	111.28	109.76	111.28

Table 5.6: Experiment 5: Bus Rapid Transit Ecovia Instance 1

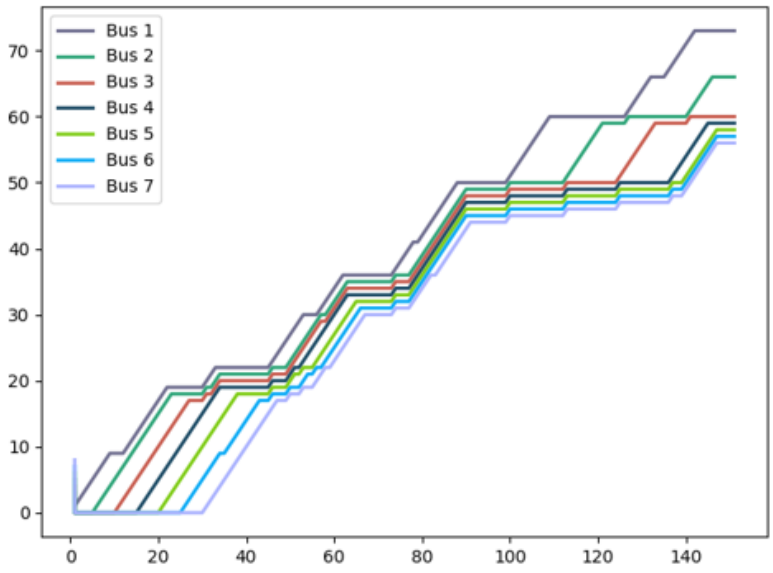
Stops	40	40	40	40	40
Buses	10	10	10	10	10
Bus holding solver calls	0	2	4	6	15
Bus Alight	0.14	0.14	0.14	0.14	0.14
Bus Dwell	0.14	0.14	0.14	0.14	0.14
Overtake	FALSE	FALSE	FALSE	FALSE	FALSE
Circular	FALSE	FALSE	FALSE	FALSE	FALSE
Average Headway	1.838	1.848	1.843	1.838	1.838
Average Passengers Waiting Time	115.46	112.42	113.03	115.46	115.46

Table 5.7: Experiment 6: Bus Rapid Transit Ecovia Instance 2

Table 5.6 and 5.7 display the results of the rapid transit network Ecovia Monterrey [16] using real system data [38]. Given that this is a rapid transit via, the alighting and dwelling times are short. This time the average headway is higher, as well as the average passenger waiting times.



(a) Experiment 1: 0 Calls

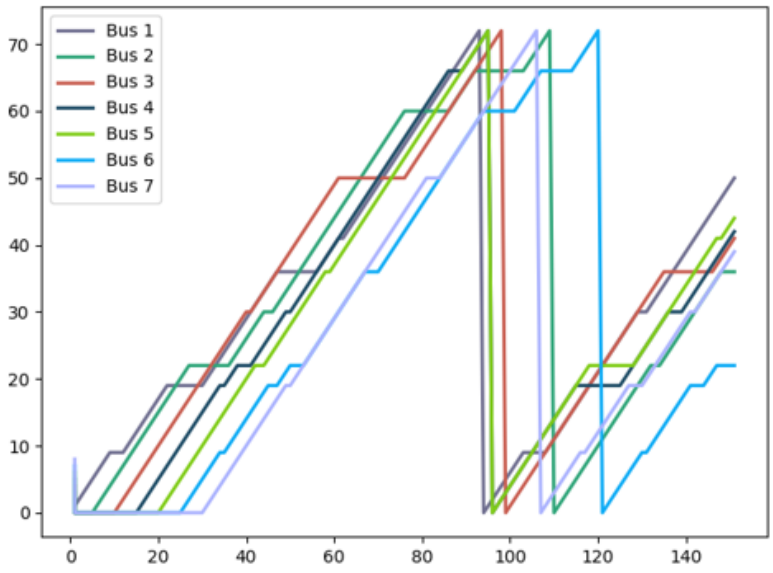


(b) Experiment 2: 2 Calls

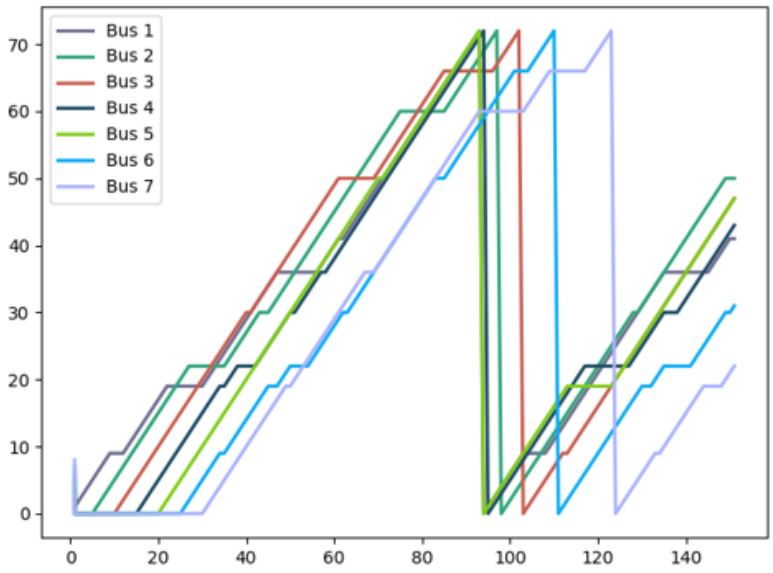
Figure 5.1: Public Bus Transport Experiments

Figures 5.1a, 5.1b, 5.2a, 5.2b, 5.3a and 5.3b show the average headway between bus units across the simulation horizon time. 5.1a and 5.1b correspond to the first set of experiments (i.e., the rapid bus transit network with artificial data). Notice that, at some point in the simulation, the bus bunching phenomenon becomes critical. Just

calling the mathematical model at least two times helps in increasing the headway between buses at the end of the planning horizon.



(a) Experiment 3: 0 Calls

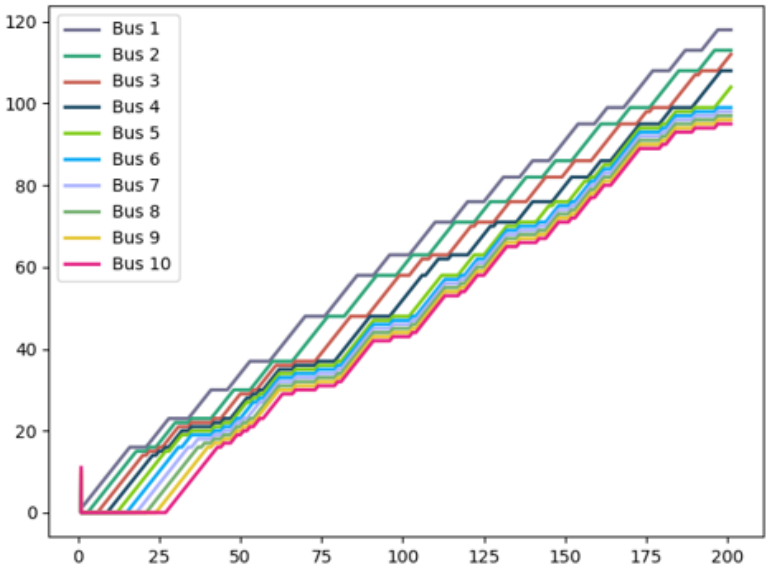


(b) Experiment 4: 6 Calls

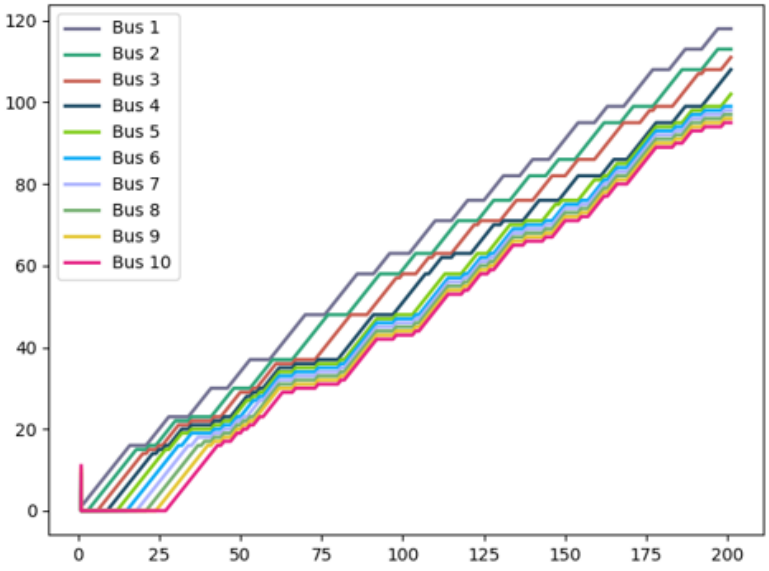
Figure 5.2: Rapid Bus Transit Experiments

Figures 5.1b and 5.2a, from the bus routing system, show similar behavior, independently of the number of calls to the mathematical model. Notice that, because

overtaking is allowed, buses start sooner a second journey.



(a) Experiment 5: 2 Calls



(b) Experiment 6: 6 Calls

Figure 5.3: ECOVIA Experiments

Chapter 6

Phase 2: Centralized Multiagent System

The second architecture involves the implementation of the centralized architecture of the multiagent system. In this model, the CP-agent can take decisions based on the environment information that B-agents sends. With the information received from the agents, the CP-agent commands each B-agent to perform a specific action to reduce the bunching that the buses create during the route. The decision of what actions will the agents perform depends on the headway tolerance range, which is the relative distance that each bus has with respect to its front and rear bus. Another important aspect that this architecture adds is that every B-agent keeps its state as dummy, this is, any action that the CP-agent commands to perform will be performed by the B-agent blindly (except if by any reason is incapable of doing it). This chapter will resume important theoretical aspects of the architecture and new experimental results using the same experimental design than the previous chapter.

6.1 Communication between agents

The communication between agents in the architecture is bidirectional. B-Agents send information to the CP-Agent regarding their current speed, position, and the number of passengers aboard. Meanwhile, the CP-Agent commands actions to B-Agents that could perform in their current state. It is important to note that the

CP-Agent will only send this information when every single one of the B-Agents of the route has transmitted their information since partial facts might lead to wrong decisions. In this architecture, the B-Agents are dummies in the sense that they do not reason or object to the decisions that the CP-Agent sends to them.

6.2 Headway Tolerance Range Metric

In the Centralized MAS architecture, the CP-Agent can carry decisions based on the information received from B-Agents. To do so, the CP-Agent needs to estimate how close are B-Agents to incur in bus bunching situations. We introduce, in this chapter, the Headway Tolerance Range (HTR) metric to prevent Bus Bunching. The HTR metric computes, for a given bus, a relative distance to the front and the rear bus adjacent to it. Then, we use this estimate to catalog the buses in the route with a risk of incurring in bus bunching. The metric first calculates the distance from the rear bus to the front. Then, we take the midpoint of this distance, which we believe is the ideal distance that needs to be maintained along the route to reduce the bus bunching phenomenon. The HTR metric uses this midpoint estimate to compute a tolerance range to the adjacent buses in the line. For the case when the HTR metric is 0% to the front bus in the line, then that implies that there is almost no distance between the current bus to the front bus, which is likely to induce the bus bunching phenomenon. On the other hand, if the HTR is 50%, then the current bus is actually at the midpoint, maintaining a stable headway balance between the neighboring buses. The CP-Agent uses the HTR metric to trigger actions to B-Agents. If a given B-Agent maintains a good HTR to the adjacent buses, then the CP-Agent might issue the order to maintain its speed. Figure 6.1 shows graphically the different scenarios that could occur between B-Agents and the potential actions to maintain a reasonable HTR estimate. Notice that if B-Agents have a small HTR percentage estimate to the front bus, then the CP-Agent may request it to slow down or to bus-hold at the next stop. On the other hand, a small HTR to the rear bus might trigger the action to speed up or skip-stop if the bus stop is ahead of it.

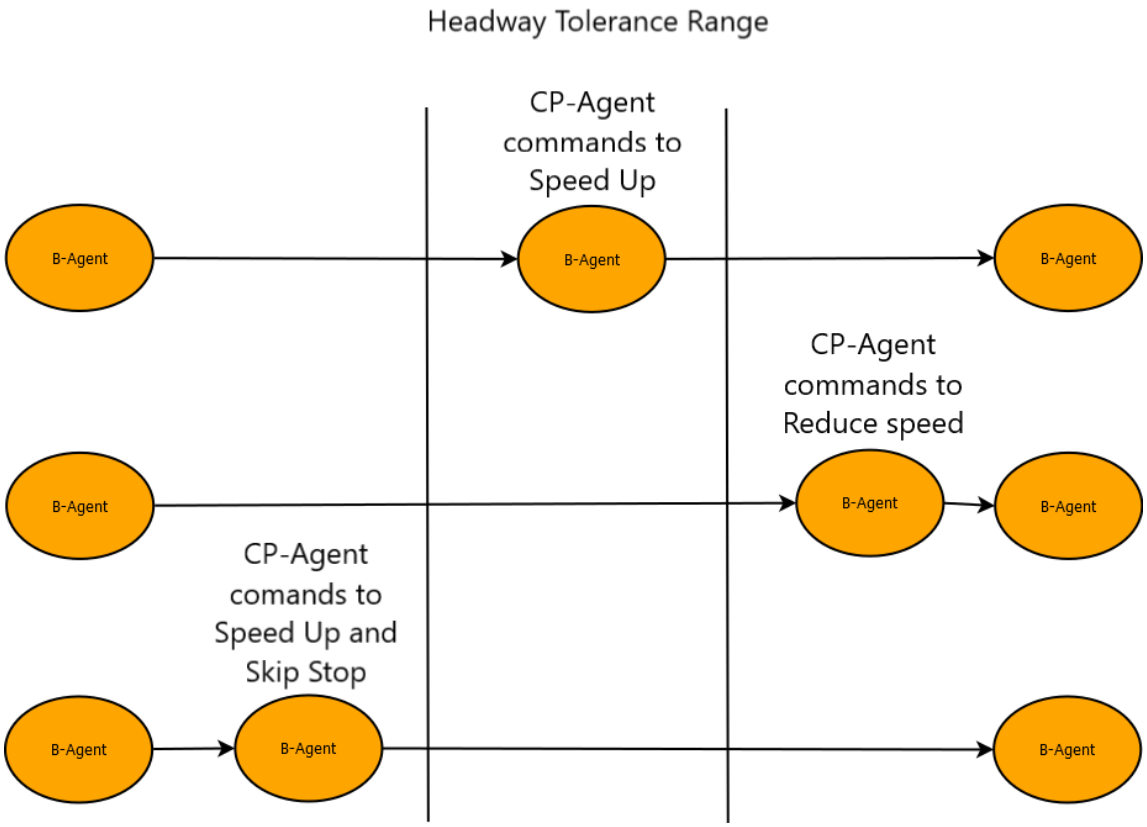


Figure 6.1: Control Point Agent planning.

6.3 Results

The results shown below in tables 6.1,6.2,6.3 are based on the same instances data that we used in the previous phase, with the addition of the HTR variable for the centralized model:

Stops	10	10	10
Buses	7	7	7
Bus holding solver calls	15	15	15
Bus Alight	0.15	0.15	0.15
Bus dwell	0.25	0.25	0.25
Overtake	TRUE	TRUE	TRUE
Circular	TRUE	TRUE	TRUE
HTR	10%	20%	30%
Average Headway	4.103	2.613	1.724
Average Passengers waiting time	47.84	57.52	64.33

Table 6.1: Experiment 1: Public Bus Transport

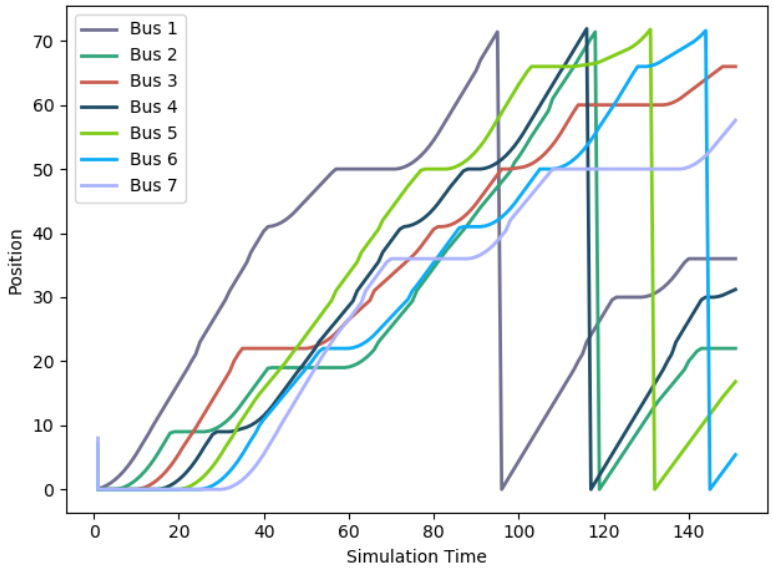
Stops	10	10	10
Buses	7	7	7
Bus holding solver calls	15	15	15
Bus Alight	0.15	0.15	0.15
Bus dwell	0.25	0.25	0.25
Overtake	FALSE	FALSE	FALSE
Circular	TRUE	TRUE	TRUE
HTR	10%	20%	30%
Average Headway	3.882	3.517	2.213
Average Passengers Waiting Time	50.24	53.67	60.13

Table 6.2: Experiment 2: Bus Rapid Transit

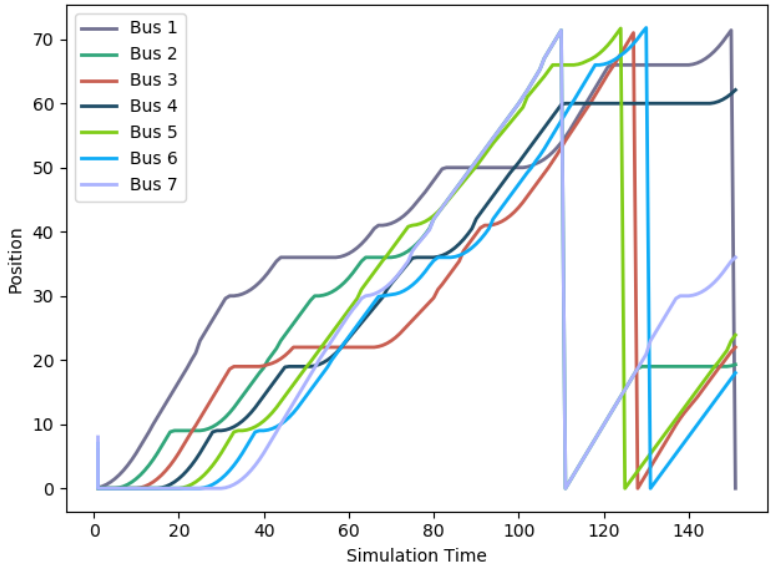
Stops	40	40	40
Buses	10	10	10
Bus holding solver calls	15	15	15
Bus Alight	0.14	0.14	0.14
Bus dwell	0.14	0.14	0.14
Overtake	FALSE	FALSE	FALSE
Circular	FALSE	FALSE	FALSE
HTR	10%	20%	30%
Average Headway	9.35	9.314	8.157
Average Passengers Waiting Time	97.14	97.74	106.82

Table 6.3: Experiment 3: Bus Rapid Transit Ecovia

Analyzing the results from the tables, we can notice that in the three different instances of bus routes, the lower the headway tolerance range is, the average headway increases while the average passenger waiting time decreases. Notice that the introduction of the HTR metric has a positive impact on reducing the waiting times of users with respect to the baseline (i.e., the mathematical solver). Figures 6.2a, 6.2b, 6.3a, 6.3b, 6.4a and 6.4b show the relationship of space (location) and time of buses during the simulation for the different transport networks of our empirical evaluation. For example, Figures 6.2a and 6.2b, correspond to rapid transit networks with artificial data and with 10% y 30% of HTR respectively. Notice that the introduction of the HTR metric helps us to maintain a more stable headway across the simulation. Smaller HTRs narrow the behavior of the buses, in other words, they have less flexibility of movement, which allows maintaining the order of operations during the route with a few exceptions (i.e., buses that start first tend to end first).

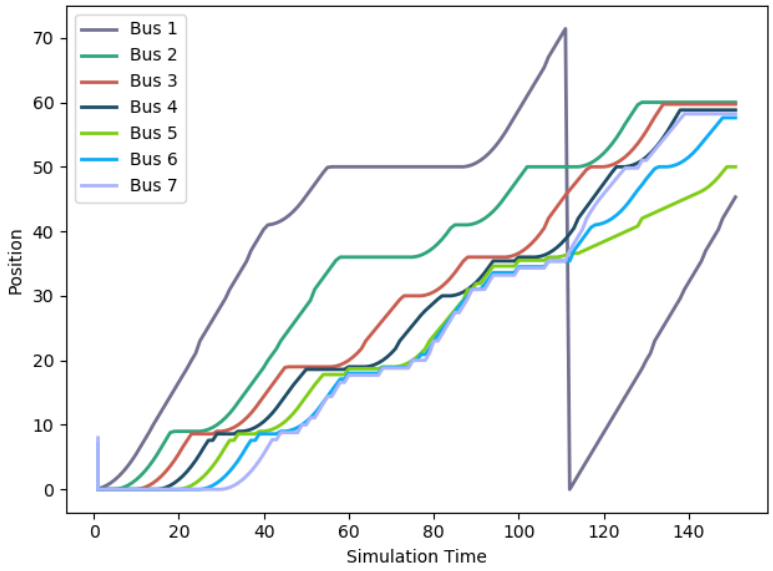


(a) PBT: 10% HTR

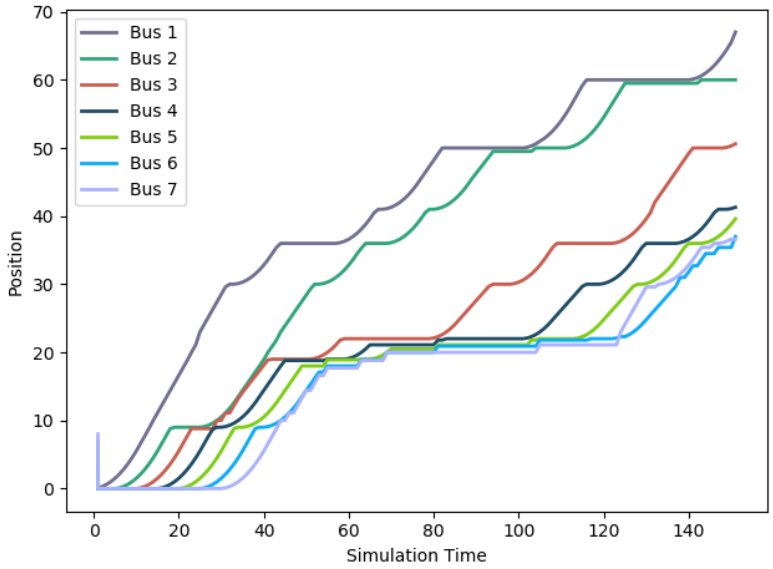


(b) PBT: 30% HTR

Figure 6.2: Public Bus Transport Experiments

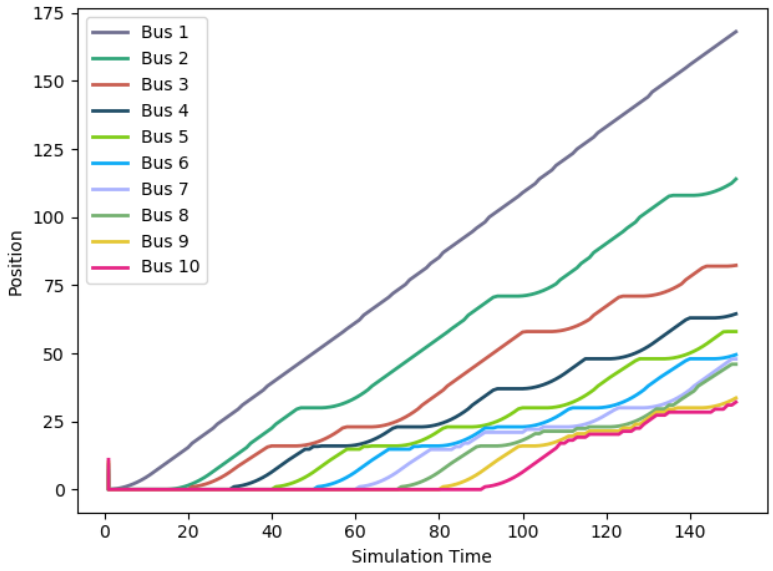


(a) BRT: 10% HTR

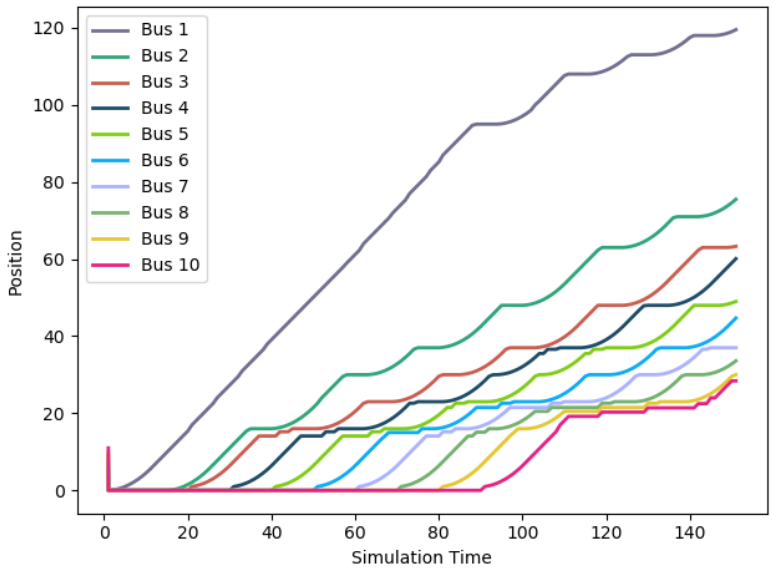


(b) BRT: 30% HTR

Figure 6.3: Rapid Bus Transit Experiments



(a) ECOVIA: 10% HTR



(b) ECOVIA: 30% HTR

Figure 6.4: ECOVIA Experiments

Chapter 7

Phase 3: Distributed Multiagent System

7.1 Introduction

The distributed model on the multiagent system will bring intelligent capabilities to the B-Agents to choose what actions to do on the route. In the previous phase, we added the headway tolerance range (HTR) algorithm for the CP Agent to select the actions B-Agents must perform to reduce bus bunching. In this phase, the B-Agent will be directly responsible for deciding the task it must execute based on the headway tolerance range. The deadheading strategy is added now in this phase, in which the CP-Agent can mark stops with a deadheading level, to decide how often buses should pass through them. However, B-Agents determine if they should follow the advice from the CP-Agent in terms of skipping or holding a particular stop. Experiments are made with similar instances from the previous two phases, with the additional configuration that the distributed model involves.

7.2 Headway Tolerance Range - B-Agents

One of the main problems, in the central model for the HTR metric, is that it takes time to develop a plan for all buses in the route because the CP-Agent needs to gather information from all of them to command actions. In the distributed model,

each B-Agent computes its HTR with respect to the front and rear buses taking into account the current state of the environment.

7.3 Deadheading Strategy

As described in chapter 1, the deadheading strategy involves marking some of the stops to reduce the number of buses that pass through them. The intuition behind this strategy is to provide more attention to highly demanded bus stops while decreasing the service times to those with lesser demand. We expect this strategy to attend more efficiently the passengers waiting at the bus stops. The distributed model supports the deadheading strategy by allowing B-Agents to broadcast information from the network to its peers and CP-Agent. For example, when B-Agents pass-through bus stops, they inform about the number of waiting passengers and those that board and descend. Besides, each B-Agent broadcasts its speed and location in the route. The CP-Agent keeps collecting this information, and every given deadheading time, it calculates the average number of passengers passing through a bus stop. Then, the CP-Agent uses this information to assign a deadheading level to each bus stop. For example, the bus stop with the lowest average of passengers will increase its deadheading level by one, while that one with the highest number will decrease its level by one. The deadheading level determines the number of buses that must skip a given stop before gets service. We intend to balance the passenger demand at bus stops to decrease the global waiting time of passengers.

7.4 Belief-Desire-Intention Model

The following codes corresponds to the B-Agent and CP-Agent belief-desire-intention model. These represents the behavior that the agents will have depending on the information that it has about the environment on the corresponding time.

```
/*
Bus agent

* bcBusPosition: broadcast the current bus position
* bcBusPassengers: broadcast the current passengers on the bus
* bcBusSpeed(bus,speed): broadcast the current bus speed
* bcBusNextStop(bus,stop): broadcast the next stop of the bus
* bcPeopleOnStop(stop,people): broadcast the number of people waiting
 ↪  at a given stop
* busHold(stop,time): holds the bus on a stop for a given time
* regulateSpeed: reegulates the speed of the bus
*/

+bcBusPosition(bus,position): bcBusPosition <-
 ↪  .broadcast(tell,busPosition(bus,position)).
+bcBusPassengers(bus,passengers): bcBusPassengers <-
 ↪  .broadcast(tell,passengersOnBus(bus,passengers)).
+bcBusSpeed(bus,speed): bcBusSpeed <-
 ↪  .broadcast(tell,busSpeed(bus,speed)).
+bcBusNextStop(bus,stop): bcBusNextStop <-
 ↪  .broadcast(tell,busNextStop(bus,stop)).
+bcPeopleOnStop(stop,people): bcPeopleOnStop <-
 ↪  .broadcast(tell,peopleOnStop(stop,people)).


+busHold(stop,time): true <- doBushold(stop,time).
+regulateSpeed(bus,speed): true <- doSpeedRegulation(bus,speed).
    \label{fig:busBDI}
```

```
/*
```

```
2  Control agent
3
4  . tellBH: tells a bus to do bus holding
5  . tellDH: tells a bus to mark the given stop with deadheading mark
6  . validateSkipStop: checks if the bus must skip the next stop
7  * start: keeps the simulation going
8  * finish: ends the simulation
9  */
10 !start.
11
12 +!tellBH(BUS,STOP,TIME): tellBH <-
   ↪  .send(BUS,tell,busHold(STOP,TIME)).
13 +!tellDH(BUS,STOP,TIME): tellDH <-
   ↪  .send(BUS,tell,deadhead(STOP,TIME)).
14
15 +!validateSkipStop(BUS): validateSkipStop(BUS) <-
   ↪  skipStop(BUS).abolish(validateSkipStop(BUS))!start.
16
17 +!start: start & tellBH(BUS,STOP,TIME) & not validateSkipStop(BUS) &
   ↪  not finish <- !tellBH(BUS,STOP,TIME).
18 +!start: start & validateSkipStop(BUS) & not tellBH(BUS,STOP,TIME) &
   ↪  not finish <- !validateSkipStop(BUS).
19 +!start: start & not validateSkipStop(BUS) & not
   ↪  tellBH(BUS,STOP,TIME) & not finish<- start;!start.
20 +!start:finish<- .print("Simulation end").
21     \label{fig:CPBDI}
```

7.5 Results

Stops	10	10	10	10
Buses	7	7	7	7
Bus holding solver calls	15	15	15	0
Bus Alight	0.15	0.15	0.15	0.15
Bus dwell	0.25	0.25	0.25	0.25
Overtake	TRUE	TRUE	TRUE	TRUE
Circular	TRUE	TRUE	TRUE	TRUE
HTR	10%	10%	10%	10%
Deadheading Time Tick	5	10	15	5
Average Headway	6.415	6.142	5.78	6.348
Average Passengers Waiting Time	40.77	34.88	31.47	42.4

Table 7.1: Experiment 1: Public Bus Transport

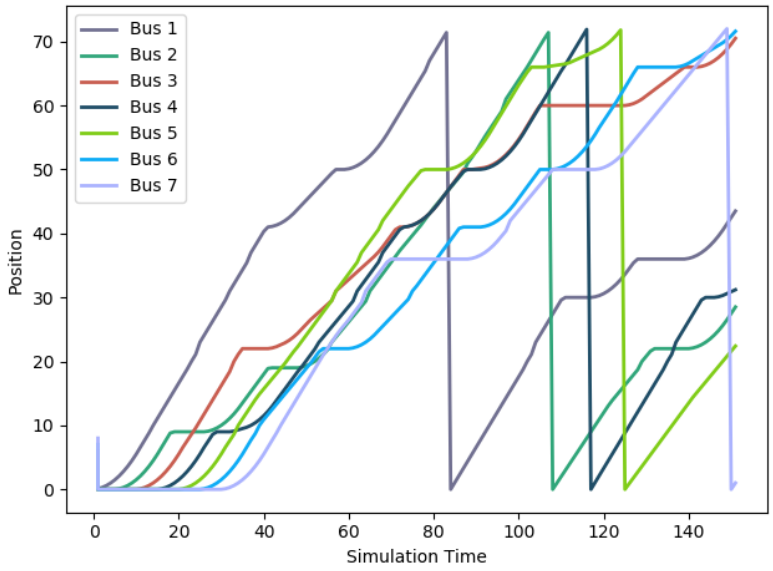
Stops	10	10	10	10
Buses	7	7	7	7
Bus holding solver calls	15	15	15	0
Bus Alight	0.15	0.15	0.15	0.15
Bus dwell	0.25	0.25	0.25	0.25
Overtake	FALSE	FALSE	FALSE	FALSE
Circular	TRUE	TRUE	TRUE	TRUE
HTR	10%	10%	10%	10%
Deadheading Time Tick	5	10	15	5
Average Headway	6.089	5.86	5.54	5.96
Average Passengers Waiting Time	36.45	32.47	29.75	36.79

Table 7.2: Experiment 2: Bus Rapid Transit

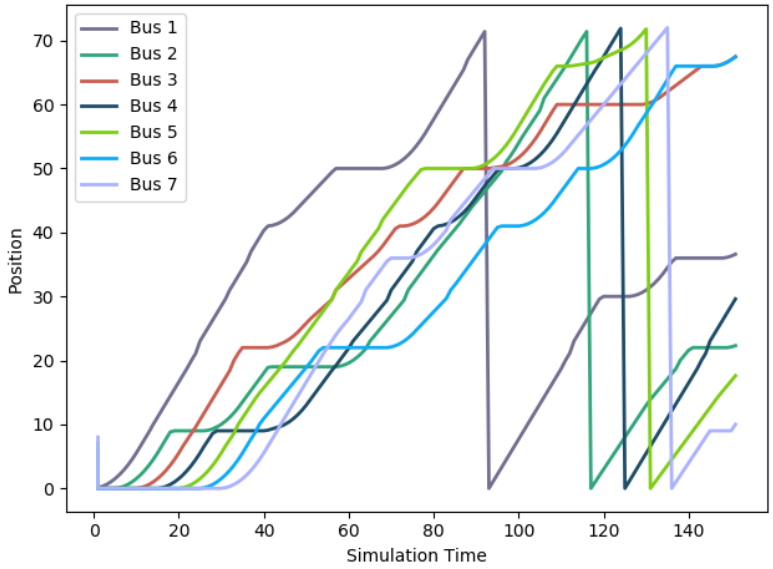
Stops	40	40	40	40
Buses	10	10	10	10
Bus holding solver calls	15	15	15	15
Bus Alight	0.14	0.14	0.14	0.14
Bus dwell	0.14	0.14	0.14	0.14
Overtake	FALSE	FALSE	FALSE	FALSE
Circular	FALSE	FALSE	FALSE	FALSE
HTR	10%	10%	10%	10%
Deadheading Time Tick	5	10	15	5
Average Headway	13.14	12.75	12.04	13.08
Average Passengers Waiting Time	89.24	84.2	81.91	90.12

Table 7.3: Experiment 3: Bus Rapid Transit Ecovia

With the previous results, we can observe that the average headway reduces when we have a lower deadheading time tick, implying that the strategy of deadheading is having a positive effect on decreasing the bus bunching. However, we can also notice that the lower the deadheading is, the higher is the average passenger waiting times. This phenomenon occurs since buses tend to skip more stops, letting users wait longer for the service. Figures 7.1a, 7.1b, 7.2a, 7.2b, 7.3a and 7.3b show the results from the instances with the higher average headway and the lowest average passenger waiting times.

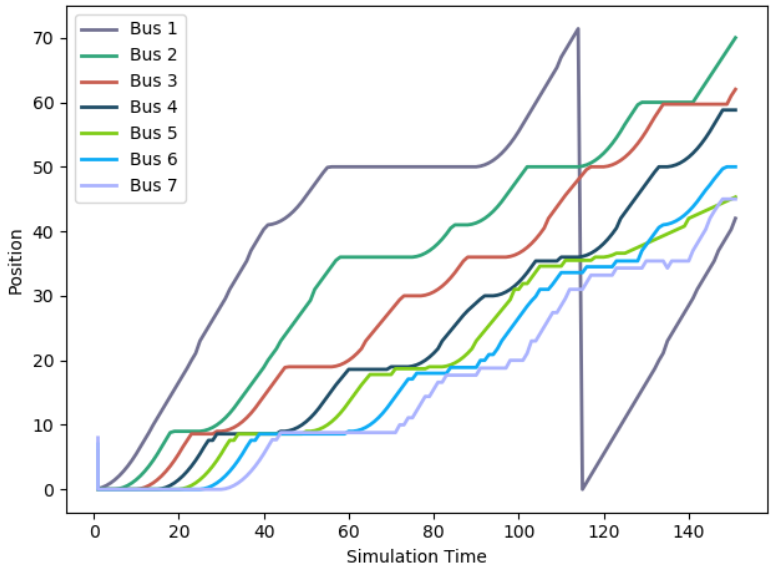


(a) PBT: 10% HTR

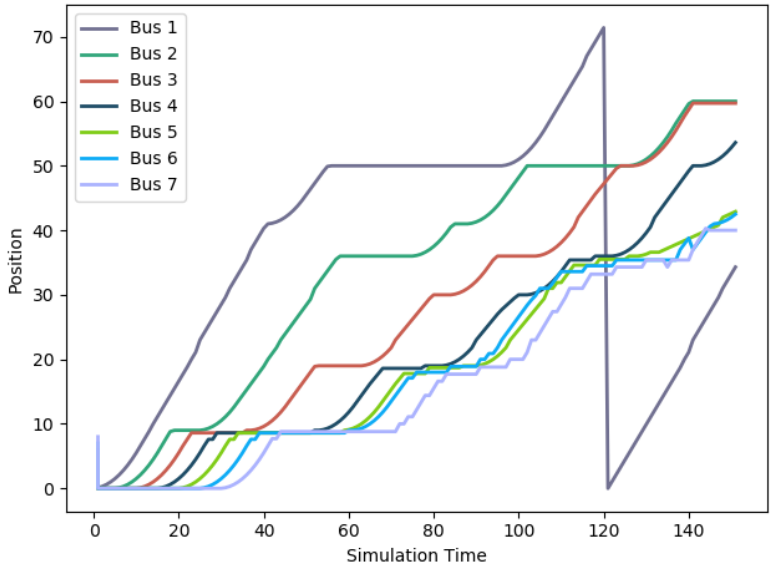


(b) PBT: 30% HTR

Figure 7.1: Public Bus Transport Experiments

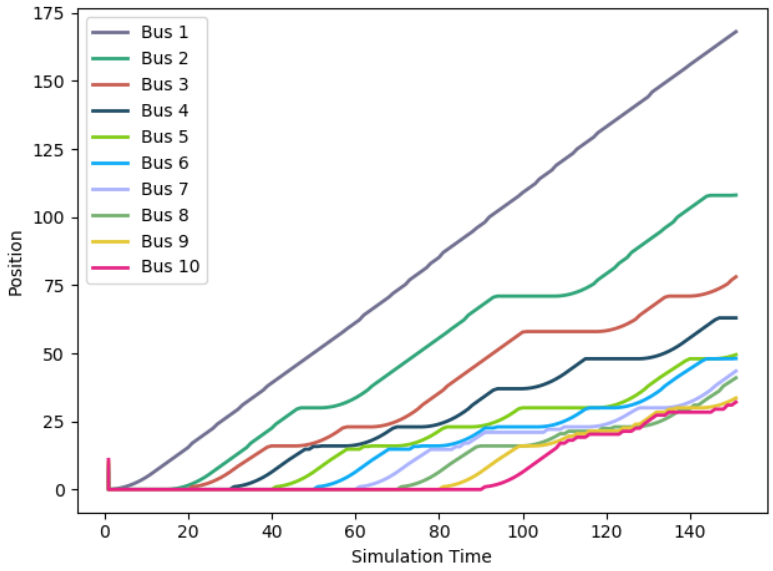


(a) BRT: 10% HTR

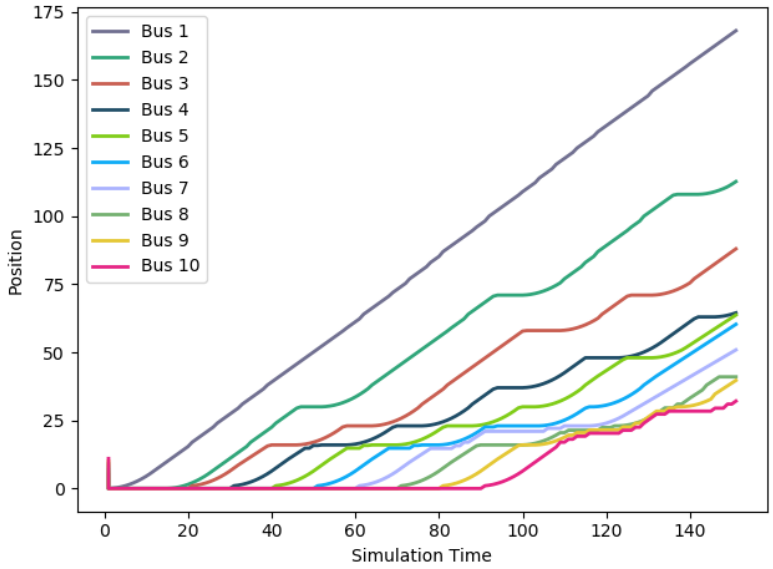


(b) BRT: 30% HTR

Figure 7.2: Rapid Bus Transit Experiments



(a) ECOVIA: 10% HTR



(b) ECOVIA: 30% HTR

Figure 7.3: ECOVIA Experiments

Chapter 8

Result analysis

Table 8.1 summarizes the results from the three architectures. The table presents the instance with the highest average headway. Notice that shorter headways between buses increase the probability of the Bus Bunching phenomenon, and in consequence, the average passenger waiting time. We can also observe that the introduction of a MAS improves the baseline results of simulation and the mathematical model. Furthermore, communications in distributed settings seem to play a prominent role in stabilizing the transport network while reducing even further the passenger waiting times.

		Average Headway	Average Passengers Waiting Time
Phase 1: Bus Simulation	**Public Bus Transport**	1.593	69.78
	Bus Rapid Transit	1.007	84.38
	Ecovia	1.848	112.42
Phase 2: Centralized Model	**Public Bus Transport**	4.103	47.84
	Bus Rapid Transit	3.882	50.24
	Ecovia	9.35	97.14
Phase 3: Distributed Model	**Public Bus Transport**	6.415	40.77
	Bus Rapid Transit	6.089	36.45
	Ecovia	13.14	89.24

Table 8.1: Summary of Results

With this data, let us retake and answer the objectives and hypothesis stated in chapter 1:

8.1 How can multiple strategies improve the solution of bus bunching problem?

The implementation of the strategies of bus holding, speed regulation, skip-stop and deadheading on phases 2 and 3 show a positive impact on reducing the bus bunching. Based on the results of phase 2, we noticed that using the strategies of bus holding, speed regulation and skip-stop gave us a better result than only using the bus holding strategy. In phase 3 we can see a better performance when we add the deadheading strategy to the system compared to only using bus holding, speed regulation and skip-stop. Since most of the strategies are controlled by the CP-Agent, the strategies do not conflict with each other by giving commands that might contradict the desires of the buses to separate. We conclude that the addition of multiple strategies to the multiagent system helps to reduce the bus bunching on the route.

8.2 How does the communication between agents in the multiagent system improve the solution of bus bunching problem?

Comparing the results of the same instances between phases 2 and 3, we can notice that the bunching between buses was reduced on the phase 3, the distributed model, compared to the phase 2, the centralized model. Since some of the strategies on the distributed model were now planned by each one of the buses' agents, the time for each bus to decide what is the best strategy to perform was drastically reduced since they only need the information of the frontal and rear bus; compared to the

centralized model in which the CP-Agent needs the information of every bus on the route to plan the action that every bus must perform. The difference of the planning time between both models impacts the performance of the bus route, since the bus route is constantly changing and the decisions that the CP-Agent commands to the buses' agents to perform might not be the best for the time it is commanded. We conclude that the communication between the agents have a positive impact on reducing the bus bunching between buses.

8.3 How does a centralized multiagent system performs against the bus bunching problem, compared to a linear programming model?

The linear programming model, as stated in chapter 3, is used to determine the bus holding waiting times on each of the stops for each bus. Comparing the results of the same instances between phases 1 and 2, we can notice a better performance using the centralized model with no calls to the bus holding solver compared to the phase 1 model in which only the bus holding solver calls were used to reduce the bus bunching. This is related to how the multiple strategies that were added to the centralized multiagent system gave a better performance on reducing the bus bunching: the CP-Agent has the capability, based on the headway tolerance range, to plan which action has a better impact on the route based on the information that is received by every agent. We conclude that the centralized multiagent system has a better performance on dealing with the bus bunching problem compared to the linear programming model.

8.4 How does a distributed multiagent system performs against the bus bunching problem, compared to a linear programming model?

Similarly to the case of the centralized model, the distributed model has multiple strategies to deal with the bus bunching problem, and since the headway tolerance range is now planned by each one of the buses' agents, it has an even greater performance than the linear programming model solution. We conclude that the distributed multiagent system do have a better performance on dealing with the bus bunching problem compared to the linear programming model.

8.5 How can a multiagent system improve the solution of bus bunching problem compared to linear programming?

All the characteristics of a multiagent system can be analyzed in performance based on the results of the phase 3 model. In phase 3, we specified the BDI model in which the buses' agents must decide if the action commanded by the CP-Agent does have a positive impact based on their local environment. The decisions made by the buses' agents and the headway tolerance range that each buses determine based on the frontal and rear buses both work together in the agents algorithm to reduce the bunching between the buses with more efficiency compared to using only the linear programming model to calculate the bus holding times. We conclude that the multiagent system does improves the solution of reducing the bus bunching compared to the linear programming model.

8.6 Does the distributed multiagent system performs better than the centralized multiagent system in dealing with the bus bunching problem?

Similarly to the question of the communication between agents, we can see that the instances of phase 3 had a better performance than the same instances of phase 2. The communication characteristic of the distributed model and the autonomy of the buses' agents to plan their actions did have an impact on reducing the bus bunching in the route. We conclude that the distributed multiagent system has a better performance than the centralized multiagent system, based on the models that were presented on this dissertation.

Chapter 9

Conclusions, Contributions & Future Work

Bus bunching is a critical problem in public transportation networks because it reduces the effectiveness of the route, increasing passenger waiting times. Previous work considers several strategies to reduce Bus Bunching, like control, optimization models, and multiagent systems. In this work, we presented three different architectures for dealing with Bus Bunching. The first one uses simulation and a mathematical model to control the traffic of buses in the route. The last two architectures use Multi-agent Systems. While the first MAS is a centralized model, the second one is a Distributed Model that leverages agent communications and problem-solving to reduce Bus Bunching. We showed that the Distributed MAS presented the best results in our empirical evaluation. We believe the communication exchange between the different types of agents in the system allows the MAS to respond more efficiently to the changes in the network environment.

9.1 Future Work

In this investigation, we conclude that the distributed model of the multiagent system has a better performance compared to the centralized model. With this in mind, we can start looking for distributed planning algorithms to determine which strategy and when the strategy must be applied to reduce the bus bunching on the system.

Another implementation that could be done is a prediction algorithm for the number of passengers that any stop may have in a specific time, and use this prediction to make a decision. We can also analyze how the strategies interact with each other, and develop a new algorithm based on the headway tolerance range to decide the actions that the buses can perform. As stated on chapter 2, the passengers waiting time has bigger value in the perspective of the users, when they are waiting for the bus at each stop compared to the time in which they are already in the bus. Thus, the waiting time should be analized with respect to each strategy in isolation. Some more configuration might be added in the future to test the effectiveness of the multiagent system in different scenarios. For example, adding random events that may disable a bus in the route, changing the rate that passengers arrive to the stops dynamically through time, changing the speed limit between stops through time, etc. With the possibility to add any linear model to calculate the holding times of the buses on the multiagent system, we can analyze how other models interact with the multiagent system algorithms to identify those models that have better performance with the agents. Since the results from the simulation are shown until the end of the simulation, we are planning to add a graphic interface that can represent the bus position, the actions performed, the stops positions and more characteristics that the agents perform so we can analyze graphically the results during the simulation. With the idea of the graphic interface of the simulation, we are also planning on having a log of the actions that every agent performs during the simulation, and add the function to reverse back actions during the simulation, so we can study the agents' behavior when adding new beliefs, desires or intentions.

9.2 Contributions

The main contribution in this research is the multiagent system architecture, which gave a better performance compared to the mathematical model tested in the simulations. This multiagent system has the option to enter some of the data as parameter to easily test various scenarios with the centralized and the distributed architecture.

Moreover, this system is freely available to use to anyone on a GitHub repository.

Appendix A

Appendix: BusiMA

BusiMA, he software used to simulate the bus system in this work was developed in java. Anyone can integrate their own models supported by Gurobi to formulate the data that the buses may use to perform their strategies, for example, the model used in this work returns the holding times that the buses must perform in each stop to reduce the bus bunching. This appendix shows is the requirements needed to run the Java project on a local computer and an example of an instance that can be used to run on the system.

A.1 Installation

The following software is required to run the BusiMA project locally:

- Java JDK version 8 (minimum) [26]
- Eclipse Framework (This investigation experiments were tested in the 2018 distribution) [18]
- Jason Library [26]
- Gurobi Library (A student license was used during the investigation experiments) [20]

Follow the installation instructions for each respective software, and then download the project from the GitHub repository [40]. Import this project into the Eclipse

environment and once imported, it will be ready to execute through the Jason environment.

A.2 Configuration instance example

The following code is an example of an instance of a bus route with its respective properties. This file is loaded through the main class BusiMA.java, in the variable folderName it must be specified the folder in which this .txt file is located. It must be the only file on that folder for it to load correctly.

```
1  @INSTANCE PROPERTIES@
2  #Snapshot Time= 0
3  #Number of Stops= 10
4  #Number of Buses= 7
5  #Bus capacity= 70
6  #Max Holding Time= 50
7  #Aboarding Time (per passenger)= 0.15
8  #Descending Time (per passenger)= 0.25
9  #Release Time of each bus= 5
10 #Simulation end time= 150
11 #Number of calls to the Bus Holding Solver= 15
12 #Buses overtake = false
13 #Circular route = true
14 #Bus Holding Method (based on the modelSolver.java options) = CMOT
15 #Headway Tolerance Range (Percentage from 0 to 1) = 0.3
16 #CP-Agent Planning tick = 5
17 #MA Architecture (CENTRALIZED/DISTRIBUTED) = CENTRALIZED
18 #Deadheading tick = 10
19
20 @STOPS PROPERTIES@
```

```
#Arriving Rate Values
0.4081|0.8871|0.9197|0.08417|0.998|0.02413|0.9262|0.953|0.186|0.0001
#Descending Ratio Values
0|0.1126|0.2291|0.838|0.97|0.918|0.934|0.0225|0.3925|1
#Distance between stops
9|10|3|8|6|5|9|10|6|6
#Number of passengers waiting in each stop at Snapshot Time
20|3|12|0|7|0|2|25|0|0
#Speed limit between stops
0.9|0.8|0.7|0.8|0.9|0.9|1|1|0.8|1

@BUSES PROPERTIES@
#Buses Index
1|2|3|4|5|6|7
#Buses current position
0|0|0|0|0|0|0
#Buses current passengers
0|0|0|0|0|0|0
#Last stop visited of each bus
1|1|1|1|1|1|1
```

A.3 Final notes

This software project is available in the following GitHub repository [40].

Bibliography

[1] APTA, «Public Transportation Facts», , 2020, URL `https://www.apta.com/news-publications/public-transportation-facts/`.

[2] Banks, J., *Discrete event system simulation*, Pearson Education India, 2005.

[3] Bartholdi III, J. J. y D. D. Eisenstein, «A self-coördinating bus route to resist bus bunching», *Transportation Research Part B: Methodological*, **46**(4), págs. 481–491, 2012.

[4] Bazaraa, M. S., J. J. Jarvis y H. D. Sherali, *Linear programming and network flows*, John Wiley & Sons, 2011.

[5] Ben-Akiva, M. E., S. R. Lerman y S. R. Lerman, *Discrete choice analysis: theory and application to travel demand*, tomo 9, MIT press, 1985.

[6] Bordini, R. H., H. J. Fred. y M. J. Wooldridge, *Programming multi-agent systems in AgentSpeak using Jason*, J. Wiley Sons, 2007.

[7] Cao, Z. y A. A. Ceder, «Autonomous shuttle bus service timetabling and vehicle scheduling using skip-stop tactic», *Transportation Research Part C: Emerging Technologies*, **102**, págs. 370–395, 2019.

[8] Ceder, A. y H. I. Stern, «Deficit function bus scheduling with deadheading trip insertions for fleet size reduction», *Transportation Science*, **15**(4), págs. 338–363, 1981.

[9] CHEN, C., W. CHEN y Z. CHEN, «A Multi-Agent Reinforcement Learning approach for bus holding control strategies.», *Advances in Transportation Studies*, 2015.

[10] CHEN, W., K. ZHOU y C. CHEN, «Real-time bus holding control on a transit corridor based on multi-agent reinforcement learning», en *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, págs. 100–106, 2016.

[11] CORPORATION, O., «Java Software Platform», , 2020, URL `https://www.java.com/es/`.

[12] CORTÉS, C. E., S. JARA-DÍAZ y A. TIRACHINI, «Integrating short turning and deadheading in the optimization of transit services», *Transportation Research Part A: Policy and Practice*, **45**(5), págs. 419–434, 2011.

[13] DAGANZO, C. F. y J. PILACHOWSKI, «Reducing bunching with bus-to-bus cooperation», *Transportation Research Part B: Methodological*, **45**(1), págs. 267–277, 2011.

[14] DUBÉ, L., B. H. SCHMITT y F. LECLERC, «Consumers' Affective Response to Delays at Different Phases of a Service Delivery 1», *Journal of Applied Social Psychology*, **21**(10), págs. 810–820, 1991.

[15] EBERLEIN, X. J., N. H. WILSON, C. BARNHART y D. BERNSTEIN, «The real-time deadheading problem in transit operations control», *Transportation Research Part B: Methodological*, **32**(2), págs. 77–100, 1998.

[16] ECOVIA, «Sistema de transito rapido Ecovia», , 2020, URL `http://ecovia.nl.gob.mx/`.

[17] EVERITT, B. y A. SKRONDAL, *The Cambridge dictionary of statistics*, tomo 106, Cambridge University Press Cambridge, 2002.

[18] FOUNDATION, E., «Eclipse IDE», , 2020, URL `https://www.eclipse.org/`.

[19] GKIOTSALITIS, K. y O. CATS, «Multi-constrained bus holding control in time windows with branch and bound and alternating minimization», *Transportmetrica B: Transport Dynamics*, **7**(1), págs. 1258–1285, 2019.

[20] GUROBI OPTIMIZATION, L., «Gurobi Optimizer Reference Manual», , 2020, URL `http://www.gurobi.com`.

[21] HE, S.-X., S.-D. LIANG, J. DONG, D. ZHANG, J.-J. HE y P.-C. YUAN, «A holding strategy to resist bus bunching with dynamic target headway», *Computers & Industrial Engineering*, **140**, pág. 106 237, 2020.

[22] HERNÁNDEZ, J. Z., S. OSSOWSKI y A. GARCIA-SERRANO, «Multiagent architectures for intelligent traffic management systems», *Transportation Research Part C: Emerging Technologies*, **10**(5-6), págs. 473–506, 2002.

[23] HERNÁNDEZ-LANDA, L. G., M. L. MORALES-MARROQUÍN, R. S. NIGENDA y Y. Á. RÍOS-SOLÍS, «Linear bus holding model for real-time traffic network control», en *Applied Simulation and Optimization*, Springer, págs. 303–319, 2015.

[24] HUANG, Q., B. JIA, R. JIANG y S. QIANG, «Simulation-based optimization in a bidirectional A/B skip-stop bus service», *Ieee Access*, **5**, págs. 15 478–15 489, 2017.

[25] HUNTER, J. D., «Matplotlib: A 2D graphics environment», *Computing in Science & Engineering*, **9**(3), págs. 90–95, 2007.

[26] JOMI F. HÜBNER, R. H. B., «Jason a Java-based interpreter for an extended version of AgentSpeak», , 2020, URL `https://www.java.com/es/`.

[27] KIEU, L.-M., A. BHASKAR y E. CHUNG, «Insights into the bus bunching problem: a multi-agent simulation approach», *Informe técnico*, 2016.

[28] KOLP, M., P. GIORGINI y J. MYLOPOULOS, «Multi-agent architectures as organizational structures», *Autonomous Agents and Multi-Agent Systems*, **13**(1), págs. 3–25, 2006.

[29] LETKOWSKI, J., «Applications of the Poisson probability distribution», en *Proc. Acad. Business Res. Inst. Conf*, págs. 1–11, 2012.

[30] LI, S., R. LIU, L. YANG y Z. GAO, «Robust dynamic bus controls considering delay disturbances and passenger demand uncertainty», *Transportation Research Part B: Methodological*, **123**, págs. 88–109, 2019.

[31] LISCO, T., *THE VALUE OF COMMUTERS'TRAVEL TIME: A STUDY IN URBAN TRANSPORTATION, University of Chicago*, Tesis Doctoral, Ph. D. dissertation, 1967.

[32] MODICA, L. y L. STENNETH, «Real-time vehicle spacing control», US Patent 9,659,492, mayo 23 2017.

[33] MOOVIT, «SURVEY: U.S. Commuters Wait Approximately 40 Minutes per Day for Public Transit, Costing Them 150 Hours per Year», , 2014, URL `https://www.globenewswire.com/news-release/2014/12/09/1126354/0/en/SURVEY-U-S-Commuters-Wait-Approximately-40-Minutes-per-Day-for-Public-Transi html`.

[34] NESHELI, M. M., A. CEDER y V. A. GONZALEZ, «Real-time public-transport operational tactics using synchronized transfers to eliminate vehicle bunching», *IEEE Transactions on Intelligent Transportation Systems*, **17**(11), págs. 3220–3229, 2016.

[35] NEUMANN, A. y K. NAGEL, «Avoiding bus bunching phenomena from spreading: A dynamic approach using a multi-agent simulation framework», *VSP WorkingPaper10-08, TU Berlin, Transport Systems Planning and Transport Telematics. See www. vsp. tu-berlin. de/publications*, 2010.

[36] NEWELL, G. F. y R. B. POTTS, «Maintaining a bus schedule», en *Australian Road Research Board (ARRB) Conference, 2nd, 1964, Melbourne*, tomo 2, 1964.

[37] NIU, H., «Determination of the skip-stop scheduling for a congested transit line by bilevel genetic algorithm», *International journal of computational intelligence systems*, **4**(6), págs. 1158–1167, 2011.

[38] OLVERA TOSCANO, C. M., *Modelo matemático e implementación de un simulador basado en un sistema de transporte urbano.*, Tesis Doctoral, Universidad Autónoma de Nuevo León, 2018.

[39] OSUNA, E. y G. F. NEWELL, «Control strategies for an idealized public transportation system», *Transportation Science*, **6**(1), págs. 52–72, 1972.

[40] PATLAN, J., «Bus Bunching MAS Project», , 2020, URL `https://github.com/JAPatlanC/BusBunchingMAS`.

[41] PILACHOWSKI, J. M., *An approach to reducing bus bunching*, Tesis Doctoral, UC Berkeley, 2009.

[42] ROSS, S. M., *A course in simulation*, Prentice Hall PTR, 1990.

[43] RUSSELL, S. y P. NORVIG, «Artificial intelligence: a modern approach», , 2002.

[44] SALONER, D., Y. XUAN, J. ARGOTE y C. DAGANZO, «Automated system for preventing vehicle bunching», US Patent 9,224,295, diciembre 29 2015.

[45] SHEHORY, O. M., *Architectural properties of multi-agent systems*, Carnegie Mellon University, The Robotics Institute, 1998.

[46] UITP, «Urban Public Transport in the 21st Century: Statistics Brief», , 2017, URL `https://www.uitp.org/sites/default/files/cck-focus-papers-files/UITP_Statistic%20Brief_national%20PT%20stats.pdf`.

[47] VAN AART, C., *Organizational Principles for Multi-Agent Architectures*, Springer Science & Business Media, 2004.

[48] VAN ROSSUM, G. y F. L. DRAKE, *Python 3 Reference Manual*, CreateSpace, Scotts Valley, CA, 2009.

[49] WALCK, C. *et al.*, «Hand-book on statistical distributions for experimentalists», *University of Stockholm*, **10**, 2007.

[50] WANG, J. y L. SUN, «Dynamic holding control to avoid bus bunching: A multi-agent deep reinforcement learning framework», *Transportation Research Part C: Emerging Technologies*, **116**, pág. 102 661, 2020.

[51] WANG, T., «Flexible fare bus framework to reduce bus bunching», US Patent 9,842,375, diciembre 12 2017.

[52] WEISS, G., *Multiagent systems: a modern approach to distributed artificial intelligence*, MIT press, 1999.

[53] WU, W., R. LIU y W. JIN, «Modelling bus bunching and holding control with vehicle overtaking and distributed passenger boarding behaviour», *Transportation Research Part B: Methodological*, **104**, págs. 175–197, 2017.

[54] XUAN, Y., J. ARGOTE y C. F. DAGANZO, «Dynamic bus holding strategies for schedule reliability: Optimal linear control and performance analysis», *Transportation Research Part B: Methodological*, **45**(10), págs. 1831–1845, 2011.

[55] YU, B., Z. YANG y S. LI, «Real-time partway deadheading strategy based on transit service reliability assessment», *Transportation Research Part A: Policy and Practice*, **46**(8), págs. 1265–1279, 2012.

[56] ZHAO, J., S. BUKKAPATNAM y M. M. DESSOUKY, «Distributed architecture for real-time coordination of bus holding in transit networks», *IEEE Transactions on Intelligent Transportation Systems*, **4**(1), págs. 43–51, 2003.

[57] ZHOU, L., Y. WANG y H. CUI, «The Bus Auxiliary Driving System Based on Multi-Agent Strategy.», *JSW*, **12**(9), págs. 722–731, 2017.

Index

Ficha autobiográfica

Jesús Ángel Patlán Castillo

Candidato para el grado de Maestría en Ingeniería
con especialidad en Ingeniería de Sistemas

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Tesis:

Solving the Bus Bunching Problem with a Multiagent System

Nací el 28 de diciembre de 1995 en Monterrey, Nuevo León. Soy egresado de la carrera de ingeniero administrador de sistemas de la Facultad de Ingeniería Mecánica y Eléctrica, actualmente curso mi noveno semestre en la licenciatura en matemáticas en la Facultad de Ciencias Físico Matemáticas. Mis intereses principales van por la rama de las matemáticas, con un gusto en particular en materias como teoría de la medida, análisis matemático, teoría de números; y también tengo el gusto por el área de la inteligencia artificial en sistemas multiagentes, la cual es la rama a la que dedico mis estudios.